NOS/VE Global File Management : Assign and Free FDE Entries

```
  3 MODULE gfm$file_table_manager;
  4 {
  5 { PURPOSE:
  6 {   This module contains procedures for assigning and freeing file descriptor table entries.
  7 {
  8 { DESIGN:
  9 {   File descriptors are kept in either mainframe wired or in job fixed. They are kept in
 10 {   an array at a large offset; they are NOT part of the heap. The address of the array and
 11 {   structures used to manage the array are defined in GFC$CONSTANTS.
 12 {   The tables used to manage FDEs are kept in mainframe/wired/job fixed at offset
 13 {   GFC$FDE_CONTROL_TABLE_BASE. A multi-level index structure is used to manage assignment
 14 {   of entries.
 15 {     o A packed array of 65535 booleans (organized as array [0 .. 1023] of words) is used
 16 {       to manage assignment of indivual FDEs. If bit <n> of the array is FALSE, then
 17 {       FDE number <n> is free; if bit <n> is TRUE then FDE number <n> is assigned.
 18 {     o In order to improve search time to find an available entry, a second level
 19 {       index (packed array [0 .. 1023] of booleans) is kept to indicate which words in
 20 {       the lower level table have available entries. If bit <m> of this array is FALSE then
 21 {       word <m> of the lower level table contains free entries.
 22 {     o A first level index is maintained to indicate which words in the second level
 23 {       index have free entries.
 24 {     o A free entry can be located by examining 3 words; first level index word, second level index
 25 {       word, in_use word. A hardware instruction (CNIF - convert_integer_to_float) is used
 26 {       that will give the bit number of the first "zero" bit in a word.
 27 {
 28 {
 29 { NOTE:
 30 {     o The table structure will support assignment of up to 262K entries. Only 65K are
 31 {       currently used because  SFID.INDEX is only 2 bytes. Increasing this to 3 bytes
 32 {       would cause incompatibilities.
 33 {
 34 {     o Create an SCL variable GFC$TEST_HARNESS to compile a standalone
 35 {       version of this module that can be used for testing.
 36 {
 37
 38   CONST
 39     gfc$debug = TRUE;
 40
```


NOS/VE Global File Management : Assign and Free FDE Entries
Global Declarations Referenced by this MODULE

```
o   689
o   690   PROCEDURE [INLINE] osp$clear_mainframe_sig_lock
o   691     (VAR lock: ost$signature_lock);
o   692
o  2377
o  2378   PROCEDURE [XREF] osp$fatal_system_error (error_message: string ( * );
o  2379         status: ^ost$status);
o  2380   PROCEDURE [INLINE] osp$set_mainframe_sig_lock
o  2381     (VAR lock: ost$signature_lock);
o  2382
o  2425
o  2426 {System page size.}
o  2427
o  2428   VAR
o  2429     osv$page_size: [XREF] ost$page_size;
o  2430
```

NOS/VE Global File Management : Assign and Free FDE Entries
FDE Initialization value

```
o  2435 {
o  2436 { The following table defines the initial value of a newly assigned FDE. Callers
o  2437 { of gfp$assign_fde may depend on values defined in this table. Values in the table
o  2438 { specified as "*" normally are filled in by the caller.
o  2439 {
o  2440 {
o  2441
o  2442
o  2443 ?? FMT (FORMAT := OFF) ??
o  2444
o  2445   VAR
o  2446     initial_fde_entry: [READ, oss$mainframe_paged_literal] gft$file_descriptor_entry :=
o  2447
o  2448           [*,                                              {job_lock - not locked
o  2449            [FALSE, 0],                                     {monitor_interlock
o  2450            [FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE], {flags}
o  2451  {*}       [0, osc$cyber_180_model_unknown, 1980, 1, 1, 0, 0, 0, 0, 0], {global_file_name
o  2452            NIL,                                            {file hash thread
o  2453            0,                                              {attached_in_write_count
o  2454            0,                                              {attach_count
o  2455            0,                                              {open_count
o  2456            gfc$fk_unnamed_file,                            {file_kind
o  2457            *, {Random 1 .. 250}                            {file_hash
o  2458            [0, FALSE, [0, 0]],                             {segment_lock
o  2459            0,                                              {asti
o  2460            0,                                              {eoi_byte_address
o  2461            mmc$eoi_actual,                                 {eoi_state
o  2462            16384,                                          {allocation_unit_size
o  2463            16384,                                          {transfer_unit_size
o  2464            7fffffff(16),                                   {file_limit  (!! is this right)
o  2465            gfc$qs_job_working_set,                         {queue_status
o  2466            pmc$initialize_to_zero,                         {preset_value
o  2467            0,                                              {time last modified
o  2468            0,                                              {last_segment_number
o  2469            [0, 0],                                         {global_task_id
o  2470            0,                                              {stack_for_ring
o  2471            gfc$fm_transient_segment];                      {media
o  2472
o  2473
o  2474 ?? FMT (FORMAT := ON) ??
o  2475
```


NOS/VE Global File Management : Assign and Free FDE Entries
BUILT-IN LIKE FUNCTIONS - min, max

```
o  2478
o  2479   FUNCTION [INLINE] max
o  2480     (     i: integer;
o  2481           j: integer): integer;
o  2482
o  2483     IF i > j THEN
o  2484       max := i;
o  2485     ELSE
o  2486       max := j;
o  2487     IFEND;
o  2488
o  2489   FUNCEND max;



o  2491
o  2492   FUNCTION [INLINE] min
o  2493     (     i: integer;
o  2494           j: integer): integer;
o  2495
o  2496     IF i < j THEN
o  2497       min := i;
o  2498     ELSE
o  2499       min := j;
o  2500     IFEND;
o  2501
o  2502   FUNCEND min;
```

```
o  2505 {
o  2506 { This routine is called to free pages assigned to file descriptors that have been freed.
o  2507 { Since file descriptors reside in wired/fixed memory, aging will never free the
o  2508 { unused pages; the only way pages get freed is to explicitly issue a MMP$FREE_PAGES
o  2509 { request to free them.
o  2510 {
o  2511
o  2512   PROCEDURE [INLINE] free_unused_pages
o  2513     (     control_p: ^gft$file_descriptor_control;
o  2514           free_word_index: 0 .. 1023);
o  2515
o  2516     VAR
o  2517       address_to_free: ^cell,
o  2518       b64: bool64,
o  2519       end_page: integer,
o  2520       first_fde_index_to_free: gft$file_descriptor_index,
o  2521       last_fde_index_to_free: gft$file_descriptor_index,
o  2522       low_bit_index: integer,
o  2523       low_word_index: integer,
o  2524       high_bit_index: integer,
o  2525       high_word_index: integer,
o  2526       max_words_to_search: integer,
o  2527       pages_to_free: integer,
o  2528       start_page: integer,
o  2529       status: ost$status,
o  2530       stop: integer,
o  2531       word: integer,
o  2532       words_p: ^array [0 .. gfc$max_level_2_index] of integer;
o  2533
o  2534
o  2535 { Calculate number of IN_USE words to search for free entries. The raximum number is
o  2536 { determined by the page size and FDE size. It is necessary to search multiple words because
o  2537 { more than 64 FDEs may fit in a word.
o  2538
o  2539     max_words_to_search := ((osv$page_size DIV gfc$fde_size) DIV 64) + 1;
o  2540
o  2541
o  2542 { Calculate the FDE index of the last FDE entry that is in use that has
o  2543 { an FDE.INDEX lower than the one just freed. Make sure not to run off the bottom
o  2544 { of the array. Terminate the search after checking a few words worth of bits;
o  2545 { exact number determined by <max_words_to_search>. Theres no since freeing
o  2546 { pages that have already been freed. NOTE: there's no tricky way to find the last "1"
o  2547 { bit in a word; keep shifting the word right until it is ODD.
o  2548
o  2549     words_p := #LOC (control_p^.in_use);
o  2550     low_word_index := free_word_index - 1;
o  2551     stop := max (0, free_word_index - max_words_to_search + 1);
o  2552     WHILE (low_word_index >= stop) AND (words_p^ [low_word_index] = 0) DO
o  2553       low_word_index := low_word_index - 1;
o  2554     WHILEND;
o  2555     low_bit_index := 64;
o  2556     IF low_word_index >= stop THEN
o  2557       word := words_p^ [low_word_index];
o  2558       WHILE #SHIFT (#SHIFT (word, -1), 1) = word DO
o  2559         word := #SHIFT (word, -1);
```



```
o  2560         low_bit_index := low_bit_index - 1;
o  2561       WHILEND;
o  2562     IFEND;
o  2563     first_fde_index_to_free := low_bit_index + low_word_index * 64;
o  2564
o  2565
o  2566 { Calculate the FDE index of the first FDE entry that is in use that has
o  2567 { and FDE.INDEX higher than the one just freed. Make sure not to run off the top
o  2568 { of the array. Terminate the search after checking a few words worth of bits;
o  2569 { exact number determined by <max_words_to_search>.
o  2570
o  2571     high_word_index := free_word_index + 1;
o  2572     stop := min (UPPERBOUND (words_p^), free_word_index + max_words_to_search - 1);
o  2573     WHILE (high_word_index <= stop) AND (words_p^ [high_word_index] = 0) DO
o  2574       high_word_index := high_word_index + 1;
o  2575     WHILEND;
o  2576     IF high_word_index > stop THEN
o  2577       high_bit_index := 0;
o  2578     ELSE
o  2579       word := -(words_p^ [high_word_index] + 1);
o  2580       #UNCHECKED_CONVERSION (word, b64);
o  2581       high_bit_index := find_zero_bit (b64);
o  2582     IFEND;
o  2583     last_fde_index_to_free := high_bit_index + high_word_index * 64 - 1;
o  2584
o  2585
o  2586 { Calculate addresses to be freed. Round starting and ending address to page boundaries.
o  2587 { Dont actually issue the monitor request to free pages unless there are really pages
o  2588 { to be freed.
o  2589
o  2590     start_page := gfc$fde_size * first_fde_index_to_free;
o  2591     start_page := (start_page + osv$page_size - 1) DIV osv$page_size;
o  2592
o  2593     end_page := gfc$fde_size * (last_fde_index_to_free + 1);
o  2594     end_page := end_page DIV osv$page_size;
o  2595
o  2596     pages_to_free := end_page - start_page;
o  2597
o  2598
o  2599     IF pages_to_free <> 0 THEN
o  2600       address_to_free := #ADDRESS (1, #SEGMENT (control_p), gfc$fde_table_base + start_page * osv$page_size);
o  2601 {!!!###  mmp$free_pages (address_to_free, pages_to_free * osv$page_size, osc$wait, status);
o  2602     IFEND;
o  2603
o  2604   PROCEND free_unused_pages;
```

find_zero_bit

```
o  2607 {
o  2608 { This tricky little routine returns the bit number of the first "zero" bit in a 64-bit word
o  2609 { (or in this case a packed array of 64 booleans). The algorithm uses trick CYBIL code to convert
o  2610 { the word to an integer, then convert the integer to a REAL. The exponent portion of
o  2611 { the REAL gives the bit number of the first "zero" bit.
o  2612
o  2613
o  2614   FUNCTION [INLINE] find_zero_bit
o  2615     (    s64: bool64): 0 .. 63;
o  2616
o  2617     VAR
o  2618       int: integer,
o  2619       r: real,
o  2620       trick: record
o  2621         case boolean of
o  2622         = FALSE =
o  2623           int: integer,
o  2624         = TRUE =
o  2625           fill: 0 .. 255,
o  2626           bit: 0 .. 255,
o  2627         casend,
o  2628       recend,
o  2629       zero_bit: integer;
o  2630
o  2631
o  2632 { If the integer is positive, then the first zero bit must be bit 0.
o  2633
o  2634     #UNCHECKED_CONVERSION (s64, int);
o  2635     IF int >= 0 THEN
o  2636       zero_bit := 0;
o  2637
o  2638 { Otherwise, convert the integer to REAL and get the bit number from the exponent. Note that the bits
o  2639 { in the integer are complemented ((-int-1) changes 1's to 0's and 0's to 1's) before converting to
o  2640 { real because the exponent actually give the first "one" bit.
o  2641
o  2642     ELSE
o  2643       r := $REAL (-int - 1);
o  2644       #UNCHECKED_CONVERSION (r, trick.int);
o  2645       zero_bit := 64 - trick.bit;
o  2646     IFEND;
o  2647
o  2648     find_zero_bit := zero_bit;
o  2649
o  2650   FUNCEND find_zero_bit;
```


[XDCL] gfp$assign_fde

```
o  2653
o  2654 {
o  2655 { This procedure is used to assign a new FDE entry. It searchs the FDE array in job
o  2656 { fixed or mainframe wired (depending on table residency) and returns an SFID & pointer
o  2657 { to the first available entry found. On return from this procedure the entry is NOT locked
o  2658 { for the task that assigned it.
o  2659 {
o  2660 { Most fields in the newly assigned FDE are initialized to a default value. See the module
o  2661 { GFM$FILE_TABLE_MANAGER for a definition of the values.
o  2662 {
o  2663 {
o  2664 {     GFP$ASSIGN_FDE (RESIDENCE, SEGMENT_NUMBER, SFID, FDE_P)
o  2665 {
o  2666 {
o  2667 { RESIDENCE: (INPUT) Specifies whether the FDE should be assigned in job fixed or
o  2668 {                mainframe wired.
o  2669 { SEGMENT_NUMBER: (INPUT) If residence is GFC$TR_NULL, then this parameter specifies
o  2670 {            an alternate segment number for the job fixed segment.
o  2671 { SFID:       (OUTPUT) The SFID of entry assigned is returned here. The SFID.HASH
o  2672 {             field in the SFID and FDE is initially set to ZERO by this procedure. The
o  2673 {             caller is responsible for changing these fields.
o  2674 { FDE_P:      (OUTPUT) This parameter contains a pointer to the FDE assigned. The FDE
o  2675 {             is NOT locked for task the created it.
o  2676
o  2677
o  2678   PROCEDURE [XDCL] gfp$assign_fde
4  2679     (     residence: gft$table_residence;
4  2680           segment_number: ost$segment;
4  2681      VAR sfid: gft$system_file_identifier;
4  2682      VAR fde_p: gft$file_desc_entry_p);
4  2683
4  2684     VAR
4  2685       control_p: ^gft$file_descriptor_control,
4  2686       file_entry_index: gft$file_descriptor_index,
4  2687       level1: 0 .. 63,
4  2688       level2: 0 .. 63,
4  2689       seg: ost$segment,
4  2690       trick_int: integer,
4  2691       zinuse: 0 .. 63;
4  2692
4  2693
4  2694 { Get a pointer to the control structures for the FDEs. This pointer may be either
4  2695 { a pointer to job fixed or to mainframe wired.
4  2696
4  2697     IF residence = gfc$tr_job THEN
E  2698       seg := osc$segnum_job_fixed_heap;
14 2699     ELSEIF residence = gfc$tr_system THEN
1A 2700       seg := osc$segnum_mainframe_wired;
22 2701     ELSE
22 2702       seg := segment_number;
28 2703     IFEND;
28 2704
28 2705     control_p := #ADDRESS (1, seg, gfc$fde_control_table_base);
28 2706
28 2707
28 2708 { Lock the tables to prevent other users from assigning FDEs.
```

[XDCL] gfp$assign_fde

```
      28  2709
      28  2710      osp$set_mainframe_sig_lock (control_p^.lock);
     126  2711
     126  2712
     126  2713 [ Scan the level 1 index to find the first level 2 table that has free entries.
     126  2714
     126  2715      level1 := find_zero_bit (control_p^.index1);
     150  2716
     150  2717
     150  2718 [ If the level 1 index is greater than 15, then tables are full. (Although the table structure will support
     150  2719 [ more entries, it would require an SFID.INDEX > 65K. This breaks compatibility).
     150  2720
     150  2721      IF level1 > 15 THEN
     158  2722        fde_p := NIL;
     168  2723      ELSE
     168  2724
     168  2725 [ Scan reset of the indices to find the index of the FDE to be assigned.
     168  2726
     168  2727        level2 := find_zero_bit (control_p^.index2 [level1]);
     196  2728        zinuse := find_zero_bit (control_p^.in_use [level2 + 64 * level1]);
     1CC  2729
     1CC  2730
     1CC  2731 [ Mark the entry as assigned. If the array entry containing the IN_USE bit for the entry just assigned
     1CC  2732 [ is full (all entries in the block assigned), mark the level 2 index as full. If the array entry
     1CC  2733 [ containing the level 2 bit is full, mark the level 1 table as full.
     1CC  2734
     1CC  2735        control_p^.in_use [level2 + 64 * level1] [zinuse] := TRUE;
     1CC  2736
     1CC  2737        #UNCHECKED_CONVERSION (control_p^.in_use [level2 + 64 * level1], trick_int);
     1CC  2738        IF trick_int = -1 THEN
     1EE  2739          control_p^.index2 [level1] [level2] := TRUE;
     1EE  2740          #UNCHECKED_CONVERSION (control_p^.index2 [level1], trick_int);
     1EE  2741          IF trick_int = -1 THEN
     208  2742            control_p^.index1 [level1] := TRUE;
     210  2743          IFEND;
     212  2744        IFEND;
     212  2745
     212  2746
     212  2747 [ Create the SFID  and FDE_P for the entry just assigned. Note that the hash field must be initialized by the
     212  2748 [ caller.
     212  2749
     212  2750        file_entry_index := ((level1 * 64) + level2) * 64 + zinuse;
     212  2751        sfid.file_entry_index := file_entry_index;
     212  2752        IF residence = gfc$tr_system THEN
     22E  2753          sfid.residence := gfc$tr_system;
     236  2754        ELSE
     236  2755          sfid.residence := gfc$tr_job;
     23A  2756        IFEND;
     23A  2757        fde_p := #ADDRESS (1, #SEGMENT (control_p), gfc$fde_table_base + gfc$fde_size * file_entry_index);
     23A  2758
     23A  2759
     23A  2760 [ Initialize the table entry with the default FDE value.
     23A  2761
     23A  2762        fde_p^ := initial_fde_entry;
     26C  2763        fde_p^.file_hash := (#free_running_clock (0) MOD 249) + 1;
     276  2764        sfid.file_hash := fde_p^.file_hash;
```


[XDCL] gfp$assign_fde

```
     292  2765      IFEND;
     292  2766
     292  2767      osp$clear_mainframe_sig_lock (control_p^.lock);
     3A8  2768
     3A8  2769    PROCEND gfp$assign_fde;
       0  2770
```

[XDCL] gfp$free_fde

```
        o  2773
        o  2774 {
        o  2775 { This procedure is used to free an FDE entry. Before calling this procedure,
        o  2776 { all memory assigned to the file should be freed. All tables subordinate to he FDE
        o  2777 { should be freed. The FDE cannot be accessed after being freed.
        o  2778 {
        o  2779 { Before calling this procedure, the FDE entry should be unlocked with gfp$unlock_fde if it is
        o  2780 { locked.
        o  2781 {
        o  2782 {     GFP$FREE_FDE (FDE_P)
        o  2783 {
        o  2784 { FDE_P: (INPUT) This parameter contains a pointer to the entry being freed.
        o  2785 {
        o  2786
        o  2787   PROCEDURE [XDCL] gfp$free_fde
        o  2788     (     fde_p: gft$file_desc_entry_p);
        o  2789
        o  2790     VAR
        o  2791       control_p: ^gft$file_descriptor_control,
        o  2792       gtid_int: integer,
        o  2793       i: gft$file_descriptor_index,
        o  2794       int: integer,
        o  2795       level1: 0 .. 63,
        o  2796       level2: 0 .. 63,
        o  2797       xcb_p: ^ost$execution_control_block,
        o  2798       zinuse: 0 .. 63;
        o  2799
        o  2800
        o  2801 { Verify that the FDE_P is valid.
        o  2802
        o  2803     IF (#SEGMENT (fde_p) <> 1) AND (#SEGMENT (fde_p) <> 3) THEN
       1A  2804       osp$system_error ('GF - Bad FDE_P on FREE', NIL);
       42  2805     IFEND;
       42  2806     int := (#OFFSET (fde_p) - gfc$fde_table_base) DIV gfc$fde_size;
       42  2807     IF (int < 0) OR (int > 65535) OR ((int * gfc$fde_size + gfc$fde_table_base) <> #OFFSET (fde_p)) THEN
       78  2808       osp$system_error ('GF - Bad FDE_P on FREE', NIL);
       9C  2809     IFEND;
       9C  2810     IF fde_p^.job_lock.locked THEN
       A8  2811       osp$system_error ('GF - freed locked FDE', NIL);
       CC  2812     IFEND;
       CC  2813     IF fde_p^.asti <> 0 THEN
       D4  2814       osp$system_error ('GF - freed FDE with asti <> 0', NIL);
       F8  2815     IFEND;
       F8  2816
       F8  2817
       F8  2818 { Calculate the indexes to the index levels.
       F8  2819
       F8  2820     i := (#OFFSET (fde_p) - gfc$fde_table_base) DIV gfc$fde_size;
       F8  2821     zinuse := i MOD 64;
       F8  2822     i := i DIV 64;
       F8  2823     level2 := i MOD 64;
       F8  2824     i := i DIV 64;
       F8  2825     level1 := i MOD 64;
       F8  2826
       F8  2827
       F8  2828 { Halt if we attempt to free an FDE with an open_count > 0.
```


[XDCL] gfp$free_fde

```
       F8  2829
       F8  2830     IF fde_p^.open_count > 0 THEN
      128  2831       osp$system_error ('GF - open_count > 0 during FREE_FDE', NIL);
      150  2832     IFEND;
      150  2833
      150  2834 { Get a pointer to the control structures for the FDEs. This pointer may be either
      150  2835 { a pointer to job fixed or to mainframe wired.
      150  2836
      150  2837     control_p := #ADDRESS (1, #SEGMENT (fde_p), gfc$fde_control_table_base);
      150  2838
      150  2839
      150  2840 { Lock the tables to prevent other users from assigning FDEs.
      150  2841
      150  2842     osp$set_mainframe_sig_lock (control_p^.lock);
      25C  2843
      25C  2844
      25C  2845 { Set each index level to indicate free entries. Its faster to mark each level to
      25C  2846 { show free entries than to actually check
      25C  2847
      25C  2848     control_p^.in_use [level2 + 64 * level1] [zinuse] := FALSE;
      25C  2849     control_p^.index2 [level1] [level2] := FALSE;
      25C  2850     control_p^.index1 [level1] := FALSE;
      25C  2851
      25C  2852
      25C  2853 { Change the file hash in the FDE being freed to cause errors if an attempt is made to
      25C  2854 { reference the entry again. NOTE that the job_lock is not cleared and will contain the GTID
      25C  2855 { of the task that freed the entry until the entry is reused.
      25C  2856
      25C  2857     fde_p^.file_hash := gfc$null_file_hash;
      25C  2858
      25C  2859
      25C  2860 { If the word containing the 'in_use' bit for the entry just freed is all zeros, attempt to
      25C  2861 { free unused pages.
      25C  2862
      25C  2863     #UNCHECKED_CONVERSION (control_p^.in_use [level2 + 64 * level1], int);
      25C  2864     IF int = 0 THEN
      29A  2865       free_unused_pages (control_p, level2 + 64 * level1);
      3E8  2866     IFEND;
      3E8  2867
      3E8  2868     osp$clear_mainframe_sig_lock (control_p^.lock);
      506  2869
      506  2870   PROCEND gfp$free_fde;
```

[XDCL] gfp$initialize

```
        o  2873 {
        o  2874 { This procedure should be called earily in deadstart. The primary function of this call is to
        o  2875 { verify that compile time constants are correct. CYBIL does not have the language
        o  2876 { constructs that would allow this type of checking to be done at compile time.
        o  2877 { If constants are incorrect, deadstart is aborted with a nice message.
        o  2878 {
        o  2879
        o  2880   PROCEDURE [XDCL] gfp$initialize;
        4  2881
        4  2882     IF #SIZE (gft$file_descriptor_entry) > gfc$fde_size THEN
        6  2883       osp$fatal_system_error ('GF - FDE size is incorrect', NIL);
       2E  2884     IFEND;
       2E  2885
       2E  2886   PROCEND gfp$initialize;
```


[XDCL] gfp$reassign_fde

```
        o  2889
        o  2890
        o  2891 {
        o  2892 { This procedure is used in job begin to recreate the cloned template FDEs. In the original cloning
        o  2893 { process, copies of the FDEs were made. During LOGIN of subsequent jobs, it is necessary to recreate
        o  2894 { the identical FDEs with the same hash and index.
        o  2895 {
        o  2896 { Most fields in the newly assigned FDE are set to the same value as in the original FDE.
        o  2897 { The MEDIA is reset to transient segment and EOI is set to zero.
        o  2898 {
        o  2899 {
        o  2900 {     GFP$REASSIGN_FDE (SFID, OLD_FDE_P)
        o  2901 {
        o  2902 {
        o  2903 { SFID:  (INPUT) This parameter specifies the SFID of the entry to be created.
        o  2904 {           A system error occurs if the entry is already in use.
        o  2905 { OLD_FDE_P: (INPUT) This parameter is a pointer to a copy of the FDE in the
        o  2906 {           cloned template.
        o  2907
        o  2908
        o  2909   PROCEDURE [XDCL] gfp$reassign_fde
        o  2910     (     sfid: gft$system_file_identifier;
        o  2911           old_fde_p: gft$file_desc_entry_p);
        o  2912
        o  2913     VAR
        o  2914       control_p: ^gft$file_descriptor_control,
        o  2915       fde_p: gft$file_desc_entry_p,
        o  2916       ignore_status: ost$status;
        o  2917
        o  2918
        o  2919 { Get a pointer to the control structures for the job FDEs.
        o  2920
        o  2921     control_p := #ADDRESS (1, osc$segnum_job_fixed_heap, gfc$fde_control_table_base);
        4  2922
        4  2923
        4  2924 { Validate the SFID. (Note: code doesn't currently set level1 or level2 indexes as INUSE so don't allow
        4  2925 { file_entry_index > 62).
        4  2926
        4  2927     IF (sfid.residence <> gfc$tr_job) OR (sfid.file_entry_index > 62) THEN
       32  2928       osp$system_error ('GF - invalid SFID on recreate', NIL);
       56  2929     IFEND;
       56  2930
       56  2931
       56  2932 { Mark the entry as 'INUSE' and restore the FDE data.
       56  2933
       56  2934       control_p^.in_use_bits [sfid.file_entry_index] := TRUE;
       56  2935       fde_p := #ADDRESS (1, #SEGMENT (control_p), gfc$fde_table_base + gfc$fde_size * sfid.file_entry_index);
       56  2936
       56  2937
       56  2938 { Initialize the table entry with the default FDE value.
       56  2939
       56  2940       fde_p^ := old_fde_p^;
       8A  2941       fde_p^.media := gfc$fm_transient_segment;
       8A  2942       fde_p^.eoi_byte_address := 0;
       8A  2943       fde_p^.asti := 0;
       8A  2944       IF fde_p^.file_kind = gfc$fk_job_local_file THEN
```

```
[XDCL] gfp$reassign_fde

      A2  2945          mmp$assign_mass_storage (0, sfid, 0, ignore_status);
      C6  2946        IFEND;
      C6  2947
      C6  2948    PROCEND gfp$reassign_fde;
       o  2949
       o  2951
       o  2952 MODEND gfm$file_table_manager

   **** I=$05578173AS0102D19890821T183254  L=ZZXXLIST  B=LGO  DA=NONE  LO=R  RC=NONE  OPT=SCHED  EL=F  LF=CS612  PAD=0

   **** NO DIAGNOSTICS
```


IDENTIFIER	DEFINED ON LINE	REFERENCES						
actual_value	712	722						
actual_value	2392	2404	2412					
actual_value	2678	2710	2710					
actual_value	2678	2767						
actual_value	2787	2842	2842					
actual_value	2787	2868						
address_to_free	2517	2600/M						
address_to_free	2787	2865/M						
amc$cell_pointer	598	602						
amc$file_byte_limit	185	188	190					
amc$heap_pointer	598	604						
amc$sequence_pointer	599	606						
amt$file_byte_address	188	149						
amt$file_limit	190	153						
amt$pointer_kind	598	589	601					
amt$segment_pointer	600	591						
asti	148	2813	2943/M					
b64	2518	2580	2581/P					
b64	2787	2865	2865/P					
bit	2512	2581						
bit	2626	2645						
bit	2678	2715	2727	2728				
bit	2787	2865						
bool64	74	62	63	66	2518	2615		
cnv	690	748/M	748					
cnv	2154	2166/M	2167					
cnv	2380	2411/M	2411					
cnv	2678	2710/M	2710	2767/M	2767			
cnv	2787	2842/M	2842	2868/M	2868			
code	690	748						
code	2149	2165						
code	2187	748/M	2165/M	2411/M	2710/M	2767/M	2842/M	2868/M
code	2380	2411						
code	2678	2710	2767					
code	2787	2842	2868					
control_p	2513	2549	2600					
control_p	2685	2705/M	2710/P	2715/P	2727/P	2728/P	2735/M	2737 2739/M
		2740	2742/M	2757	2767/P			
control_p	2787	2865	2865					
control_p	2791	2837/M	2842/P	2848/M	2849/M	2850/M	2863	2865/P 2868/P
control_p	2914	2921/M	2934/M	2935				
cs_status	714	723	724	725				
cs_status	2393	2404	2405	2406				
cs_status	2678	2710	2710	2710				
cs_status	2678	2767	2767	2767				
cs_status	2787	2842	2842	2842				
cs_status	2787	2868	2868	2868				
cycle_task	690	748/M	748/M	748/M	748/M	748/M	748/P	748/P
cycle_task	2162	2164/M	2165/M	2167/M	2168/M	2169/M	2170/P	2170/P
cycle_task	2380	2411/M	2411/M	2411/M	2411/M	2411/M	2411/P	2411/P
cycle_task	2678	2710/M	2710/M	2710/M	2710/M	2710/M	2710/P	2710/P 2767/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		2767/M	2767/M	2767/M	2767/M	2767/P	2767/P		
cycle_task	2787	2842/M	2842/M	2842/M	2842/M	2842/M	2842/P	2842/P	2868/M
		2868/M	2868/M	2868/M	2868/M	2868/P	2868/P		
end_page	2519	2593/M	2594/M	2594	2596				
end_page	2787	2865/M	2865/M	2865	2865				
eoi_byte_address	149	2942/M							
fde_p	2682	2722/M	2757/M	2762/M	2763/M	2764			
fde_p	2788	2803	2803	2806	2807	2810	2813	2820	2830
		2837	2857/M						
fde_p	2915	2935/M	2940/M	2941/M	2942/M	2943/M	2944		
file_entry_index	520	2751/M	2927	2934/S	2935				
file_entry_index	2686	2750/M	2751	2757					
file_hash	146	2763/M	2764	2857/M					
file_hash	522	2764/M							
file_kind	145	2944							
find_zero_bit	2614	2581	2650	2715	2727	2728	2865		
find_zero_bit	2615	2581/M	2648/M	2715/M	2727/M	2728/M	2865/M		
first_fde_index_to_free	2520	2563/M	2590						
first_fde_index_to_free	2787	2865/M	2865						
free_unused_pages	2512	2604	2865						
free_word_index	2514	2550	2551/P	2571	2572/P				
free_word_index	2787	2865	2865/P	2865	2865/P				
gfc$fde_control_table_base	52	2705	2837	2921					
gfc$fde_size	53	2539	2590	2593	2757	2806	2807	2820	2865
		2865	2865	2882	2935				
gfc$fde_table_base	51	52	2600	2757	2806	2807	2820	2865	2935
gfc$fk_catalog	225	237							
gfc$fk_job_local_file	227	236	2944						
gfc$fk_unnamed_file	228	2456							
gfc$fm_mass_storage_file	240	162							
gfc$fm_served_file	241	165							
gfc$fm_transient_segment	240	2471	2941						
gfc$max_level_1_index	77	63							
gfc$max_level_2_bit_index	79	68							
gfc$max_level_2_index	78	66	70	2532					
gfc$null_file_hash	526	2857							
gfc$qs_job_working_set	281	2465							
gfc$tr_job	530	2697	2755	2927					
gfc$tr_system	530	2699	2752	2753					
gfp$assign_fde	2678	2769							
gfp$free_fde	2787	2870							
gfp$initialize	2880	2886							
gfp$reassign_fde	2909	2948							
gft$allocation_unit_size	196	151							
gft$attach_count	201	142	143						
gft$fde_flags	171	139							
gft$file_desc_entry_p	515	2682	2788	2911	2915				
gft$file_descriptor_control	60	2513	2685	2791	2914				
gft$file_descriptor_entry	136	141	515	2446	2446	2882			
gft$file_descriptor_index	210	520	2520	2521	2686	2793			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
gft$file_kind	221	145	233						
gft$file_media	240	161							
gft$open_count	270	144	286						
gft$queue_status	281	154							
gft$segment_lock_info	285	147							
gft$signature_lock	246	137							
gft$system_file_identifier	519	535	817	1231	1433	2681	2910		
gft$table_residence	530	521	2679						
gft$transfer_unit_size	207	152							
global_task_id	778	718	2397	2710	2767	2842	2868		
high_bit_index	2524	2577/M	2581/M	2583					
high_bit_index	2787	2865/M	2865/M	2865					
high_word_index	2525	2571/M	2573	2573/S	2573	2573/S	2574/M	2574	2576
		2579/S	2583						
high_word_index	2787	2865/M	2865	2865/S	2865/M	2865	2865	2865/S	2865
		2865/S	2865						
i	690	748/M							
i	2157	2166/M							
i	2380	2411/M							
i	2480	2483	2484						
i	2493	2496	2497						
i	2512	2551	2551						
i	2512	2572	2572						
i	2678	2710/M	2767/M						
i	2787	2842/M	2868/M						
i	2787	2865	2865						
i	2787	2865	2865						
i	2793	2820/M	2821	2822/M	2822	2823	2824/M	2824	2825
i#call_monitor	2181	748	2170	2411	2710	2767	2842	2868	
ignore_status	2916	2945/P							
in_use	66	2549	2728/P	2735/M	2737	2848/M	2863	2865	
in_use_bits	68	2934/M							
index	257	719	2398	2710	2767	2842	2868		
index1	62	2715/P	2742/M	2850/M					
index2	63	2727/P	2739/M	2740	2849/M				
initial_fde_entry	2446	2762							
initial_value	713	719/M	722						
initial_value	2678	2767/M	2767						
initial_value	2787	2868/M	2868						
int	2512	2581	2581	2581					
int	2512	2581							
int	2618	2634	2635	2643					
int	2623	2644							
int	2678	2715	2715	2715	2727	2727	2727	2728	2728
		2728							
int	2678	2715	2727	2728					
int	2787	2865	2865	2865					
int	2787	2865							
int	2794	2806/M	2807	2807	2807	2863	2864		
iot$transfer_count	1521	1509							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
j	2481	2483	2486						
j	2494	2496	2499						
j	2512	2551	2551						
j	2512	2572	2572						
j	2787	2865	2865						
j	2787	2865	2865						
jmc$highest_prio_age_interval	1062	1053	1063						
jmc$highest_service_accumulator	1094	1095							
jmc$highest_service_factor_valu	1146	1139							
jmc$keyword_offset_maximum	1079	1054							
jmc$kjl_maximum_entries	1027	1020	1021						
jmc$kol_maximum_entries	1037	1022							
jmc$max_active_jobs	1018	1005	1013	1014					
jmc$max_ajl_ord	1019	1018							
jmc$max_dispatching_control	963	967							
jmc$max_dispatching_priority	864	824	827	828					
jmc$maximum_job_count	1034	1027							
jmc$maximum_output_count	1044	1037							
jmc$maximum_service_classes	1112	1115							
jmc$min_dispatching_control	962	966							
jmc$null_service_class	1105	1106							
jmc$priority_aging_interval_max	1053	1050							
jmc$priority_p1	878	825							
jmc$priority_p10	887	826							
jmc$priority_p14	891	826							
jmc$priority_p8	885	825							
jmc$reserved_ajls	1023	1018							
jmc$service_accumulator_maximum	1086	1083							
jmc$service_factor_value_max	1139	1136							
jmc$system_default_offset	1078	1079							
jmc$unlimited_offset	1075	1064	1096						
jmt$dispatching_control	933	916							
jmt$dispatching_control_index	966	933							
jmt$dispatching_controls	936	934							
jmt$dispatching_priority	824	789	791	938					
jmt$job_priority	994	925	926	927	928				
jmt$maximum_active_jobs	1005	910							
jmt$priority_aging_interval	1050	918							
jmt$scheduling_priority	924	917							
jmt$service_accumulator	1083	908	909						
jmt$service_class_index	1115	901	911						
jmt$service_class_name	1118	903	904						
jmt$service_factor_value	1136	912							
jmt$service_factors	1132	912							
jmt$task_time_slice	976	956	957						
jmt$time_slice_values	955	802	940						
job_lock	137	2810							
last_fde_index_to_free	2521	2583/M	2593						
last_fde_index_to_free	2787	2865/M	2865						
level1	2687	2715/M	2721	2727/S	2728/S	2735/S	2737/S	2739/S	2740/S
		2742/S	2750						
level1	2795	2825/M	2848/S	2849/S	2850/S	2863/S	2865/P		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
level2	2688	2727/M	2728/S	2735/S	2737/S	2739/S	2750		
level2	2796	2823/M	2848/S	2849/S	2863/S	2865/P			
lock	61	2710/P	2767/P	2842/P	2868/P				
lock	691	722	748/P						
lock	2381	2404	2411/P						
lock	2678	2710	2710/P						
lock	2678	2767	2767/P						
lock	2787	2842	2842/P						
lock	2787	2868	2868/P						
lock_id	90	722	748	2169	2404	2411	2710	2710	2767
		2767	2842	2842	2868	2868			
lock_loop	2400	2400	2415						
lock_loop	2678	2710	2710						
lock_loop	2787	2842	2842						
lock_value	2190	748/M	2169/M	2411/M	2710/M	2767/M	2842/M	2868/M	
locked	247	2810							
low_bit_index	2522	2555/M	2560/M	2560	2563				
low_bit_index	2787	2865/M	2865/M	2865	2865				
low_word_index	2523	2550/M	2552	2552/S	2552	2552/S	2553/M	2553	2556
		2557/S	2563						
low_word_index	2787	2865/M	2865	2865/S	2865/M	2865	2865	2865/S	2865
		2865/S	2865						
max	2479	2489	2551	2865					
max	2481	2484/M	2486/M	2551/M	2551/M	2865/M	2865/M		
max_words_to_search	2526	2539/M	2551/P	2572/P					
max_words_to_search	2787	2865/M	2865/P	2865/P					
media	161	2941/M							
min	2492	2502	2572	2865					
min	2494	2497/M	2499/M	2572/M	2572/M	2865/M	2865/M		
mmc$assign_active_null	1271	1272							
mmc$cell_pointer	1370	1375							
mmc$eoi_actual	312	2461							
mmc$heap_pointer	1371	1379							
mmc$kw_asid	1296	1332							
mmc$kw_clear_space	1294	1319							
mmc$kw_current_segment_length	1293	1313							
mmc$kw_error_exit_procedure	1295	1323							
mmc$kw_gl_key	1295	1317							
mmc$kw_hardware_attributes	1297	1326							
mmc$kw_inheritance	1297	1334							
mmc$kw_max_segment_length	1294	1315							
mmc$kw_preset_value	1296	1321							
mmc$kw_ps_transfer_size	1298	1342							
mmc$kw_ring_numbers	1292	1308							
mmc$kw_segment_access_control	1296	1330							
mmc$kw_segment_number	1293	1311							
mmc$kw_shadow_segment	1298	1336							
mmc$kw_software_attributes	1295	1328							
mmc$kw_wired_segment	1298	1339							
mmc$segment_fault_processor_id	1864	1918							
mmc$sequence_pointer	1370	1377							
mmc$ssk_none	1465	1437							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
mmc$ssk_segment_number	1466	1435						
mmp$assign_mass_storage	533	2945						
mmt$access_selections	614	590						
mmt$ast_index	303	148	1165					
mmt$attribute_keyword	1292	1307						
mmt$eoi_state	312	150						
mmt$hardware_attribute_set	1361	1327						
mmt$hardware_attributes	1349	1361						
mmt$lock_segment_status	1445	1236						
mmt$max_sdt	1175	1179						
mmt$max_sdtx	1260	1264						
mmt$sdtx_stream_data	1243	1239						
mmt$segment_access_condition	1891	1918						
mmt$segment_access_rights	1409	1235						
mmt$segment_access_state	1415	1230						
mmt$segment_descriptor	1162	1172	1176					
mmt$segment_descriptor_extended	1228	1257	1261					
mmt$segment_inheritance	1278	1232	1335					
mmt$segment_pointer_kind	1370	1374						
mmt$segment_reservation_state	1455	1233						
mmt$shadow_info	1430	1237						
mmt$shadow_reference_info	1478	815						
mmt$shadow_segment_kind	1465	1434						
mmt$software_attribute_set	1363	1234	1329					
mmt$software_attributes	1357	1363						
mmt$xcb_page_wait_info	1489	801						
mtt$monitor_interlock	319	138						
nat$received_message_descriptor	1505	1498	1507					
nat$received_message_list	1497	783						
new_value	2391	2398/M	2404	2412				
new_value	2678	2710/M	2710	2710				
new_value	2787	2842/M	2842	2842				
nlc$cc_connect_confirm	1537	1528						
nlc$cc_connect_request	1536	1526						
nlc$cc_expedited_data	1542	1528						
nlc$cc_max_pdu_kind	1544	1547						
nlc$channel_connection_pdu	1560	1512						
nlc$channelnet_pdu	1560	1514						
nlt$cc_pdu_kind	1547	1525						
nlt$cc_seq#_or_connect_time	1524	1513						
nlt$cc_sequence_number	1550	1529						
nlt$device_identifier	1557	1508						
nlt$pdu_type	1560	1511						
old_fde_p	2911	2940						
open_count	144	2830						
osc$call_instruction	1755	1763						
osc$cs_successful	122	725	2406	2710	2767	2842	2868	
osc$cs_variable_locked	124	724	2405	2710	2767	2842	2868	
osc$cyber_180_model_unknown	422	2451						
osc$data_read	1754	1763						
osc$free_running_clock_maximum	505	502						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$invalid_ring	340	380							
osc$max_fault_contents	1931	1925							
osc$max_name_size	1122	1126	1129						
osc$max_page_size	493	489							
osc$max_ring	339	380	381						
osc$max_segment_length	363	386	1240	1271					
osc$max_status_condition_code	558	554	570						
osc$max_string_size	574	577	580	585					
osc$max_tasks	265	262							
osc$maximum_offset	362	363	383	383	384				
osc$maximum_processor_id	1803	1799							
osc$maximum_segment	361	382							
osc$min_page_size	492	489							
osc$min_ring	338	381							
osc$pr_base_constant	2334	717	2396	2710	2767	2842	2868		
osc$segnum_job_fixed_heap	669	716	2395	2698	2710	2767	2842	2868	2921
osc$segnum_mainframe_wired	666	2700							
osc$task_time_slice_maximum	987	990							
osp$clear_mainframe_sig_lock	690	757	2767	2868					
osp$fatal_system_error	2378	2883							
osp$mfh_for_segment_manager	759	744	2767	2868					
osp$set_mainframe_sig_lock	2380	2423	2710	2842					
osp$system_error	2371	726	732	2413	2710	2767	2767	2804	2808
		2811	2814	2831	2842	2868	2868	2928	
ost$asid	639	635	1196	1333					
ost$binary_unique_name	401	140							
ost$byte_count	629	619							
ost$cp_time	1593	800							
ost$cp_time_value	1591	813	1594	1595					
ost$cs_lock	88	781							
ost$debug_code	1754	1742							
ost$debug_list	1750	1654							
ost$debug_list_entry	1741	1750							
ost$debug_mask	1760	1653							
ost$exchange_package	1603	768							
ost$execute_privilege	1209	1191	1204						
ost$execution_control_block	767	711	793	2390	2797				
ost$flags	1660	1610							
ost$frame_descriptor	1718	1733							
ost$free_running_clock	502	156	799	939					
ost$global_task_id	256	158	249	710	778	779	1815	2018	2389
ost$key_lock	369	1197	1318						
ost$key_lock_value	375	372	1677	1679					
ost$keypoint_class	1692	1623	1694						
ost$keypoint_mask	1694	1626							
ost$minimum_save_area	1728	1615	1703	1912					
ost$monitor_condition	1564	1571							
ost$monitor_conditions	1571	1616	1620	1708	1987	2001			
ost$monitor_fault	1908	1857							
ost$monitor_fault_contents	1925	1921							
ost$name	1129	902	1118	1950					
ost$p_register	1675	1604	1729	1979	1985				
ost$page_size	489	470	2429						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
ost$paging_statistics	1781	808						
ost$processor_id	1799	771	1793					
ost$processor_id_set	1793	770						
ost$processor_model_number	419	403						
ost$processor_serial_number	497	402						
ost$pva	391	1648	1666	1680	1909	2002		
ost$read_privilege	1212	1192	1205					
ost$register_number	1671	1645	1714	1722	1723	1724		
ost$ring	380	392	1194	1195	1229	1309	1310	1665
ost$ring1_termination_reason	1811	804						
ost$segment	382	157	393	534	1312	1436	1643	1744 2680
		2689						
ost$segment_access_control	1202	1331						
ost$segment_descriptor	1189	1163						
ost$segment_length	386	536	1314	1316	1338	1340	1343	
ost$segment_offset	383	394	636	1244	1745	1747		
ost$signature_lock	89	61	691	2150	2381			
ost$stack_frame_save_area	1702	1736	1945					
ost$status	542	537	592	621	1325	1980	2372	2379 2529
		2916						
ost$status_condition	566	2204						
ost$status_condition_code	570	545	566					
ost$string	583	546						
ost$string_size	577	584						
ost$system_flag	2086	2082						
ost$task_index	262	257	296	297				
ost$task_time_slice	990	976						
ost$top_of_stack_pointer	1663	1655						
ost$trap_enable	1697	1612	1976					
ost$user_condition	1574	1581						
ost$user_conditions	1581	1614	1618	1706	1735	1948	1988	
ost$valid_relative_pointer	389	163	166	797	798			
ost$valid_ring	381	1655						
ost$virtual_machine_identifier	1685	1606	1608	1730				
ost$wait	656	620						
ost$write_privilege	1215	1193	1206					
ost$x_register	1672	1645	1714					
osv$page_size	2429	2539	2591	2591	2594	2600	2865	2865 2865
		2865	2865					
p	690	748	748					
p	690	748						
p	2150	2168	2169					
p	2160	2167						
p	2380	2411	2411					
p	2380	2411						
p	2678	2710	2710	2767	2767			
p	2678	2710	2767					
p	2787	2842	2842	2868	2868			
p	2787	2842	2868					
p1	2188	748/M	2167/M	2411/M	2710/M	2767/M	2842/M	2868/M
p2	2189	748/M	2168/M	2411/M	2710/M	2767/M	2842/M	2868/M
pages_to_free	2527	2596/M	2599					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES						
pages_to_free	2787	2865/M	2865					
pmc$initialize_to_zero	509	2466						
pmc$kill_task_flag	2086	2102						
pmc$max_signal_contents	2069	2063						
pmc$max_task_id	1824	1821						
pmt$condition_identifier	1898	1892						
pmt$cpu_model_number	479	468	475					
pmt$cpu_serial_number	482	469	474					
pmt$initialization_value	509	155	1322					
pmt$signal	2025	2019						
pmt$signal_contents	2063	2027						
pmt$signal_id	2030	2026						
pmt$task_id	1821	795	1816					
r	2512	2581/M	2581					
r	2619	2643/M	2644					
r	2678	2715/M	2715	2727/M	2727	2728/M	2728	
r	2787	2865/M	2865					
reqcode	2186	748/M	2164/M	2411/M	2710/M	2767/M	2842/M	2868/M
residence	521	2753/M	2755/M	2927				
residence	2679	2697	2699	2752				
s64	2512	2581						
s64	2615	2634						
s64	2678	2715	2727	2728				
s64	2787	2865						
seg	2689	2698/M	2700/M	2702/M	2705			
segment_number	2680	2702						
seqno	258	719	720	2398	2398	2710	2710	2767 2767
		2842	2842	2868	2868			
sfid	2681	2751/M	2753/M	2755/M	2764/M			
sfid	2910	2927	2927	2934/S	2935	2945/P		
sft$file_space_limit_kind	1474	1238						
start_page	2528	2590/M	2591/M	2591	2596	2600		
start_page	2787	2865/M	2865/M	2865	2865	2865		
stlc_allocation	776	742	743/M	2767	2767/M	2868	2868/M	
stop	2530	2551/M	2552	2552	2556	2572/M	2573	2573 2576
stop	2787	2865/M	2865	2865	2865	2865/M	2865	2865 2865
syc$rc_cycle	2231	748	2164	2411	2710	2767	2842	2868
syc$ucr_condition	1936	1947						
syc$user_defined_condition	1937	1949						
syp$cycle_for_lock	2149	748	2171	2411	2710	2767	2842	2868
system_give_up_cpu	786	747	2767	2868				
system_table_lock_count	781	731	737/M	737	742	747	2402/M	2402 2410/M
		2410	2710/M	2710	2710/M	2710	2767	2767/M 2767
		2767	2767	2842/M	2842	2842/M	2842	2868 2868/M
		2868	2868	2868				
syt$monitor_flag	1843	1828						
syt$monitor_flags	1828	769						
syt$monitor_request_code	2214	2186						
task_id	710	718/M	719	719	719			
task_id	2389	2397/M	2398	2398	2398			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
task_id	2678	2710/M	2710	2710	2710				
task_id	2678	2767/M	2767	2767	2767				
task_id	2787	2842/M	2842	2842	2842				
task_id	2787	2868/M	2868	2868	2868				
tmc$broken_task_fault_id	1864	1914							
tmc$btc_invalid_a0	1962	1983							
tmc$btc_invalid_p	1962	1983							
tmc$btc_mcr_traps_disabled	1963	1984							
tmc$btc_mf_traps_disabled	1962	1982							
tmc$btc_mntr_fault_buffer_full	1961	1982							
tmc$btc_system_error	1964	1978							
tmc$btc_ucr_traps_disabled	1963	1984							
tmc$cyc_clear_sys_lock	2194	748/P	2767/P	2868/P					
tmc$cyc_set_sys_lock	2194	2411/P	2710/P	2842/P					
tmc$cycle_reason	2193	2149	2187						
tmc$dummy_fault	1865	1920							
tmc$flag_available_31	2099	2103							
tmc$maximum_monitor_faults	1869	1860							
tmc$maximum_signals	2079	2076							
tmc$maximum_system_task_id	2112	2115							
tmc$mcr_fault	1864	1916							
tmc$signal_available_63	2061	2072							
tmc$stid_null_task	2118	2115							
tmt$broken_task_condition	1961	1977							
tmt$broken_task_monitor_fault	1975	1915							
tmt$mcr_faults	2000	1917							
tmt$monitor_fault_buffer	1854	806							
tmt$monitor_fault_buffers	1860	1855	1856	1857					
tmt$monitor_fault_identifiers	1863	1913	1989						
tmt$rb_cycle	2185	2162							
tmt$signal	2017	2012							
tmt$signal_buffer	2009	807							
tmt$signal_buffers	2076	2010	2011	2012					
tmt$system_flags	2082	782							
tmt$system_task_id	2115	773							
tmt$task_queue_link	295	288							
trick	2512	2581	2581						
trick	2620	2644	2645						
trick	2678	2715	2715	2727	2727	2728	2728		
trick	2787	2865	2865						
trick_int	2690	2737	2738	2740	2741				
word	2531	2557/M	2558	2558	2558	2558	2559/M	2559	2579/M
		2580							
word	2787	2865/M	2865	2865	2865/M	2865	2865	2865	2865/M
		2865							
words_p	2532	2549/M	2552	2552	2557	2572/P	2573	2573	2579
words_p	2787	2865/M	2865	2865	2865	2865/P	2865	2865	2865
xcb_p	711	716/M	718	731	737/M	737	742	742	743/M
		747	747						
xcb_p	2390	2395/M	2397	2402/M	2402	2410/M	2410		
xcb_p	2678	2710/M	2710	2710/M	2710	2710/M	2710		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

IDENTIFIER	DEFINED ON LINE	REFERENCES							
xcb_p	2678	2767/M	2767	2767	2767/M	2767	2767	2767	2767/M
		2767	2767						
xcb_p	2787	2842/M	2842	2842/M	2842	2842/M	2842		
xcb_p	2787	2868/M	2868	2868	2868/M	2868	2868	2868	2868/M
		2868	2868						
zero_bit	2512	2581/M	2581/M	2581					
zero_bit	2629	2636/M	2645/M	2648					
zero_bit	2678	2715/M	2715/M	2715	2727/M	2727/M	2727	2728/M	2728/M
		2728							
zero_bit	2787	2865/M	2865/M	2865					
zinuse	2691	2728/M	2735/S	2750					
zinuse	2798	2821/M	2848/S						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode

```
              3 MODULE jmm$job_scheduler_monitor_mode;
              4
```


NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
              6 {Pointer to the AJL.}
              7   VAR
              8     jmv$ajl_p: [XREF] ^jmt$active_job_list;
        o  1144 {Define pointer to Initiated Job List (IJL).
        o  1145
        o  1146   VAR
        o  1147     jmv$ijl_p: [XREF]  jmt$ijl_p;
           1175
           1176 VAR
           1177   jmv$max_ajl_ordinal_in_use: [XREF] jmt$ajl_ordinal;
           1178
           1179
        o  1182
        o  1183 VAR
        o  1184   jmv$number_free_ajl_entries: [XREF] integer;
        o  1185 {Define boolean that specifies whether jobs that go into long wait should be
        o  1186 {swapped immediately.
        o  1187
        o  1188   VAR
        o  1189     jmv$swap_jobs_in_long_wait: [XREF] boolean;
        o  1190
        o  1191 {Define value of AJL ORDINAL used by the system job
        o  1192
        o  1193   VAR
        o  1194     jmv$system_ajl_ordinal: [XREF]jmt$ajl_ordinal;
        o  1195
           1198   VAR
           1199     jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
           1200
        o  1203  VAR
        o  1204     jsv$ijl_swap_queue_list: [XREF] jst$ijl_swap_queue_list;
        o  1205
           1223
           1224 { This variable is set to TRUE by monitor mode scheduler when it is notified of thrashing
           1225 { (low memory) and there is only one job active.  Setting the variable will cause the
           1226 { job's working set to be reduced.  This is done instead of swapping out the job.
           1227
           1228   VAR
           1229     mmv$reduce_jws_for_thrashing: [XREF] boolean;
           1230
           1231
           1232   VAR
           1233     tmv$cpu_execution_statistics: [XREF] tmt$cpu_execution_statistics;
           1234
        o  1242
        o  1243   VAR
        o  1244     tmv$dispatch_priority_integer: [XREF] ARRAY [jmt$dispatching_priority] of integer;
        o  1245
        o  1246   VAR
        o  1247     tmv$dispatching_controls: [XREF] tmt$dispatching_controls;
        o  1248
           1279
           1280   VAR
           1281     tmv$dispatching_control_sets: [XREF] tmt$dispatching_control_sets;
           1282
```

NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
o  1300
o  1301    VAR
o  1302      tmv$dispatching_control_time: [XREF] tmt$dispatching_prio_controls;
o  1303
   1306  VAR
   1307      tmv$ptl_lock: [XREF] tmt$ptl_lock;
o  1322
o  1323    PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
o  1324      VAR ijle_p: ^jmt$initiated_job_list_entry);
o  1325
   1333
   1334    PROCEDURE [XREF] jmp$calculate_service
   1335      (     ijle_p: ^jmt$initiated_job_list_entry;
   1336       VAR service_used: integer);
   1337
o  1340
o  1341  { PURPOSE:
o  1342  {   This is the monitor mode procedure to change the entry status of a job.  The caller
o  1343  {   of procedure must set the PTL lock if the entry status change is a SWAPPED/NOT SWAPPED
o  1344  {   transition because the swapped job counts will be changed.
o  1345
o  1346    PROCEDURE [INLINE] jmp$change_ijl_entry_status
o  1347      (     ijle_p: ^jmt$initiated_job_list_entry;
o  1348            new_entry_status: jmt$ijl_entry_status);
o  1349
o  1350      VAR
o  1351        old_entry_status: jmt$ijl_entry_status;
o  1352
o  1353      old_entry_status := ijle_p^.entry_status;
o  1354
o  1355      jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] :=
o  1356            jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] + 1;
o  1357
o  1358      ijle_p^.entry_status := new_entry_status;
o  1359
o  1360      IF (old_entry_status <= jmc$ies_swapin_in_progress) AND
o  1361            (new_entry_status > jmc$ies_swapin_in_progress) THEN
o  1362        jmp$increment_swapped_job_count (ijle_p);
o  1363
o  1364      ELSEIF (old_entry_status > jmc$ies_swapin_in_progress) AND
o  1365            (new_entry_status <= jmc$ies_swapin_in_progress) THEN
o  1366        jmp$decrement_swapped_job_count (ijle_p);
o  1367      IFEND;
o  1368
o  1369    PROCEND jmp$change_ijl_entry_status;
o  1370    PROCEDURE [XREF] jmp$find_jsn (jsn: string (* <= jmc$system_supplied_name_size);
o  1371      VAR ijle_p: ^jmt$initiated_job_list_entry;
o  1372      VAR ijlo: jmt$ijl_ordinal);
o  1373
   1376    PROCEDURE [XREF] jsp$monitor_advance_swap (ijl_ordinal: jmt$ijl_ordinal);
   1377
o  1380   PROCEDURE [XREF] jsp$monitor_swap_in (ijl_ordinal: jmt$ijl_ordinal);
o  1381
   1384   PROCEDURE [XREF] jsp$monitor_swap_out (ijl_ordinal: jmt$ijl_ordinal);
   1385
```


NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
o  1388
o  1389    PROCEDURE [XREF] jsp$relink_swap_queue (ijl_ordinal: jmt$ijl_ordinal;
o  1390          ijle_p: ^jmt$initiated_job_list_entry;
o  1391          new_queue: jst$ijl_swap_queue_id);
o  1392
   1395
   1396    PROCEDURE [INLINE] mmp$nudge_periodic_call;
   1397
   1398      mmv$time_to_call_mem_mgr := 0;
   1399      osv$time_to_check_asyn := 0;
   1400
   1401    PROCEND mmp$nudge_periodic_call;
   1402
o  1412
o  1413  {  PURPOSE:     procedure mtp$error_stop
o  1414  {    Prefixes 'ERR=VEOS1000-' to the string and calls mtp$step_unstep_system to write string and step system}
o  1415
o  1416    PROCEDURE [XREF] mtp$error_stop (text: string(*<=63) );
o  1417
o  1418    PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
o  1419          condition: osc$max_status_condition_number + 1 .. 0fffffffff(16);
o  1420      VAR status: syt$monitor_status);
o  1421
   1431   PROCEDURE [INLINE] osp$fetch_locked_variable (VAR variable: integer;
   1432      VAR value: integer);
   1433
o  1447   PROCEDURE [INLINE] osp$set_locked_variable (VAR variable: integer;
o  1448          initial: integer;
o  1449          final: integer;
o  1450      VAR actual: integer;
o  1451      VAR succeeded: boolean);
o  1452  {
o  1453  {   The purpose of this procedure is to set a compare_swap lock
o  1454  {  when the user knows the initial contents of the lock.
o  1455  {  This procedure has been generated to help users avoid problems
o  1456  {  with #compare_swap.
o  1457  {  CAUTION: Variables referenced by this procedure may not be
o  1458  {  referenced (read or written) any way other than by the
o  1459  {  following procedures:
o  1460  {                              osp$increment_locked_variable
o  1461  {                              osp$decrement_locked_variable
o  1462  {                              osp$fetch_locked_variable
o  1463  {        and the intrinsic     #compare_swap.
o  1464  {
o  1465  {     OSP$SET_LOCKED_VARIABLE (VARIABLE, INITIAL, FINAL, ACTUAL, SUCCEEDED)
o  1466  {
o  1467  {  VARIABLE: (input,output) This parameter is the variable on which the
o  1468  {                     compare_swap operation is to be performed.
o  1469  {  INITIAL: (input) This parameter is the value that the variable must contain
o  1470  {                     initial content of the lock must be for the swap
o  1471  {                     operation to be successful.
o  1472  {  FINAL: (input) This parameter is the variable that specifies the value to be
o  1473  {                     stored in the lock if the swap is successful.
o  1474  {  ACTUAL: (output) This parameter is the variable into which the initial
o  1475  {                     contents of the lock is returned.
```

NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
o  1476 {  SUCCEEDED: (output) This parameter specifies whether the swap was successful
o  1477 {                      or not.
o  1478 {
o  1479
   1495
   1496   PROCEDURE [XREF] tmp$calculate_dct_priority_int;
   1497
   1498
   1499 PROCEDURE [INLINE] tmp$clear_lock (VAR lock: tmt$ptl_lock);
   1500
   1501     IF osv$cpus_logically_on > 1 THEN
   1502       IF lock.id <> #READ_REGISTER (osc$pr_base_constant) THEN
   1503         i#program_error; {interlock failure - no message passed for performance reasons}
   1504       IFEND;
   1505       IF lock.count > 0 THEN
   1506         lock.count := lock.count - 1;
   1507       ELSE
   1508         lock.clear := 0;
   1509       IFEND;
   1510     IFEND;
   1511
   1512 PROCEND   tmp$clear_lock;
   1513
o  1583
o  1584   PROCEDURE [XREF] tmp$free_unrecovered_tasks
o  1585     (     ijle_p: ^jmt$initiated_job_list_entry);
o  1586
   1589
   1590   PROCEDURE [XREF] tmp$monitor_ready_system_task (stid: tmt$system_task_id;
   1591     VAR status: syt$monitor_status);
   1592
o  1691
o  1692   PROCEDURE [XREF] tmp$reset_dispatching_control
o  1693     (     ijle_p: ^jmt$initiated_job_list_entry;
o  1694           ijlo: jmt$ijl_ordinal;
o  1695           excess_service_used: integer;
o  1696           expired_dispatching_control: boolean);
o  1697
   1700
   1701   PROCEDURE [INLINE] tmp$set_lock (VAR lock: tmt$ptl_lock);
   1702
   1703     VAR
   1704       b: boolean,
   1705       bc: integer;
   1706
   1707     IF osv$cpus_logically_on > 1 THEN
   1708       bc := #read_register (osc$pr_base_constant);
   1709       IF lock.id <> bc  THEN
   1710         REPEAT
   1711           #TEST_SET (lock.locked, b);
   1712         UNTIL NOT b;
   1713         lock.id := bc;
   1714       ELSE
   1715         lock.count := lock.count + 1;
   1716       IFEND;
```


NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
   1717     IFEND;
   1718
   1719 PROCEND tmp$set_lock;
   1720
o  1723
o  1724   PROCEDURE [XREF] tmp$update_job_task_environment (ijle_p: ^jmt$initiated_job_list_entry;
o  1725         ijl_ordinal: jmt$ijl_ordinal;
o  1726         xcb_search: tmt$fnx_search_type);
o  1727
   1743
o  3192
o  3193   VAR
o  3194
o  3195     jmv$classes_in_maxaj_limit_wait: [XDCL, #GATE] jmt$service_class_set := $jmt$service_class_set [],
o  3196
o  3197     jmv$classes_in_resource_wait: [XDCL, #GATE] jmt$service_class_set := $jmt$service_class_set [],
o  3198
o  3199     jmv$change_dispatching_list: [XDCL, #GATE, oss$mainframe_wired] jmt$change_dispatching_list := [[0], NIL],
o  3200
o  3201     jmv$idle_dispatching_controls: [XDCL, #GATE, oss$mainframe_wired] jmt$idle_dispatching_controls,
o  3202
o  3203     jmv$ijl_entry_status_statistics: [XDCL, #GATE, oss$mainframe_wired] jmt$ijl_entry_status_statistics,
o  3204
o  3205 { NOTE:  Because jmv$ijl_ready_task_list is read/written by both job mode and monitor mode scheduler,
o  3206 { it is a locked variable and can be referenced only via the compare_swap procedures.
o  3207
o  3208     jmv$ijl_ready_task_list: [XDCL, #GATE, oss$mainframe_wired] integer,
o  3209
o  3210     jmv$job_counts: [XDCL, #GATE] jmt$job_counts,
o  3211
o  3212     jmv$job_scheduler_event: [XDCL, #GATE, oss$mainframe_wired] jmt$job_scheduler_event := [REP 19 of FALSE],
o  3213
o  3214     jmv$job_sched_events_selected: [XDCL, #GATE, oss$mainframe_wired] jmt$job_sched_event_selections :=
o  3215           [TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
o  3216           FALSE, FALSE, FALSE],
o  3217
o  3218     jmv$job_scheduler_table: [XDCL, #GATE, oss$mainframe_wired] jmt$job_scheduler_table :=
o  3219           [40000, FALSE, 60, jmc$sched_profile_deadstart_id, 10, [REP 8 of [0, 100, FALSE]], 1, [[1, 8],
o  3220           [1, 8], [2, 8], [2, 8], [3, 8], [3, 8], [4, 8], [4, 8], [5, 8], [5, 8]], 360000000, [], [],
o  3221           [20, 60], NIL, 0],
o  3222
o  3223     jmv$last_service_calc_time: [XDCL, #GATE, oss$mainframe_wired] ost$free_running_clock := 0,
o  3224
o  3225     jmv$long_wait_swap_threshold: [XDCL, #GATE] integer,
o  3226
o  3227     jmv$max_class_working_set: [XDCL, #GATE] jmt$working_set_size := 3000,
o  3228
o  3229     jmv$max_service_class_in_use: [XDCL, #GATE] jmt$service_class_index,
o  3230
o  3231     jmv$min_think_time: [XDCL, #GATE] integer := 500000, {Dont update THINK TIME if estimated think
o  3232
o  3233 {time is less than this value.
o  3234
o  3235     jmv$max_think_time: [XDCL, #GATE] integer := 60000000, {THINK TIMEs > this value are rounded to this
o  3236
```

NOS/VE Job Management : job scheduler monitor mode
Global Declarations Referenced by This Module

```
o  3237 (value.
o  3238
o  3239     jmv$memory_needed_by_scheduler: [XDCL, #GATE] mmt$page_frame_index,
o  3240
o  3241     jmv$null_ijl_ordinal: [XDCL, #GATE] jmt$ijl_ordinal := [0, 0],
o  3242
o  3243     jmv$prevent_activation_of_jobs: [XDCL, #GATE] boolean := TRUE,
o  3244
o  3245     jmv$scan_idle_dispatch_interval: [XDCL, #GATE] integer := 15000000,
o  3246
o  3247     jmv$sched_profile_is_loading: [XDCL, #GATE, oss$mainframe_wired] boolean := FALSE,
o  3248
o  3249     jmv$sched_service_calc_time: [XDCL, #GATE] ost$free_running_clock,
o  3250
o  3251     jmv$service_class_stats_lock: [XDCL] tmt$ptl_lock := [FALSE, 0],
o  3252
o  3253     jmv$service_classes: [XDCL, #GATE, oss$mainframe_wired] array [jmt$service_class_index] of
o  3254           ^jmt$service_class_entry := [REP jmc$maximum_service_classes + 1 of NIL],
o  3255
o  3256     jmv$ssn_previous_sequence: [XDCL, #GATE] jmt$ssn_sequence_number,
o  3257
o  3258     jmv$subsystem_priority_changes: [XDCL, #GATE] packed array [jmt$service_class_index] of boolean,
o  3259
o  3260     jmv$swapin_candidate_queue: [XDCL, #GATE] array [jmt$service_class_index] of
o  3261           jmt$swapin_candidate_q_header,
o  3262
o  3263     jmv$swapped_idle_disp_count: [XDCL] integer := 0,
o  3264
o  3265     jmv$system_supplied_name: [XDCL, #GATE] jmt$system_supplied_name_mask,
o  3266
o  3267     jmv$time_to_wake_scheduler: [XDCL, #GATE] ost$free_running_clock;
o  3268
```


NOS/VE Job Management : job scheduler monitor mode
check_for_class_switch

```
o  3270
o  3271   PROCEDURE check_for_class_switch
4  3272     (     ijle_p: ^jmt$initiated_job_list_entry);
4  3273
4  3274     VAR
4  3275       new_class: jmt$service_class_index,
4  3276       rb: jmt$rb_scheduler_requests,
4  3277       service_class_p: ^jmt$service_class_attributes;
4  3278
4  3279 { Change the job's service class if the job has reached the class service threshold.
4  3280 { Only switch classes if the new class to switch to is currently defined.
4  3281
4  3282     service_class_p := ^jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes;
4  3283     IF (ijle_p^.job_scheduler_data.service_accumulator > service_class_p^.class_service_threshold) AND
38 3284           (service_class_p^.class_service_threshold <> jmc$unlimited_service_accum) AND
38 3285           (NOT jmv$sched_profile_is_loading) THEN
38 3286       IF ijle_p^.job_scheduler_data.service_class <> service_class_p^.next_service_class_index THEN
40 3287         new_class := service_class_p^.next_service_class_index;
40 3288         IF (jmv$service_classes [new_class] <> NIL) AND jmv$service_classes [new_class]^.attributes.
5E 3289               defined THEN
5E 3290           rb.reqcode := syc$rc_job_scheduler_request;
5E 3291           rb.sub_reqcode := jmc$src_class_switch;
5E 3292           rb.system_supplied_name := ijle_p^.system_supplied_name;
76 3293           rb.new_service_class := new_class;
76 3294           rb.new_service_accumulator := 0;
76 3295           rb.old_service_class := ijle_p^.job_scheduler_data.service_class;
76 3296           rb.old_service_accumulator := ijle_p^.job_scheduler_data.service_accumulator;
76 3297
76 3298           jmp$process_class_switch (rb);
98 3299
98 3300         IFEND;
98 3301       IFEND;
98 3302     IFEND;
98 3303   PROCEND check_for_class_switch;
 0 3304
```

NOS/VE Job Management : job scheduler monitor mode
insert_job_in_ready_task_list

```
 o  3306
 o  3307 { PURPOSE:
 o  3308 {   This procedure inserts a job with a ready task into the list for job mode
 o  3309 {   scheduler to process.
 o  3310 { DESIGN:
 o  3311 {   The head of the list is a global variable which must be changed by both
 o  3312 {   monitor mode and job mode scheduler.  To synchronize the monitor/job mode
 o  3313 {   references, the head of the list is a locked variable which can be referenced
 o  3314 {   only via the #compare_swap procedures.
 o  3315
 o  3316   PROCEDURE insert_job_in_ready_task_list
 o  3317     (     ijl_ordinal: jmt$ijl_ordinal;
 o  3318           ijle_p: ^jmt$initiated_job_list_entry);
 o  3319
 o  3320     VAR
 o  3321       ijlo: jmt$trick_ijlo_variant_record,
 o  3322       list_head: jmt$trick_ijlo_variant_record,
 o  3323       old_list_head: integer,
 o  3324       succeeded: boolean;
 o  3325
 o  3326     ijlo.ijl_ordinal := ijl_ordinal;
 4  3327
 4  3328     REPEAT
14  3329       osp$fetch_locked_variable (jmv$ijl_ready_task_list, list_head.ijl_integer);
44  3330       ijle_p^.job_scheduler_data.ready_task_link := list_head.ijl_ordinal;
44  3331       old_list_head := list_head.ijl_integer;
44  3332       osp$set_locked_variable (jmv$ijl_ready_task_list, old_list_head, ijlo.ijl_integer,
44  3333             list_head.ijl_integer, succeeded);
88  3334     UNTIL succeeded;
90  3335
90  3336   PROCEND insert_job_in_ready_task_list;
 o  3337
```


NOS/VE Job Management : job scheduler monitor mode
remove_class_from_maxaj_limit

```
 o  3339
 o  3340   PROCEDURE remove_class_from_maxaj_limit
 o  3341     (     service_class: jmt$service_class_index);
 o  3342
 o  3343     VAR
 o  3344       ignore_status: syt$monitor_status;
 o  3345
 o  3346     jmv$classes_in_maxaj_limit_wait := jmv$classes_in_maxaj_limit_wait - $jmt$service_class_set
 4  3347           [service_class];
 4  3348     jmv$job_scheduler_event [jmc$scheduler_wake_time] := TRUE;
 4  3349     jmv$job_scheduler_event [jmc$examine_input_queue] := TRUE;
 4  3350     jmv$job_scheduler_event [jmc$examine_swapin_queue] := TRUE;
 4  3351     tmp$monitor_ready_system_task (tmc$stid_job_scheduler, ignore_status);
40  3352
40  3353   PROCEND remove_class_from_maxaj_limit;
 o  3354
```

NOS/VE Job Management : job scheduler monitor mode
[INLINE, UNSAFE] swapin_queue_empty

```
o  3357
o  3358 { PURPOSE:
o  3359 {   This function is called when determining if a monitor swapin should be allowed to take place.
o  3360 {   If the input dispatching priority is blocked or if there are queued jobs with a higher
o  3361 {   dispatching priority FALSE will be returned.
o  3362
o  3363   FUNCTION [INLINE, UNSAFE] swapin_queue_empty
o  3364     (    dispatching_priority: jmt$dispatching_priority): boolean;
o  3365
o  3366     VAR
o  3367       ijle_p: ^jmt$initiated_job_list_entry,
o  3368       service_class: jmt$service_class_index;
o  3369
o  3370     swapin_queue_empty := TRUE;
o  3371
o  3372     IF jmv$idle_dispatching_controls.controls [dispatching_priority].blocked THEN
o  3373       swapin_queue_empty := FALSE;
o  3374       RETURN;
o  3375     IFEND;
o  3376
o  3377   /check_swapin_queue/
o  3378     FOR service_class := jmc$system_service_class TO jmv$max_service_class_in_use DO
o  3379       IF (jmv$swapin_candidate_queue [service_class].swapin_candidate_queue <> jmv$null_ijl_ordinal) AND
o  3380            (NOT too_many_active_jobs_for_class (service_class)) THEN
o  3381         jmp$get_ijle_p (jmv$swapin_candidate_queue [service_class].swapin_candidate_queue, ijle_p);
o  3382         IF ijle_p^.scheduling_dispatching_priority > dispatching_priority THEN
o  3383           swapin_queue_empty := FALSE;
o  3384           EXIT /check_swapin_queue/;
o  3385         IFEND;
o  3386       IFEND;
o  3387     FOREND /check_swapin_queue/;
o  3388
o  3389   FUNCEND swapin_queue_empty;
o  3390
```


NOS/VE Job Management : job scheduler monitor mode
[INLINE] too_many_active_jobs_for_class

```
o  3392
o  3393   FUNCTION [INLINE] too_many_active_jobs_for_class
o  3394     (    service_class: jmt$service_class_index): boolean;
o  3395
o  3396     too_many_active_jobs_for_class := (jmv$job_counts.service_class_counts [service_class].
o  3397           scheduler_initiated_jobs - jmv$job_counts.service_class_counts [service_class].swapped_jobs) >=
o  3398           jmv$service_classes [service_class]^.attributes.maximum_active_jobs;
o  3399
o  3400   FUNCEND too_many_active_jobs_for_class;
o  3401
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$activate_job_mode_swapper

```
 0  3403
 0  3404   PROCEDURE [XDCL] jmp$activate_job_mode_swapper;
 0  3405
 0  3406     VAR
 0  3407       status: syt$monitor_status;
 0  3408
 0  3409     jmp$set_scheduler_event (jmc$call_job_swapper);
38  3410
38  3411   PROCEND jmp$activate_job_mode_swapper;
 0  3412
```


NOS/VE Job Management : job scheduler monitor mode
jmp$change_dispatching_alloc

```
 0  3414
 0  3415 { PURPOSE:
 0  3416 {   This procedure changes the dispatching allocation controls in dispatcher's
 0  3417 {   tables.
 0  3418 { DESIGN:
 0  3419 {   The scheduler table has been changed in job mode.  Dispatcher's tables must
 0  3420 {   be changed in mtr mode with the PTL lock set so that task switch cannot be
 0  3421 {   referencing the tables.  The scheduler table is kept in units of seconds for
 0  3422 {   the time interval and percentages for the minimum and maximum values; those
 0  3423 {   values must all be converted to microseconds for the dispatching table.
 0  3424 {   This procedure is called infrequently (only when a site is changing its
 0  3425 {   dispatching allocation controls).
 0  3426
 0  3427   PROCEDURE jmp$change_dispatching_alloc;
 0  3428
 0  3429     CONST
 0  3430       u_second = 1000000;
 0  3431
 0  3432     VAR
 0  3433       controls_defined: boolean,
 0  3434       dp: jmt$user_dispatching_priority,
 0  3435       dp_unblocked: boolean,
 0  3436       local_set: tmt$dispatching_control_sets,
 0  3437       normalized_interval: integer;
 0  3438
 0  3439 { Decide if controls are being defined; the site may be setting controls back to defaults
 0  3440 { (0% minimum and 100% maximum).
 0  3441
 0  3442     controls_defined := FALSE;
 4  3443     dp_unblocked := FALSE;
 4  3444
 4  3445   /check_controls/
 4  3446     FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
18  3447       IF (jmv$job_scheduler_table.cpu_dispatching_allocation [dp].minimum <> 0) OR
32  3448             (jmv$job_scheduler_table.cpu_dispatching_allocation [dp].maximum <> 100) THEN
32  3449         controls_defined := TRUE;
32  3450         EXIT /check_controls/;
3A  3451       IFEND;
3A  3452     FOREND /check_controls/;
3E  3453
3E  3454 { Set the PTL lock while changing the dispatching tables.
3E  3455
3E  3456     tmp$set_lock (tmv$ptl_lock);
78  3457
78  3458 { Reset the dispatching control sets.  Reset the minimums_to_satisfy field in the dispatching table
78  3459 { (it is used in mmp$periodic to determine if a dispatching priority is blocked).
78  3460 { NOTE:  'System' priorities always have minimums_to_satisfy.  This guarantees that they will always be
78  3461 { in the first set considered by task selection in task switch.
78  3462 { If controls are not defined, clear the controls_defined field in the dispatching table.  Other fields
78  3463 { in the dispatching table can be left with 'garbage' in them; nothing references them when
78  3464 { controls_defined is FALSE.
78  3465 { If controls are defined, reset the values in the dispatching table.
78  3466 { NOTE:  Elements in dispatching priority sets are converted so that the highest priority in the set
78  3467 { is the leftmost bit in the set.  Setting bit 1 in a dispatching priority set is adding priority 15
78  3468 { to the set.  (See jmt$dispatching_priority.)
```

NOS/VE Job Management : job scheduler monitor mode
jmp$change_dispatching_alloc

```
78  3469
78  3470      tmv$dispatching_control_sets.minimums_to_satisfy := $jmt$dispatching_priority_set [1, 2, 3, 4, 5, 6];
78  3471      tmv$dispatching_control_sets.maximums_exceeded := $jmt$dispatching_priority_set [];
78  3472      tmv$dispatching_control_sets.enforce_maximums := $jmt$dispatching_priority_set [];
78  3473      tmv$dispatching_controls.minimums_to_satisfy := $jmt$dispatching_priority_set [1, 2, 3, 4, 5, 6];
78  3474
78  3475      IF NOT controls_defined THEN
98  3476        tmv$dispatching_controls.controls_defined := FALSE;
A0  3477      ELSE
A0  3478        tmv$dispatching_controls.controls_defined := TRUE;
A0  3479        tmv$dispatching_controls.maximums_defined := $jmt$dispatching_priority_set [];
A0  3480        tmv$dispatching_controls.enforce_maximums := $jmt$dispatching_priority_set [];
A0  3481        tmv$dispatching_controls.controls.time_left_in_interval :=
A0  3482              jmv$job_scheduler_table.dispatching_allocation_interval * u_second;
A0  3483        normalized_interval := tmv$dispatching_controls.controls.time_left_in_interval DIV 100;
A0  3484        FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
D0  3485          IF jmv$job_scheduler_table.cpu_dispatching_allocation [dp].minimum <> 0 THEN
DE  3486            tmv$dispatching_controls.controls.dispatching_priority_time [dp].
DE  3487                  minimum_time := (normalized_interval) * jmv$job_scheduler_table.
DE  3488                  cpu_dispatching_allocation [dp].minimum;
DE  3489            tmv$dispatching_controls.minimums_to_satisfy := tmv$dispatching_controls.minimums_to_satisfy +
DE  3490                  $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
DE  3491            IF jmv$idle_dispatching_controls.controls [dp].blocked THEN
11A 3492              jmv$idle_dispatching_controls.controls [dp].blocked := FALSE;
11A 3493              jmv$idle_dispatching_controls.controls [dp].timestamp := #FREE_RUNNING_CLOCK (0);
124 3494              jmv$idle_dispatching_controls.controls [dp].last_cp_time :=
124 3495                    tmv$cpu_execution_statistics [dp].time_spent_in_job_mode +
124 3496                    tmv$cpu_execution_statistics [dp].time_spent_in_mtr_mode;
124 3497              jmv$idle_dispatching_controls.unblocked_priorities :=
124 3498                    jmv$idle_dispatching_controls.unblocked_priorities +
124 3499                    $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
124 3500              dp_unblocked := TRUE;
158 3501            IFEND;
15C 3502          ELSE
15C 3503            tmv$dispatching_controls.controls.dispatching_priority_time [dp].minimum_time := 0;
16C 3504          IFEND;
16C 3505          IF jmv$job_scheduler_table.cpu_dispatching_allocation [dp].maximum <> 100 THEN
17E 3506            tmv$dispatching_controls.controls.dispatching_priority_time [dp].
17E 3507                  maximum_time := (normalized_interval) * jmv$job_scheduler_table.
17E 3508                  cpu_dispatching_allocation [dp].maximum;
17E 3509            tmv$dispatching_controls.maximums_defined := tmv$dispatching_controls.maximums_defined +
1AE 3510                  $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
1AE 3511          ELSE
1AE 3512            tmv$dispatching_controls.controls.dispatching_priority_time [dp].maximum_time :=
1C2 3513                  tmv$dispatching_controls.controls.time_left_in_interval;
1C2 3514          IFEND;
1C2 3515          IF jmv$job_scheduler_table.cpu_dispatching_allocation [dp].enforce_maximum THEN
1D0 3516            tmv$dispatching_controls.enforce_maximums := tmv$dispatching_controls.enforce_maximums +
1EA 3517                  $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
1EA 3518          IFEND;
1EA 3519        FOREND;
1EE 3520        tmv$dispatching_control_sets.minimums_to_satisfy := tmv$dispatching_controls.minimums_to_satisfy;
1EE 3521        tmv$dispatching_control_time := tmv$dispatching_controls.controls;
204 3522      IFEND;
204 3523
```


NOS/VE Job Management : job scheduler monitor mode
jmp$change_dispatching_alloc

```
204 3524 { Calculate the dispatching priority integers used by task switch and ready task to determine
204 3525 { which dispatching priority is the highest.
204 3526
204 3527      tmp$calculate_dct_priority_int;
20C 3528
20C 3529      local_set := tmv$dispatching_control_sets;
20C 3530
20C 3531      FOR dp := jmc$priority_p1 TO jmc$priority_p8 DO
218 3532        local_set.ready_tasks := $jmt$dispatching_priority_set [jmc$dp_conversion - dp];
218 3533        local_set.minimums_to_satisfy := local_set.minimums_to_satisfy * local_set.ready_tasks;
218 3534        local_set.ready_tasks := local_set.ready_tasks XOR local_set.minimums_to_satisfy;
218 3535        #UNCHECKED_CONVERSION (local_set, tmv$dispatch_priority_integer [dp]);
218 3536      FOREND;
250 3537
250 3538      tmp$clear_lock (tmv$pt1_lock);
288 3539
288 3540      IF dp_unblocked THEN
28C 3541        jmp$set_scheduler_event (jmc$examine_swapin_queue);
28E 3542        jmp$set_scheduler_event (jmc$examine_input_queue);
2EC 3543      IFEND;
2EC 3544
2EC 3545    PROCEND jmp$change_dispatching_alloc;
  0 3546
```

NOS/VE Job Management : job scheduler monitor mode
jmp$change_dispatching_mtr_req

```
      0  3548
      0  3549 { PURPOSE:
      0  3550 {   This procedure changes the dispatching control information in the service class table
      0  3551 {   and resets the dispatching control information for all jobs in classes being changed.
      0  3552 { DESIGN:
      0  3553 {   The PTL lock must be set while the table is being changed and affected job updated to
      0  3554 {   prevent task switch from using obsolete/uninitialized dispatching control information.
      0  3555
      0  3556   PROCEDURE jmp$change_dispatching_mtr_req;
      0  3557
      0  3558     VAR
      0  3559       changes_pointer: ^jmt$dispatching_control_changes,
      0  3560       circular_service: array [jmt$service_class_index] of integer,
      0  3561       class: jmt$service_class_index,
      0  3562       classes_changed: jmt$service_class_set,
      0  3563       dispatching_control_index: jmt$dispatching_control_index,
      0  3564       dispatching_control_p: ^jmt$dispatching_control,
      0  3565       ijl_bn: jmt$ijl_block_number,
      0  3566       ijl_bi: jmt$ijl_block_index,
      0  3567       ijl_ordinal: jmt$ijl_ordinal,
      0  3568       ijle_p: ^jmt$initiated_job_list_entry,
      0  3569       service_used: integer;
      0  3570
      0  3571     classes_changed := $jmt$service_class_set [];
     12  3572
     12  3573 { Set the ptl lock so that task switch cannot be accessing the service class attribute table
     12  3574
     12  3575     tmp$set_lock (tmv$ptl_lock);
     4C  3576
     4C  3577 { Change the service class attribute table
     4C  3578
     4C  3579     changes_pointer := jmv$change_dispatching_list.dispatching_control_changes_p;
     4C  3580
     4C  3581     WHILE changes_pointer <> NIL DO
     60  3582       class := changes_pointer^.change_service_class;
     60  3583       classes_changed := classes_changed + $jmt$service_class_set [class];
     60  3584       dispatching_control_p := ^changes_pointer^.dispatching_control_info;
     60  3585       jmv$service_classes [class]^.attributes.dispatching_control := dispatching_control_p^;
     84  3586
     84  3587       circular_service [class] := 0;
     84  3588
     84  3589     /calculate_circular_service/
     84  3590       FOR dispatching_control_index := jmc$max_dispatching_control DOWNTO jmc$min_dispatching_control DO
     96  3591         IF dispatching_control_p^ [dispatching_control_index].set_defined THEN
     A6  3592           IF dispatching_control_p^ [dispatching_control_index].service_limit <>
     B4  3593                 jmc$dc_maximum_service_limit THEN
     B4  3594             circular_service [class] := circular_service [class] +
     C6  3595                   dispatching_control_p^ [dispatching_control_index].service_limit;
     C6  3596           ELSE
     C6  3597             EXIT /calculate_circular_service/;
     CA  3598           IFEND;
     CA  3599         IFEND;
     CA  3600       FOREND /calculate_circular_service/;
     D0  3601       changes_pointer := changes_pointer^.dispatching_control_changes_p;
     D0  3602     WHILEND;
```


NOS/VE Job Management : job scheduler monitor mode
jmp$change_dispatching_mtr_req

```
     DA  3603
     DA  3604 { Scan the ijl to find all jobs belonging to classes that have been changed--those jobs may need to have
     DA  3605 { their dispatching priority reset.  If the dispatching control sets are circular, MOD the service used
     DA  3606 { before calling tmp$reset_dispatching_control.  For batch jobs, total job service is used; for interactive
     DA  3607 { jobs, use zero for the service.  Interactive jobs should be reset to the first dispatching control set.
     DA  3608
     DA  3609     ijle_p := NIL;
     DA  3610
     DA  3611   /scan_ijl/
     DA  3612     FOR ijl_bn := LOWERBOUND (jmv$ijl_p.block_p^) TO jmv$ijl_p.max_block_in_use DO
     EE  3613       IF (jmv$ijl_p.block_p^ [ijl_bn].index_p <> NIL) THEN
    100  3614         ijl_ordinal.block_number := ijl_bn;
    100  3615         FOR ijl_bi := LOWERVALUE (jmt$ijl_block_index) TO UPPERVALUE (jmt$ijl_block_index) DO
    112  3616           ijl_ordinal.block_index := ijl_bi;
    112  3617           IF ijl_ordinal <> jmv$system_ijl_ordinal THEN
    12A  3618             jmp$get_ijle_p (ijl_ordinal, ijle_p);
    12A  3619             IF ijle_p^.job_scheduler_data.service_class IN classes_changed THEN
    156  3620               IF ijle_p^.job_mode = jmc$batch THEN
    15E  3621                 service_used := ijle_p^.statistics.cp_time.time_spent_in_job_mode +
    15E  3622                       ijle_p^.statistics.cp_time.time_spent_in_mtr_mode -
    15E  3623                       ijle_p^.dispatching_control.cp_service_at_class_switch;
    15E  3624                 IF circular_service [ijle_p^.job_scheduler_data.service_class] <> 0 THEN
    17A  3625                   service_used := service_used MOD circular_service
    182  3626                         [ijle_p^.job_scheduler_data.service_class];
    182  3627                 IFEND;
    186  3628               ELSE
    186  3629                 service_used := 0;
    18A  3630               IFEND;
    18A  3631               tmp$reset_dispatching_control (ijle_p, ijl_ordinal, service_used, FALSE);
    1AA  3632             IFEND;
    1AA  3633           IFEND;
    1AA  3634         FOREND;
    1B2  3635       IFEND;
    1B2  3636     FOREND /scan_ijl/;
    1B8  3637
    1B8  3638     tmp$clear_lock (tmv$ptl_lock);
    1F2  3639
    1F2  3640   PROCEND jmp$change_dispatching_mtr_req;
      0  3641
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL, INLINE] jmp$decrement_swapped_job_count

```
o  3643
o  3644   PROCEDURE [XDCL, INLINE] jmp$decrement_swapped_job_count
4  3645     (     ijle_p: ^jmt$initiated_job_list_entry);
4  3646
4  3647     VAR
4  3648       job_class: jmt$job_class,
4  3649       service_class: jmt$service_class_index;
4  3650
4  3651     job_class := ijle_p^.job_scheduler_data.job_class;
4  3652     service_class := ijle_p^.job_scheduler_data.service_class;
4  3653     jmv$job_counts.job_class_counts [job_class].swapped_jobs :=
4  3654           jmv$job_counts.job_class_counts [job_class].swapped_jobs - 1;
4  3655     jmv$job_counts.service_class_counts [service_class].swapped_jobs := jmv$job_counts.
4  3656           service_class_counts [service_class].swapped_jobs - 1;
4  3657
4  3658   PROCEND jmp$decrement_swapped_job_count;
o  3659
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL, INLINE] jmp$increment_swapped_job_count

```
o  3661
o  3662   PROCEDURE [XDCL, INLINE] jmp$increment_swapped_job_count
4  3663     (     ijle_p: ^jmt$initiated_job_list_entry);
4  3664
4  3665     VAR
4  3666       job_class: jmt$job_class,
4  3667       service_class: jmt$service_class_index;
4  3668
4  3669     job_class := ijle_p^.job_scheduler_data.job_class;
4  3670     service_class := ijle_p^.job_scheduler_data.service_class;
4  3671     jmv$job_counts.job_class_counts [job_class].swapped_jobs :=
4  3672           jmv$job_counts.job_class_counts [job_class].swapped_jobs + 1;
4  3673     jmv$job_counts.service_class_counts [service_class].swapped_jobs := jmv$job_counts.
4  3674           service_class_counts [service_class].swapped_jobs + 1;
4  3675
4  3676   PROCEND jmp$increment_swapped_job_count;
o  3677
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$mtr_job_scheduler_requests

```
 0  3679
 0  3680   PROCEDURE [XDCL] jmp$mtr_job_scheduler_requests
 4  3681     (VAR request_block: jmt$rb_scheduler_requests);
 4  3682
 4  3683     request_block.status.normal := TRUE;
 4  3684
 4  3685 { Process the job scheduler sub requests.
 4  3686
 4  3687     CASE request_block.sub_reqcode OF
4E  3688     = jmc$src_operator_swap_in =
4E  3689       jmp$process_oper_swapin_mtr_req (request_block.ijl_ordinal, request_block.status);
64  3690
64  3691     = jmc$src_idling_advance_swaps =
64  3692       jmp$process_idling_adv_swaps;
6E  3693
6E  3694     = jmc$src_class_switch =
6E  3695       jmp$process_class_switch (request_block);
7E  3696
7E  3697     = jmc$src_change_dispatching_ctrl =
7E  3698       jmp$change_dispatching_mtr_req;
88  3699
88  3700     = jmc$src_cleanup_unrecovered_job =
88  3701       jmp$process_unrecovered_job (request_block);
A4  3702
A4  3703     = jmc$src_sched_profile_loading =
A4  3704       jmp$set_sched_profile_loading;
AE  3705
AE  3706     = jmc$src_dispatching_allocation =
AE  3707       jmp$change_dispatching_alloc;
B8  3708
B8  3709     = jmc$src_swapin_recovered_jobs =
B8  3710       jmp$mtr_swapin_recovered_jobs;
C2  3711
C2  3712     ELSE
C2  3713       mtp$set_status_abnormal ('JM', jme$invalid_scheduler_request, request_block.status);
D8  3714     CASEND;
D8  3715
D8  3716   PROCEND jmp$mtr_job_scheduler_requests;
 0  3717
```


NOS/VE Job Management : job scheduler monitor mode
jmp$mtr_swapin_recovered_jobs

```
 0  3719
 0  3720 { PURPOSE:
 0  3721 {   This procedure scans the IJL and readies all jobs so that they can swapin for job recovery.
 0  3722 { DESIGN:
 0  3723 {   The PTL lock is set to prevent any kind of ready task being processed asynchronously.
 0  3724 {   *** Discuss whether this is necessary at the code review.
 0  3725
 0  3726   PROCEDURE jmp$mtr_swapin_recovered_jobs;
 0  3727
 0  3728     VAR
 0  3729       ijl_bn: jmt$ijl_block_number,
 0  3730       ijl_bi: jmt$ijl_block_index,
 0  3731       ijl_ordinal: jmt$ijl_ordinal,
 0  3732       ijle_p: ^jmt$initiated_job_list_entry;
 0  3733
 0  3734     tmp$set_lock (tmv$ptl_lock);
3E  3735
3E  3736     FOR ijl_bn := LOWERBOUND (jmv$ijl_p.block_p^) TO jmv$ijl_p.max_block_in_use DO
52  3737       IF jmv$ijl_p.block_p^ [ijl_bn].index_p <> NIL THEN
64  3738         FOR ijl_bi := LOWERVALUE (jmt$ijl_block_index) TO UPPERVALUE (jmt$ijl_block_index) DO
6E  3739           ijl_ordinal.block_number := ijl_bn;
6E  3740           ijl_ordinal.block_index := ijl_bi;
6E  3741           jmp$get_ijle_p (ijl_ordinal, ijle_p);
6E  3742           IF ijle_p^.entry_status <> jmc$ies_entry_free THEN
A2  3743             IF jmc$dsw_job_recovery IN ijle_p^.delayed_swapin_work THEN
AE  3744               jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_swapped);
134 3745               jmp$ready_task_in_swapped_job (ijl_ordinal, ijle_p);
14C 3746             IFEND;
14C 3747           IFEND;
14C 3748         FOREND;
154 3749       IFEND;
154 3750     FOREND;
15A 3751
15A 3752     tmp$clear_lock (tmv$ptl_lock);
194 3753
194 3754   PROCEND jmp$mtr_swapin_recovered_jobs;
 0  3755
```

NOS/VE Job Management : job scheduler monitor mode
jmp$process_class_switch

```
      0  3757
      0  3758 { PURPOSE:
      0  3759 {   This procedure updates dispatching control information when a job switches
      0  3760 {   service classes.  The PTL lock must be set while the dispatching control
      0  3761 {   information is changed to prevent task switch from referencing obsolete/
      0  3762 {   uninitialized information.
      0  3763
      0  3764   PROCEDURE jmp$process_class_switch
      0  3765     (VAR rb: jmt$rb_scheduler_requests);
      0  3766
      0  3767     VAR
      0  3768       old_class: jmt$service_class_index,
      0  3769       service_class_p: ^jmt$service_class_attributes,
      0  3770       ijle_p: ^jmt$initiated_job_list_entry,
      0  3771       ijlo: jmt$ijl_ordinal;
      0  3772
      0  3773     jmp$find_jsn (rb.system_supplied_name, ijle_p, ijlo);
     3A  3774
     3A  3775     tmp$set_lock (tmv$ptl_lock);
     74  3776
     74  3777     IF (ijle_p <> NIL) AND (ijle_p^.entry_status > jmc$ies_job_in_memory_non_swap) AND
     98  3778           (NOT jmv$sched_profile_is_loading) THEN
     98  3779       old_class := ijle_p^.job_scheduler_data.service_class;
     98  3780
     98  3781       IF rb.old_service_class = jmc$null_service_class THEN
     A4  3782         rb.old_service_class := old_class;
     A4  3783         rb.old_service_accumulator := ijle_p^.job_scheduler_data.service_accumulator;
     B4  3784       ELSEIF (rb.old_service_class <> old_class) OR (rb.old_service_accumulator >
     C4  3785             ijle_p^.job_scheduler_data.service_accumulator) THEN
     C4  3786         tmp$clear_lock (tmv$ptl_lock);
     FC  3787         RETURN;
     FE  3788       IFEND;
     FE  3789
     FE  3790       jmp$update_service_class_stats (ijle_p);
    112  3791
    112  3792       IF (ijle_p^.entry_status > jmc$ies_swapped_in) THEN
    120  3793         jmp$decrement_swapped_job_count (ijle_p);
    120  3794         ijle_p^.job_scheduler_data.service_class := rb.new_service_class;
    120  3795         jmp$increment_swapped_job_count (ijle_p);
    184  3796       ELSE
    184  3797         ijle_p^.job_scheduler_data.service_class := rb.new_service_class;
    18C  3798       IFEND;
    18C  3799
    18C  3800       jmv$job_counts.service_class_counts [old_class].scheduler_initiated_jobs :=
    18C  3801             jmv$job_counts.service_class_counts [old_class].scheduler_initiated_jobs - 1;
    18C  3802
    18C  3803       jmv$job_counts.service_class_counts [rb.new_service_class].scheduler_initiated_jobs :=
    18C  3804             jmv$job_counts.service_class_counts [rb.new_service_class].scheduler_initiated_jobs + 1;
    18C  3805
    18C  3806       service_class_p := ^jmv$service_classes [rb.new_service_class]^.attributes;
    18C  3807
    18C  3808       ijle_p^.job_scheduler_data.service_accumulator := 0;
    18C  3809       ijle_p^.dispatching_control.dispatching_control_index := jmc$min_dispatching_control;
    18C  3810       IF ijle_p^.dispatching_control.dispatching_priority = ijle_p^.scheduling_dispatching_priority THEN
    1D4  3811         ijle_p^.scheduling_dispatching_priority := service_class_p^.
```


NOS/VE Job Management : job scheduler monitor mode
jmp$process_class_switch

```
    1DC  3812               dispatching_control [jmc$min_dispatching_control].dispatching_priority;
    1DC  3813       IFEND;
    1DC  3814       ijle_p^.dispatching_control.dispatching_priority := service_class_p^.
    1DC  3815             dispatching_control [jmc$min_dispatching_control].dispatching_priority;
    1DC  3816       ijle_p^.dispatching_control.service_remaining := service_class_p^.
    1DC  3817             dispatching_control [jmc$min_dispatching_control].service_limit;
    1DC  3818       ijle_p^.dispatching_control.cp_service_at_class_switch :=
    1DC  3819             ijle_p^.statistics.cp_time.time_spent_in_job_mode +
    1DC  3820             ijle_p^.statistics.cp_time.time_spent_in_mtr_mode;
    1DC  3821       tmp$update_job_task_environment (ijle_p, ijlo, tmc$fnx_job);
    21A  3822
    21A  3823 { Check active job limits for the new class; cause a job to swapout if necessary.
    21A  3824
    21A  3825       IF (jmv$job_counts.service_class_counts [rb.new_service_class].scheduler_initiated_jobs -
    234  3826             jmv$job_counts.service_class_counts [rb.new_service_class].swapped_jobs) >
    234  3827             service_class_p^.maximum_active_jobs THEN
    234  3828         jmp$set_scheduler_event (jmc$swap_jobs_for_lower_maxaj);
    262  3829       IFEND;
    262  3830     IFEND;
    262  3831
    262  3832     tmp$clear_lock (tmv$ptl_lock);
    29A  3833
    29A  3834   PROCEND jmp$process_class_switch;
      0  3835
```

NOS/VE Job Management : job scheduler monitor mode
jmp$process_idling_adv_swaps

```
        0  3837
        0  3838   PROCEDURE jmp$process_idling_adv_swaps;
        0  3839
        0  3840     VAR
        0  3841       ijl_ordinal: jmt$ijl_ordinal,
        0  3842       ijle_p: ^jmt$initiated_job_list_entry,
        0  3843       next_ijl_ordinal: jmt$ijl_ordinal;
        0  3844
        0  3845     ijl_ordinal := jsv$ijl_swap_queue_list [jsc$isqi_swapped_io_completed].forward_link;
        4  3846
        4  3847     WHILE ijl_ordinal <> jmv$null_ijl_ordinal DO
       1E  3848       jmp$get_ijle_p (ijl_ordinal, ijle_p);
       1E  3849       next_ijl_ordinal := ijle_p^.swap_queue_link.forward_link;
       1E  3850       jsp$monitor_advance_swap (ijl_ordinal);
       56  3851       ijl_ordinal := next_ijl_ordinal;
       56  3852     WHILEND;
       6E  3853
       6E  3854   PROCEND jmp$process_idling_adv_swaps;
        0  3855
```


NOS/VE Job Management : job scheduler monitor mode
jmp$process_oper_swapin_mtr_req

```
        0  3857
        0  3858 { PURPOSE:
        0  3859 {   Process an operator swapin job request.
        0  3860 { DESIGN:
        0  3861 {   Re-check entry status.  Entry status was operator force out when the monitor request was issued.
        0  3862 {   The following (very unlikely) timing sequence could occur though:
        0  3863 {     The job was swapping in (swapin I/O was active) when the operator swapout occurred.  Entry status
        0  3864 {     was changed to operator force out.  The operator swapped the job in right away (I/O was still
        0  3865 {     active); the job mode operator swapin code found entry status still set to operator force out.
        0  3866 {     Before exchanging to monitor for the swapin request, process I/O completions executed.  Swapin I/O
        0  3867 {     errors would cause the entry status to be changed to system force out.
        0  3868 {   If entry status is still operator force out, then change entry_status to swapped.  Call
        0  3869 {   jmp$ready_task_in_swapped_job if the job has any ready tasks.
        0  3870
        0  3871
        0  3872   PROCEDURE jmp$process_oper_swapin_mtr_req
        0  3873     (     ijl_ordinal: jmt$ijl_ordinal;
        0  3874      VAR status: syt$monitor_status);
        0  3875
        0  3876     VAR
        0  3877       ijle_p: ^jmt$initiated_job_list_entry;
        0  3878
        0  3879     jmp$get_ijle_p (ijl_ordinal, ijle_p);
        4  3880
        4  3881     IF ijle_p^.entry_status = jmc$ies_operator_force_out THEN
       32  3882
       32  3883 [ Set the PTL lock to synchronize with the dispatcher/ready task path.
       32  3884
       32  3885       status.normal := TRUE;
       32  3886       tmp$set_lock (tmv$ptl_lock);
       70  3887       jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_swapped);
       F8  3888       IF ijle_p^.statistics.ready_task_count > 0 THEN
      104  3889         ijle_p^.job_scheduler_data.swapin_q_priority_timestamp := 0;
      104  3890         jmp$ready_task_in_swapped_job (ijl_ordinal, ijle_p);
      11E  3891       IFEND;
      11E  3892       tmp$clear_lock (tmv$ptl_lock);
      15A  3893
      15A  3894     ELSE
      15A  3895       IF ijle_p^.entry_status <> jmc$ies_system_force_out THEN
      160  3896         mtp$error_stop ('OPER SWAPIN REQUEST ERROR');
      180  3897       IFEND;
      180  3898       mtp$set_status_abnormal ('JM', jme$job_dead_cannot_swap, status);
      192  3899     IFEND;
      192  3900
      192  3901   PROCEND jmp$process_oper_swapin_mtr_req;
        0  3902
```

NOS/VE Job Management : job scheduler monitor mode
jmp$process_unrecovered_job

```
     0  3904
     0  3905 { PURPOSE:
     0  3906 {   This procedure relinks a job to the null swapping queue and changes job class counts
     0  3907 {   when a job must be terminated during job recovery.  The PTL entries for the tasks of
     0  3908 {   the job are freed.  The two reasons for the termination are that a job class is not
     0  3909 {   defined for a job or a job could not be swapped in for recovery due to an io error.
     0  3910
     0  3911   PROCEDURE jmp$process_unrecovered_job
     0  3912     (     rb: jmt$rb_scheduler_requests);
     0  3913
     0  3914     VAR
     0  3915       ijle_p: ^jmt$initiated_job_list_entry;
     0  3916
     0  3917     jmp$get_ijle_p (rb.ijl_ordinal, ijle_p);
     4  3918
     4  3919     tmp$set_lock (tmv$ptl_lock);
    64  3920
    64  3921     jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].scheduler_initiated_jobs :=
    64  3922           jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].
    64  3923           scheduler_initiated_jobs - 1;
    64  3924     jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].swapped_jobs :=
    64  3925           jmv$job_counts.service_class_counts [ijle_p^.job_scheduler_data.service_class].swapped_jobs - 1;
    64  3926     jmv$job_counts.job_class_counts [ijle_p^.job_scheduler_data.job_class].swapped_jobs :=
    64  3927           jmv$job_counts.job_class_counts [ijle_p^.job_scheduler_data.job_class].swapped_jobs - 1;
    64  3928
    64  3929     jsp$relink_swap_queue (rb.ijl_ordinal, ijle_p, jsc$isqi_null);
    BA  3930
    BA  3931     tmp$free_unrecovered_tasks (ijle_p);
    CE  3932
    CE  3933     tmp$clear_lock (tmv$ptl_lock);
   104  3934
   104  3935   PROCEND jmp$process_unrecovered_job;
     0  3936
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$ready_task_in_swapped_job

```
     0  3938
     0  3939   PROCEDURE [XDCL] jmp$ready_task_in_swapped_job
     0  3940     (     ijl_ord: jmt$ijl_ordinal;
     0  3941           ijle_p: ^jmt$initiated_job_list_entry);
     0  3942
     0  3943     VAR
     0  3944       current_time: integer,
     0  3945       service_class: jmt$service_class_index,
     0  3946       status: syt$monitor_status,
     0  3947       swap_stats_p: ^jmt$service_class_swap_stats,
     0  3948       think_time: integer;
     0  3949
     0  3950     #KEYPOINT (osk$entry, 0, jmk$ready_task_in_swapped_job);
     8  3951
     8  3952 { If a job with a memory reserve request posted has a task go ready, cancel the request.
     8  3953 { The ready task may be because of a user interrupt; do not wait for the requested memory
     8  3954 { to become available.
     8  3955
     8  3956     IF ijle_p^.memory_reserve_request.requested_page_count > 0 THEN
    14  3957       ijle_p^.memory_reserve_request.requested_page_count := 0;
    14  3958       jmv$job_sched_events_selected [jmc$examine_swapin_queue] := TRUE;
    14  3959       jmp$set_scheduler_event (jmc$examine_swapin_queue];
    4E  3960     IFEND;
    4E  3961
    4E  3962     current_time := #FREE_RUNNING_CLOCK (0);
    56  3963     IF ijle_p^.entry_status = jmc$ies_job_swapped THEN
    60  3964
    60  3965       think_time := current_time - (ijle_p^.estimated_ready_time - ijle_p^.last_think_time);
    60  3966       IF (think_time > jmv$max_think_time) THEN
    7A  3967         ijle_p^.last_think_time := jmv$max_think_time;
    82  3968       ELSEIF (think_time > jmv$min_think_time) THEN
    8A  3969         ijle_p^.last_think_time := think_time;
    8E  3970       IFEND;
    8E  3971
    8E  3972       ijle_p^.swap_data.timestamp := current_time;
    8E  3973
    8E  3974       service_class := ijle_p^.job_scheduler_data.service_class;
    8E  3975
    8E  3976       tmp$set_lock (jmv$service_class_stats_lock);
    D6  3977       swap_stats_p := ^jmv$service_classes [service_class]^.statistics.swap_stats;
    D6  3978       swap_stats_p^.swap_to_ready_time := swap_stats_p^.swap_to_ready_time +
    D6  3979             (current_time - ijle_p^.swap_data.swapout_timestamp);
    D6  3980       swap_stats_p^.swap_to_ready_count := swap_stats_p^.swap_to_ready_count + 1;
    D6  3981       tmp$clear_lock (jmv$service_class_stats_lock);
   13C  3982
   13C  3983 { If possible, swap the job in immediately through the monitor interface; otherwise notify job mode
   13C  3984 { scheduler to swap the job in.  If the dispatching priority of the job is blocked, the swapin must
   13C  3985 { be handled by job mode scheduler.
   13C  3986
   13C  3987       IF (NOT jmv$prevent_activation_of_jobs) AND (NOT jmv$job_scheduler_event [jmc$examine_input_queue]) AND
   13C  3988             (ijle_p^.job_scheduler_data.swapout_reason <> jmc$sr_thrashing) AND
   13C  3989             (swapin_queue_empty (ijle_p^.scheduling_dispatching_priority)) AND
   13C  3990             (ijle_p^.swap_status <= jmc$iss_swapped_io_complete) AND (jmv$number_free_ajl_entries > 0) AND
   238  3991             (NOT too_many_active_jobs_for_class (service_class)) THEN
   238  3992
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$ready_task_in_swapped_job

```
238  3993          jsp$monitor_swap_in (ijl_ord);
24E  3994
24E  3995 { Reset fields for scheduler data -- only reset service accumulator since swap if the job used
24E  3996 { its whole guaranteed service allotment the last time it was swapped in.
24E  3997
24E  3998          IF ijle_p^.job_scheduler_data.guaranteed_service_remaining = 0 THEN
256  3999            ijle_p^.job_scheduler_data.service_accumulator_since_swap := 0;
25A  4000          IFEND;
25A  4001          ijle_p^.job_scheduler_data.guaranteed_service_remaining := 0;
25A  4002          ijle_p^.job_scheduler_data.priority := jmv$service_classes [service_class]^.attributes.
276  4003                scheduling_priority.maximum;
276  4004
276  4005        ELSE {The swapin could not take place in monitor so notify job mode scheduler to handle the swapin.}
276  4006          jmp$change_ijl_entry_status (ijle_p, jmc$ies_ready_task);
2F2  4007          insert_job_in_ready_task_list (ijl_ord, ijle_p);
308  4008          jmv$job_scheduler_event [jmc$ready_task_in_job] := TRUE;
308  4009          IF (NOT (service_class IN jmv$classes_in_maxaj_limit_wait)) THEN
31C  4010            jmv$job_scheduler_event [jmc$examine_swapin_queue] := TRUE;
31C  4011            IF (NOT (service_class IN jmv$classes_in_resource_wait)) THEN
32E  4012              jmv$job_sched_events_selected [jmc$examine_swapin_queue] := TRUE;
32E  4013              tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
34C  4014            IFEND;
34C  4015          IFEND;
34C  4016        IFEND;
34C  4017      IFEND;
34C  4018
34C  4019      #KEYPOINT (osk$exit, 0, jmk$ready_task_in_swapped_job);
350  4020
350  4021    PROCEND jmp$ready_task_in_swapped_job;
  0  4022
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$recognize_job_dead

```
  0  4024
  0  4025 { PURPOSE:
  0  4026 {   This procedure is called when a swapin I/O error occurs to change a job's status to reflect that the
  0  4027 {   job cannot swapin.
  0  4028 { DESIGN:
  0  4029 {   The current status is checked before changing it to system_force_out.  The various statuses
  0  4030 {   are handled as follows:
  0  4031 {     Free, Terminating, In memory non swap, In memory, System force out -- CANNOT possibly be these statuses.
  0  4032 {     Swapin in progress, Swapped, Operator force out -- Change the entry status to system force out.
  0  4033 {          Swapin in progress is the usual case; swapped requires an IDLE_SYSTEM swapout while the swapin
  0  4034 {          I/O was active; operator force out would be set if the operator swapped out the job while swapin
  0  4035 {          I/O was active.
  0  4036 {     Job damaged -- do not change the entry status; job damaged is more important to know.
  0  4037 {     Ready task, Swapin candidate -- do not change the entry status; only JOB SCHEDULER can change
  0  4038 {          these statuses.  The job will be swapped in again and the I/O error will be processed then.
  0  4039 {          The possible timing for a job to be in one of these states is very remote:  An IDLE_SYSTEM
  0  4040 {          swapout would have to be processed while swapin I/O was active (a swap cannot be aborted
  0  4041 {          while swapin I/O is active).  Then RESUME_SYSTEM would have to queue the job to swap in again
  0  4042 {          while the original swapin I/O was still active.
  0  4043 {
  0  4044 {   The PTL must be locked while changing the entry status because the swapped job count will be changed
  0  4045 {   in the swapin in progress ---> system force out transition.
  0  4046
  0  4047   PROCEDURE [XDCL] jmp$recognize_job_dead
  0  4048     (     ijl_o: jmt$ijl_ordinal);
  0  4049
  0  4050     VAR
  0  4051       ijl_p: ^jmt$initiated_job_list_entry;
  0  4052
  0  4053     jmp$get_ijle_p (ijl_o, ijl_p);
  4  4054
  4  4055     IF (ijl_p^.entry_status = jmc$ies_swapin_in_progress) OR (ijl_p^.entry_status = jmc$ies_job_swapped) OR
 4O  4056           (ijl_p^.entry_status = jmc$ies_operator_force_out) THEN
 4O  4057
 4O  4058       tmp$set_lock (tmv$ptl_lock);
 7A  4059       jmp$change_ijl_entry_status (ijl_p, jmc$ies_system_force_out);
102  4060       tmp$clear_lock (tmv$ptl_lock);
13C  4061
13C  4062       IF jmc$dsw_job_recovery IN ijl_p^.delayed_swapin_work THEN
14C  4063         ijl_p^.delayed_swapin_work := ijl_p^.delayed_swapin_work +
14C  4064               $jmt$delayed_swapin_work [jmc$dsw_recovery_swap_io_error];
14C  4065         jmp$set_scheduler_event (jmc$recovery_swap_io_error);
184  4066       IFEND;
184  4067     IFEND;
184  4068
184  4069   PROCEND jmp$recognize_job_dead;
  0  4070
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$recognize_thrashing

```
       0  4072
       0  4073   PROCEDURE [XDCL] jmp$recognize_thrashing;
       0  4074
       0  4075     VAR
       0  4076       ajlo: jmt$ajl_ordinal,
       0  4077       count: jmt$ajl_ordinal;
       0  4078
       0  4079     count := 0;
       4  4080
       4  4081 { Determine if there is more than one user job active.
       4  4082
       4  4083   /count_active_jobs/
       4  4084     FOR ajlo := jmv$system_ajl_ordinal + 1 TO jmv$max_ajl_ordinal_in_use DO
      1E  4085       IF jmv$ajl_p^ [ajlo].in_use <> 0 THEN
      38  4086         count := count + 1;
      38  4087         IF count = 2 THEN
      40  4088           EXIT /count_active_jobs/;
      44  4089         IFEND;
      44  4090       IFEND;
      44  4091     FOREND /count_active_jobs/;
      48  4092
      48  4093 { If there is more than one user job active, cause scheduler to swap for thrashing.  If there is only
      48  4094 { one user job active, cause mmp$periodic_call to run so the jobs working set can be shrunk to fit in memory.
      48  4095
      48  4096     IF count = 2 THEN
      4E  4097       jmp$set_scheduler_event (jmc$system_is_thrashing);
      84  4098     ELSE
      84  4099       mmv$reduce_jws_for_thrashing := TRUE;
      84  4100       mmp$nudge_periodic_call;
      9E  4101     IFEND;
      9E  4102
      9E  4103   PROCEND jmp$recognize_thrashing;
       0  4104
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$reset_job_to_swapped_out

```
       0  4106
       0  4107 { PURPOSE:
       0  4108 {   This procedure is called from swapper when a swapin could not be completed because there
       0  4109 {   was not enough memory or there was not a free AJL ordinal.  The entry status has
       0  4110 {   to be swapin in progress when this procedure is called.
       0  4111
       0  4112   PROCEDURE [XDCL] jmp$reset_job_to_swapped_out
       0  4113     (     ijl_o: jmt$ijl_ordinal);
       0  4114
       0  4115     VAR
       0  4116       ijl_p: ^jmt$initiated_job_list_entry,
       0  4117       status: syt$monitor_status;
       0  4118
       0  4119     jmp$get_ijle_p (ijl_o, ijl_p);
       4  4120     IF ijl_p^.entry_status <> jmc$ies_swapin_in_progress THEN
      34  4121       mtp$error_stop ('RESET TO SWAPPED OUT ERROR');
      54  4122     IFEND;
      54  4123
      54  4124     tmp$set_lock (tmv$ptl_lock);
      8E  4125     jmp$change_ijl_entry_status (ijl_p, jmc$ies_ready_task);
     116  4126     insert_job_in_ready_task_list (ijl_o, ijl_p);
     12A  4127     tmp$clear_lock (tmv$ptl_lock);
     164  4128
     164  4129     jmv$job_scheduler_event [jmc$ready_task_in_job] := TRUE;
     164  4130     IF (NOT (ijl_p^.job_scheduler_data.service_class IN jmv$classes_in_maxaj_limit_wait)) THEN
     182  4131       jmv$job_scheduler_event [jmc$examine_swapin_queue] := TRUE;
     182  4132       IF (NOT (ijl_p^.job_scheduler_data.service_class IN jmv$classes_in_resource_wait)) THEN
     194  4133         tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
     1AE  4134       IFEND;
     1AE  4135     IFEND;
     1AE  4136
     1AE  4137   PROCEND jmp$reset_job_to_swapped_out;
       0  4138
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$resurrect_dead_jobs

```
       0  4140
       0  4141   PROCEDURE [XDCL] jmp$resurrect_dead_jobs;
       0  4142
       0  4143 { The purpose of this procedure is to find all jobs that have been marked as system_force_out
       0  4144 { because a disk unit was down, and find all jobs that could not be swapped completely because
       0  4145 { a disk unit was down.  This procedure is called whenenver a disk unit comes back up.
       0  4146 { Swapper will try to procede swapping the jobs normally.
       0  4147
       0  4148      VAR
       0  4149        call_job_swapper: boolean,
       0  4150        ijle_p: ^jmt$initiated_job_list_entry,
       0  4151        ijl_bn: jmt$ijl_block_number,
       0  4152        ijl_bi: jmt$ijl_block_index,
       0  4153        ijl_ordinal: jmt$ijl_ordinal,
       0  4154        status: syt$monitor_status;
       0  4155
       0  4156      call_job_swapper := FALSE;
       4  4157
       4  4158    /search_ijl/
       4  4159      FOR ijl_bn := LOWERBOUND (jmv$ijl_p.block_p^) TO jmv$ijl_p.max_block_in_use DO
      1A  4160        IF jmv$ijl_p.block_p^ [ijl_bn].index_p <> NIL THEN
      2C  4161          FOR ijl_bi := LOWERVALUE (jmt$ijl_block_index) TO UPPERVALUE (jmt$ijl_block_index) DO
      36  4162            ijle_p := ^jmv$ijl_p.block_p^ [ijl_bn].index_p^ [ijl_bi];
      36  4163            ijl_ordinal.block_number := ijl_bn;
      36  4164            ijl_ordinal.block_index := ijl_bi;
      36  4165
      36  4166            IF (ijle_p^.entry_status = jmc$ies_system_force_out) AND
      66  4167                  (ijle_p^.swap_data.swapping_io_error <> ioc$no_error) THEN
      66  4168
      66  4169 { The job was swapped out, swapping in when the io error occurred.  Try to swap the job in now.
      66  4170
      66  4171              tmp$set_lock (tmv$ptl_lock);
      A0  4172              jmp$change_ijl_entry_status (ijle_p, jmc$ies_ready_task);
     128  4173              insert_job_in_ready_task_list (ijl_ordinal, ijle_p);
     144  4174              tmp$clear_lock (tmv$ptl_lock);
     17C  4175
     17C  4176              jmv$job_scheduler_event [jmc$ready_task_in_job] := TRUE;
     17C  4177              jmv$job_scheduler_event [jmc$examine_swapin_queue] := TRUE;
     17C  4178              tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
     1A2  4179
     1A2  4180            ELSEIF (ijle_p^.swap_status = jmc$iss_swapped_io_cannot_init) OR
     1B2  4181                  (ijle_p^.swap_status = jmc$iss_job_allocate_swap_file) THEN
     1B2  4182
     1B2  4183 { The job was swapping out when the condition occurred.  Call job mode swapper to check on
     1B2  4184 { the state of the swap file and allocate it if necessary, then advance the swapout.
     1B2  4185
     1B2  4186              call_job_swapper := TRUE;
     1B8  4187
     1B8  4188            IFEND;
     1B8  4189          FOREND;
     1C0  4190        IFEND;
     1C0  4191      FOREND /search_ijl/;
     1C4  4192
     1C4  4193      IF call_job_swapper THEN
     1C8  4194        jmp$activate_job_mode_swapper;
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$resurrect_dead_jobs

```
     1D0  4195      IFEND;
     1D0  4196
     1D0  4197    PROCEND jmp$resurrect_dead_jobs;
       0  4198
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$set_entry_status_to_rt

```
        o  4200
        o  4201 { PURPOSE:
        o  4202 {   This procedure is called from swapper (job_mode_swapout) to set the entry status of the
        o  4203 {   job being swapped out to jmc$ies_ready_task and insert the job in the ready task list.
        o  4204 {   The caller has the PTL lock set.
        o  4205
        o  4206   PROCEDURE [XDCL] jmp$set_entry_status_to_rt
        o  4207     (     ijl_ordinal: jmt$ijl_ordinal;
        o  4208           ijle_p: ^jmt$initiated_job_list_entry);
        o  4209
        o  4210     VAR
        o  4211       status: syt$monitor_status;
        o  4212
        o  4213     jmp$change_ijl_entry_status (ijle_p, jmc$ies_ready_task);
       8C  4214     insert_job_in_ready_task_list (ijl_ordinal, ijle_p);
       AO  4215
       AO  4216     jmv$job_scheduler_event [jmc$ready_task_in_job] := TRUE;
       AO  4217     IF (NOT (ijle_p^.job_scheduler_data.service_class IN jmv$classes_in_maxaj_limit_wait)) THEN
       CO  4218       jmv$job_scheduler_event [jmc$examine_swapin_queue] := TRUE;
       CO  4219       IF (NOT (ijle_p^.job_scheduler_data.service_class IN jmv$classes_in_resource_wait)) THEN
       D2  4220         tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
       EC  4221       IFEND;
       EC  4222     IFEND;
       EC  4223
       EC  4224   PROCEND jmp$set_entry_status_to_rt;
        o  4225
```


NOS/VE Job Management : job scheduler monitor mode
jmp$set_job_terminated

```
        o  4227
        o  4228 { PURPOSE:
        o  4229 {   This procedure sets a job's entry status to terminating, and JOB SCHEDULER event to
        o  4230 {   process the terminating job.
        o  4231
        o  4232   PROCEDURE [XDCL] jmp$set_job_terminated
        4  4233     (     ijl_ordinal: jmt$ijl_ordinal;
        4  4234           ijle_p: ^jmt$initiated_job_list_entry);
        4  4235
        4  4236     jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_terminating);
       8E  4237     jmv$ijl_p.block_p^ [ijl_ordinal.block_number].terminated_job := TRUE;
       8E  4238     jmp$set_scheduler_event (jmc$job_terminated);
       D4  4239
       D4  4240   PROCEND jmp$set_job_terminated;
        o  4241
```

NOS/VE Job Management : job scheduler monitor mode
jmp$set_sched_profile_loading

```
      o  4243
      o  4244 { PURPOSE:
      o  4245 {   The purpose of this request is to set the flag which indicates that a
      o  4246 {   scheduling profile is being installed in the scheduler tables.
      o  4247
      o  4248   PROCEDURE jmp$set_sched_profile_loading;
      4  4249
      4  4250     jmv$sched_profile_is_loading := TRUE;
      4  4251
      4  4252   PROCEND jmp$set_sched_profile_loading;
      o  4253
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL, INLINE] jmp$set_scheduler_event

```
      o  4255
      o  4256   PROCEDURE [XDCL, INLINE] jmp$set_scheduler_event
      4  4257     (    event: jmt$job_scheduler_events);
      4  4258
      4  4259     VAR
      4  4260       status: syt$monitor_status;
      4  4261
      4  4262     IF NOT jmv$job_scheduler_event [event] THEN
     14  4263       jmv$job_scheduler_event [event] := TRUE;
     14  4264       IF jmv$job_sched_events_selected [event] THEN
     22  4265         tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
     3C  4266       IFEND;
     3C  4267     IFEND;
     3C  4268   PROCEND jmp$set_scheduler_event;
      o  4269
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$set_scheduler_memory_event

```
        0  4271
        0  4272    PROCEDURE [XDCL] jmp$set_scheduler_memory_event;
        0  4273
        0  4274      VAR
        0  4275        status: syt$monitor_status;
        0  4276
        0  4277      jmv$job_scheduler_event [jmc$needed_memory_available] := TRUE;
        4  4278      tmp$monitor_ready_system_task (tmc$stid_job_scheduler, status);
       28  4279
       28  4280    PROCEND jmp$set_scheduler_memory_event;
        0  4281
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$set_swapout_candidate

```
        0  4283
        0  4284    PROCEDURE [XDCL] jmp$set_swapout_candidate
        0  4285      (     ajl_o: jmt$ajl_ordinal;
        0  4286            swapout_reason: jmt$swapout_reasons);
        0  4287
        0  4288      VAR
        0  4289        ajle_p: ^jmt$active_job_list_entry,
        0  4290        guaranteed_service_quantum: jmt$service_accumulator,
        0  4291        ijle_p: ^jmt$initiated_job_list_entry,
        0  4292        ijl_ord: jmt$ijl_ordinal,
        0  4293        service_used: integer;
        0  4294
        0  4295      #KEYPOINT (osk$entry, 0, jmk$set_swapout_candidate);
        8  4296
        8  4297      ajle_p := ^jmv$ajl_p^ [ajl_o];
        8  4298      ijle_p := ajle_p^.ijle_p;
        8  4299      IF ijle_p^.entry_status = jmc$ies_job_in_memory THEN
       2E  4300        IF jmv$swap_jobs_in_long_wait THEN
       3A  4301          ijl_ord := ajle_p^.ijl_ordinal;
       3A  4302          ijle_p^.estimated_ready_time := #FREE_RUNNING_CLOCK (0) + ijle_p^.last_think_time;
       46  4303          jmp$calculate_service (ijle_p, service_used);
       68  4304          check_for_class_switch (ijle_p);
       76  4305          ijle_p^.job_scheduler_data.swapout_reason := swapout_reason;
       76  4306          IF (swapout_reason = jmc$sr_idle_dispatching) AND (jmv$service_classes
       A8  4307                [ijle_p^.job_scheduler_data.service_class]^.attributes.guaranteed_service_quantum =
       A8  4308                jmc$unlimited_service_accum) THEN
       A8  4309            ijle_p^.job_scheduler_data.guaranteed_service_remaining := jmc$unlimited_service_accum;
       B4  4310          ELSEIF (swapout_reason = jmc$sr_idle_dispatching) AND
       DE  4311                (ijle_p^.job_scheduler_data.service_accumulator_since_swap <
       DE  4312                jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.attributes.
       DE  4313                guaranteed_service_quantum) THEN
       DE  4314            ijle_p^.job_scheduler_data.guaranteed_service_remaining :=
       EA  4315                  guaranteed_service_quantum - ijle_p^.job_scheduler_data.service_accumulator_since_swap;
       EA  4316          IFEND;
       EA  4317          ijle_p^.job_scheduler_data.job_swap_counts.long_wait :=
       EA  4318                ijle_p^.job_scheduler_data.job_swap_counts.long_wait + 1;
       EA  4319          jsp$monitor_swap_out (ijl_ord);
      104  4320
      104  4321 { If the service class is at the maxaj limit, remove the class so a job
      104  4322 { with this service class can be activated.
      104  4323
      104  4324          IF (ijle_p^.job_scheduler_data.service_class IN jmv$classes_in_maxaj_limit_wait) THEN
      114  4325            remove_class_from_maxaj_limit (ijle_p^.job_scheduler_data.service_class);
      122  4326          IFEND;
      126  4327        ELSE
      126  4328          ajle_p^.job_is_good_swap_candidate := TRUE;
      12C  4329        IFEND;
      12C  4330      IFEND;
      12C  4331
      12C  4332      #KEYPOINT (osk$exit, 0, jmk$set_swapout_candidate);
      130  4333
      130  4334    PROCEND jmp$set_swapout_candidate;
        0  4335
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$subsystem_priority_change

```
 o  4337
 o  4338 { PURPOSE:
 o  4339 {   The purpose of this procedure is to set a scheduler event if a swapin candidate
 o  4340 {   job has had its scheduling dispatching priority changed because of subsystem locks.
 o  4341 { DESIGN:
 o  4342 {   The caller of this procedure must set the PTL lock.
 o  4343
 o  4344   PROCEDURE [XDCL] jmp$subsystem_priority_change
 4  4345     (     ijle_p: ^jmt$initiated_job_list_entry);
 4  4346
 4  4347     IF (ijle_p^.entry_status = jmc$ies_swapin_candidate) AND
1E  4348           (ijle_p^.scheduling_dispatching_priority > ijle_p^.dispatching_control.dispatching_priority) THEN
1E  4349       jmv$subsystem_priority_changes [ijle_p^.job_scheduler_data.service_class] := TRUE;
1E  4350       jmp$set_scheduler_event (jmc$subsystem_priority_change);
5E  4351     IFEND;
5E  4352
5E  4353   PROCEND jmp$subsystem_priority_change;
 o  4354
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$swap_non_dispatchable_job

```
 o  4356
 o  4357   PROCEDURE [XDCL] jmp$swap_non_dispatchable_job
 o  4358     (     ajl_ordinal: jmt$ajl_ordinal);
 o  4359
 o  4360     VAR
 o  4361       ijl_ordinal: jmt$ijl_ordinal,
 o  4362       ijle_p: ^jmt$initiated_job_list_entry;
 o  4363
 o  4364     ijle_p := jmv$ajl_p^ [ajl_ordinal].ijle_p;
 4  4365     ijl_ordinal := jmv$ajl_p^ [ajl_ordinal].ijl_ordinal;
 4  4366     jmp$set_swapout_candidate (ajl_ordinal, jmc$sr_idle_dispatching);
38  4367
38  4368 { Jmp$set_swapout_candidate swapped the job and caused entry status to be changed to job_swapped.
38  4369 { The job was artificially idled, so it must be put in the ready task list to swap back in.
38  4370
38  4371     jmv$swapped_idle_disp_count := jmv$swapped_idle_disp_count + 1;
38  4372     jmp$change_ijl_entry_status (ijle_p, jmc$ies_ready_task);
C4  4373     insert_job_in_ready_task_list (ijl_ordinal, ijle_p);
D2  4374     jmv$job_scheduler_event [jmc$ready_task_in_job] := TRUE;
D2  4375
D2  4376   PROCEND jmp$swap_non_dispatchable_job;
 o  4377
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_serv_class_stats_req

```
        o  4379
        o  4380 { PURPOSE:
        o  4381 {   This procedure processes the monitor request to update service class statistics.
        o  4382 { DESIGN:
        o  4383 {   The service class statistics updated by this procedure must be updated in monitor mode
        o  4384 {   in order to synchronize writing the statistics variable.  This procedure is called via
        o  4385 {   a monitor request at statistics emission time.  All initiated jobs are scanned for
        o  4386 {   statistics information.
        o  4387
        o  4388   PROCEDURE [XDCL] jmp$update_serv_class_stats_req
        o  4389     (VAR request_block: jmt$rb_service_class_statistics);
        o  4390
        o  4391     VAR
        o  4392       ijl_bn: jmt$ijl_block_number,
        o  4393       ijl_bi: jmt$ijl_block_index,
        o  4394       ijl_ordinal: jmt$ijl_ordinal,
        o  4395       ijle_p: ^jmt$initiated_job_list_entry;
        o  4396
        o  4397     request_block.status.normal := TRUE;
        4  4398
        4  4399     FOR ijl_bn := LOWERBOUND (jmv$ijl_p.block_p^) TO jmv$ijl_p.max_block_in_use DO
       22  4400       IF (jmv$ijl_p.block_p^ [ijl_bn].index_p <> NIL) THEN
       34  4401         ijl_ordinal.block_number := ijl_bn;
       34  4402         FOR ijl_bi := LOWERVALUE (jmt$ijl_block_index) TO UPPERVALUE (jmt$ijl_block_index) DO
       46  4403           ijl_ordinal.block_index := ijl_bi;
       46  4404           jmp$get_ijle_p (ijl_ordinal, ijle_p);
       46  4405           IF ijle_p^.entry_status <> jmc$ies_entry_free THEN
       72  4406             jmp$update_service_class_stats (ijle_p);
       82  4407           IFEND;
       82  4408         FOREND;
       8A  4409       IFEND;
       8A  4410     FOREND;
       8E  4411
       8E  4412   PROCEND jmp$update_serv_class_stats_req;
        o  4413
```


NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

```
        o  4415
        o  4416 { PURPOSE:
        o  4417 {   The purpose of this procedure is to update the service class statistics for a
        o  4418 {   service class with the information of a specific job.  The service class statistics
        o  4419 {   accumulators for the job are updated.
        o  4420
        o  4421   PROCEDURE [XDCL] jmp$update_service_class_stats
        o  4422     (    ijle_p: ^jmt$initiated_job_list_entry);
        o  4423
        o  4424     VAR
        o  4425       statistics_p: ^jmt$mtr_serv_class_stat_entry;
        o  4426
        o  4427 { Update cp statistics.
        o  4428
        o  4429     tmp$set_lock (jmv$service_class_stats_lock);
       42  4430
       42  4431     statistics_p := ^jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics;
       42  4432     statistics_p^.cp_time.job_mode := statistics_p^.cp_time.job_mode +
       42  4433           (ijle_p^.statistics.cp_time.time_spent_in_job_mode -
       42  4434           ijle_p^.service_class_statistics.cp_time.time_spent_in_job_mode);
       42  4435     ijle_p^.service_class_statistics.cp_time.time_spent_in_job_mode :=
       42  4436           ijle_p^.statistics.cp_time.time_spent_in_job_mode;
       42  4437
       42  4438     statistics_p^.cp_time.monitor_mode := statistics_p^.cp_time.monitor_mode +
       42  4439           (ijle_p^.statistics.cp_time.time_spent_in_mtr_mode -
       42  4440           ijle_p^.service_class_statistics.cp_time.time_spent_in_mtr_mode);
       42  4441
       42  4442     ijle_p^.service_class_statistics.cp_time.time_spent_in_mtr_mode :=
       42  4443           ijle_p^.statistics.cp_time.time_spent_in_mtr_mode;
       42  4444
       42  4445 { Update page fault statistics.
       42  4446
       42  4447     statistics_p^.page_faults.disk := statistics_p^.page_faults.disk +
       42  4448           (ijle_p^.statistics.paging_statistics.page_in_count -
       42  4449           ijle_p^.service_class_statistics.page_faults.disk);
       42  4450     ijle_p^.service_class_statistics.page_faults.disk := ijle_p^.statistics.paging_statistics.page_in_count;
       42  4451
       42  4452     statistics_p^.page_faults.reclaimed := statistics_p^.page_faults.reclaimed +
       42  4453           (ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue -
       42  4454           ijle_p^.service_class_statistics.page_faults.reclaimed);
       42  4455     ijle_p^.service_class_statistics.page_faults.reclaimed :=
       42  4456           ijle_p^.statistics.paging_statistics.pages_reclaimed_from_queue;
       42  4457
       42  4458     statistics_p^.page_faults.assigned := statistics_p^.page_faults.assigned +
       42  4459           (ijle_p^.statistics.paging_statistics.new_pages_assigned -
       42  4460           ijle_p^.service_class_statistics.page_faults.assigned);
       42  4461     ijle_p^.service_class_statistics.page_faults.assigned :=
       42  4462           ijle_p^.statistics.paging_statistics.new_pages_assigned;
       42  4463
       42  4464 { Update swapping statistics.
       42  4465
       42  4466     statistics_p^.swap_stats.long_wait_swaps := statistics_p^.swap_stats.long_wait_swaps +
       42  4467           (ijle_p^.job_scheduler_data.job_swap_counts.long_wait -
       42  4468           ijle_p^.service_class_statistics.swapouts.long_wait);
       42  4469     ijle_p^.service_class_statistics.swapouts.long_wait :=
```

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

```
      42  4470            ijle_p^.job_scheduler_data.job_swap_counts.long_wait;
      42  4471
      42  4472      statistics_p^.swap_stats.job_mode_swaps := statistics_p^.swap_stats.job_mode_swaps +
      FC  4473            (ijle_p^.job_scheduler_data.job_swap_counts.job_mode -
      FC  4474            ijle_p^.service_class_statistics.swapouts.job_mode);
      FC  4475      ijle_p^.service_class_statistics.swapouts.job_mode := ijle_p^.job_scheduler_data.job_swap_counts.job_mode;
      FC  4476
      FC  4477      tmp$clear_lock (jmv$service_class_stats_lock);
     14A  4478
     14A  4479    PROCEND jmp$update_service_class_stats;
       O  4480
       O  4481 MODEND jmm$job_scheduler_monitor_mode;
```

**** I=$05578173AS0102D19890821T183254 L=ZZXXLIST B=LGO DA=NONE LO=R RC=NONE OPT=SCHED EL=F LF=CS612 PAD=0

**** NO DIAGNOSTICS

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
actual	1450	1487	1490/M						
actual	3316	3333	3333/M						
ajl_o	4285	4297/S							
ajl_ordinal	4358	4364/S	4365/S	4366/P					
ajle_p	4289	4297/M	4298	4301	4328/M				
ajlo	4076	4084	4085/S						
assigned	520	4460	4461/M						
assigned	2668	4458/M	4458						
attributes	2558	3282	3288	3380	3398	3585/M	3806	3989	3991
		4002	4307	4312					
b	1704	1711	1712						
b	3427	3456	3456						
b	3556	3575	3575						
b	3726	3734	3734						
b	3764	3775	3775						
b	3872	3886	3886						
b	3911	3919	3919						
b	3939	3976	3976						
b	4047	4058	4058						
b	4112	4124	4124						
b	4141	4171	4171						
b	4421	4429	4429						
bc	1705	1708/M	1709	1713					
bc	3427	3456/M	3456	3456					
bc	3556	3575/M	3575	3575					
bc	3726	3734/M	3734	3734					
bc	3764	3775/M	3775	3775					
bc	3872	3886/M	3886	3886					
bc	3911	3919/M	3919	3919					
bc	3939	3976/M	3976	3976					
bc	4047	4058/M	4058	4058					
bc	4112	4124/M	4124	4124					
bc	4141	4171/M	4171	4171					
bc	4421	4429/M	4429	4429					
block_index	45	1327/S	3381/S	3616/M	3618/S	3740/M	3741/S	3848/S	3879/S
		3917/S	3989/S	4053/S	4119/S	4164/M	4403/M	4404/S	
block_number	44	1327/S	3381/S	3614/M	3618/S	3739/M	3741/S	3848/S	3879/S
		3917/S	3989/S	4053/S	4119/S	4163/M	4237/S	4401/M	4404/S
block_p	1153	1327	3381	3612	3613	3618	3736	3737	3741
		3848	3879	3917	3989	4053	4119	4159	4160
		4162	4237/M	4399	4400	4404			
blocked	2015	3372	3491	3492/M	3989				
calculate_circular_service	3589	3589	3597	3600					
call_job_swapper	4149	4156/M	4186/M	4193					
change_service_class	1937	3582							
changes_pointer	3559	3579/M	3581	3581	3582	3584	3601/M	3601	
check_controls	3445	3445	3450	3452					
check_for_class_switch	3271	3303	4304						
check_swapin_queue	3377	3377	3384	3387					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
check_swapin_queue	3939	3989	3989	3989					
circular_service	3560	3587/M	3594/M	3594	3624	3625			
class	3561	3582/M	3583	3585/S	3587/S	3594/S	3594/S		
class_service_threshold	2576	3283	3284						
classes_changed	3562	3571/M	3583/M	3583	3619				
clear	1318	1508/M	3538/M	3638/M	3752/M	3786/M	3832/M	3892/M	3933/M
		3981/M	4060/M	4127/M	4174/M	4477/M			
condition	1419	1424							
condition	1648	1424/M	3713/M	3898/M					
condition	3680	3713							
condition	3872	3898							
controls	1257	3481/M	3483	3486/M	3503/M	3506/M	3512/M	3513	3521
controls	2009	3372	3491	3492/M	3493/M	3494/M	3989		
controls_defined	1253	3476/M	3478/M						
controls_defined	3433	3442/M	3449/M	3475					
count	1315	1505	1506/M	1506	1715/M	1715	3456/M	3456	3538
		3538/M	3538	3575/M	3575	3638	3638/M	3638	3734/M
		3734	3752	3752/M	3752	3775/M	3775	3786	3786/M
		3786	3832	3832/M	3832	3886/M	3886	3892	3892/M
		3892	3919/M	3919	3933	3933/M	3933	3976/M	3976
		3981	3981/M	3981	4058/M	4058	4060	4060/M	4060
		4124/M	4124	4127	4127/M	4127	4171/M	4171	4174
		4174/M	4174	4429/M	4429	4477	4477/M	4477	
count	4077	4079/M	4086/M	4086	4087	4096			
count_active_jobs	4083	4083	4088	4091					
cp_service_at_class_switch	369	3623	3818/M						
cp_time	512	4434	4435/M	4440	4442/M				
cp_time	557	3621	3622	3819	3820	4433	4436	4439	4443
cp_time	2655	4432/M	4432	4438/M	4438				
cpu_dispatching_allocation	2120	3447	3448	3485	3488	3505	3508	3515	
current_time	3944	3962/M	3965	3972	3979				
defined	2568	3289							
delayed_swapin_work	99	3743	4062	4063/M	4063				
dfc$command_record_bytes	169	177							
dfc$division_overwrite_words	156	184							
dfc$esm_command_record_size	177	185							
dfc$esm_header_record_size	178	185							
dfc$esm_maintenance_buf_size	157	188							
dfc$esm_memory_base_shift	163	185	186	186					
dfc$header_record_bytes	168	178							
dfc$max_esm_memory_size	158	187							
dfc$max_number_of_mainframes	165	150							
dfc$min_data_record_bytes	173	184							
dfc$min_esm_division_size	183	187							
dft$mainframe_set	150	100	101	277	278				
disk	518	4449	4450/M						
disk	2666	4447/M	4447						
dispatching_allocation_interval	2122	3482							
dispatching_control	82	3623	3809/M	3810	3814/M	3816/M	3818/M	4348	
dispatching_control	2584	3585/M	3812	3815	3817				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dispatching_control_changes_p	1933	3579							
dispatching_control_changes_p	1939	3601							
dispatching_control_index	364	3809/M							
dispatching_control_index	3563	3590	3591/S	3592/S	3595/S				
dispatching_control_info	1938	3584							
dispatching_control_p	3564	3584/M	3585	3591	3592	3595			
dispatching_priority	365	3810	3814/M	4348					
dispatching_priority	379	3812	3815						
dispatching_priority	3364	3372/S	3382						
dispatching_priority	3939	3989/S	3989						
dispatching_priority_time	1268	3486/M	3503/M	3506/M	3512/M				
dmt$system_file_id	199	132							
dp	3434	3446	3447/S	3448/S	3484	3485/S	3486/S	3488/S	3490
		3491/S	3492/S	3493/S	3494/S	3495/S	3496/S	3499	3503/S
		3505/S	3506/S	3508/S	3510	3512/S	3515/S	3517	3531
		3532	3535/S						
dp_unblocked	3435	3443/M	3500/M	3540					
enforce_maximum	2154	3515							
enforce_maximums	1256	3480/M	3516/M	3516					
enforce_maximums	1292	3472/M							
entry_status	68	1353	1358/M	3742	3744	3744/M	3777	3792	3881
		3887	3887/M	3895	3963	4006	4006/M	4055	4055
		4056	4059	4059/M	4120	4125	4125/M	4166	4172
		4172/M	4213	4213/M	4236	4236/M	4299	4347	4372
		4372/M	4405						
estimated_ready_time	89	3965	4302/M						
event	3404	3409/S	3409/S	3409/S					
event	3427	3541/S	3541/S	3541/S	3542/S	3542/S	3542/S		
event	3764	3828/S	3828/S	3828/S					
event	3939	3959/S	3959/S	3959/S					
event	4047	4065/S	4065/S	4065/S					
event	4073	4097/S	4097/S	4097/S					
event	4232	4238/S	4238/S	4238/S					
event	4257	4262/S	4263/S	4264/S					
event	4344	4350/S	4350/S	4350/S					
final	1449	1487	1490						
final	3316	3333	3333						
forward_link	947	3849							
forward_link	1214	3845							
gft$file_descriptor_index	214	204							
gft$system_file_identifier	203	199	1016						
gft$table_residence	217	205							
guaranteed_service_quantum	2577	4307	4313						
guaranteed_service_quantum	4290	4315							
guaranteed_service_remaining	119	3998	4001/M	4309/M	4314/M				
i#program_error	1516	1503	3538	3638	3752	3786	3832	3892	3933
		3981	4060	4127	4174	4477			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
id	1316	1502	1709	1713/M	3456	3456/M	3538	3575	3575/M
		3638	3734	3734/M	3752	3775	3775/M	3786	3832
		3886	3886/M	3892	3919	3919/M	3933	3976	3976/M
		3981	4058	4058/M	4060	4124	4124/M	4127	4171
		4171/M	4174	4429	4429/M	4477			
ignore_status	3344	3351/P							
ijl_bi	3566	3615	3616						
ijl_bi	3730	3738	3740						
ijl_bi	4152	4161	4162/S	4164					
ijl_bi	4393	4402	4403						
ijl_bn	3565	3612	3613/S	3614					
ijl_bn	3729	3736	3737/S	3739					
ijl_bn	4151	4159	4160/S	4162/S	4163				
ijl_bn	4392	4399	4400/S	4401					
ijl_integer	2751	3329/P	3331	3332/P	3333/P				
ijl_o	4048	4053/P							
ijl_o	4113	4119/P	4126/P						
ijl_ord	3940	3993/P	4007/P						
ijl_ord	4292	4301/M	4319/P						
ijl_ordinal	24	4301	4365						
ijl_ordinal	1323	1327/S	1327/S						
ijl_ordinal	2416	3689/P	3917/P	3929/P					
ijl_ordinal	2749	3326/M	3330						
ijl_ordinal	3317	3326							
ijl_ordinal	3363	3381/S	3381/S						
ijl_ordinal	3556	3618/S	3618/S						
ijl_ordinal	3567	3614/M	3616/M	3617	3618/P	3631/P			
ijl_ordinal	3726	3741/S	3741/S						
ijl_ordinal	3731	3739/M	3740/M	3741/P	3745/P				
ijl_ordinal	3838	3848/S	3848/S						
ijl_ordinal	3841	3845/M	3847	3847	3848/P	3850/P	3851/M		
ijl_ordinal	3872	3879/S	3879/S						
ijl_ordinal	3873	3879/P	3890/P						
ijl_ordinal	3911	3917/S	3917/S						
ijl_ordinal	3939	3989/S	3989/S						
ijl_ordinal	4047	4053/S	4053/S						
ijl_ordinal	4112	4119/S	4119/S						
ijl_ordinal	4153	4163/M	4164/M	4173/P					
ijl_ordinal	4207	4214/P							
ijl_ordinal	4233	4237/S							
ijl_ordinal	4361	4365/M	4373/P						
ijl_ordinal	4388	4404/S	4404/S						
ijl_ordinal	4394	4401/M	4403/M	4404/P					
ijl_p	4051	4053/P	4055	4055	4056	4059/P	4062	4063/M	4063
ijl_p	4116	4119/P	4120	4125/P	4126/P	4130	4132		
ijle_p	25	4298	4364						
ijle_p	1324	1327/M							
ijle_p	1346	1362	1362						
ijle_p	1346	1366	1366						
ijle_p	1347	1353	1358/M	1362/P	1366/P				
ijle_p	3272	3282/S	3283	3286	3292	3295	3296		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijle_p	3318	3330/M							
ijle_p	3363	3381/M							
ijle_p	3367	3381/P	3382						
ijle_p	3556	3618/M							
ijle_p	3568	3609/M	3618/P	3619	3620	3621	3622	3623	3624/S
		3626/S	3631/P						
ijle_p	3645	3651	3652						
ijle_p	3663	3669	3670						
ijle_p	3726	3741/M							
ijle_p	3726	3744	3744/M	3744/P	3744/P				
ijle_p	3726	3744	3744						
ijle_p	3726	3744	3744						
ijle_p	3732	3741/P	3742	3743	3744/P	3745/P			
ijle_p	3764	3793	3793						
ijle_p	3764	3795	3795						
ijle_p	3770	3773/P	3777	3777	3779	3783	3785	3790/P	3792
		3793/P	3794/M	3795/P	3797/M	3808/M	3809/M	3810	3810
		3811/M	3814/M	3816/M	3818/M	3819	3820	3821/P	
ijle_p	3838	3848/M							
ijle_p	3842	3848/P	3849						
ijle_p	3872	3879/M							
ijle_p	3872	3887	3887/M	3887/P	3887/P				
ijle_p	3872	3887	3887						
ijle_p	3872	3887	3887						
ijle_p	3877	3879/P	3881	3887/P	3888	3889/M	3890/P	3895	
ijle_p	3911	3917/M							
ijle_p	3915	3917/P	3921/S	3922/S	3924/S	3925/S	3926/S	3927/S	3929/P
		3931/P							
ijle_p	3939	3989/P	3989						
ijle_p	3939	3989/M							
ijle_p	3939	4006	4006/M	4006/P	4006/P				
ijle_p	3939	4006	4006						
ijle_p	3939	4006	4006						
ijle_p	3941	3956	3957/M	3963	3965	3965	3967/M	3969/M	3972/M
		3974	3979	3988	3989/P	3990	3998	3999/M	4001/M
		4002/M	4006/P	4007/P					
ijle_p	4047	4053/M							
ijle_p	4047	4059	4059/M	4059/P	4059/P				
ijle_p	4047	4059	4059						
ijle_p	4047	4059	4059						
ijle_p	4112	4119/M							
ijle_p	4112	4125	4125/M	4125/P	4125/P				
ijle_p	4112	4125	4125						
ijle_p	4112	4125	4125						
ijle_p	4141	4172	4172/M	4172/P	4172/P				
ijle_p	4141	4172	4172						
ijle_p	4141	4172	4172						
ijle_p	4150	4162/M	4166	4167	4172/P	4173/P	4180	4181	
ijle_p	4206	4213	4213/M	4213/P	4213/P				
ijle_p	4206	4213	4213						
ijle_p	4206	4213	4213						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijle_p	4208	4213/P	4214/P	4217	4219				
ijle_p	4232	4236	4236/M	4236/P	4236/P				
ijle_p	4232	4236	4236						
ijle_p	4232	4236	4236						
ijle_p	4234	4236/P							
ijle_p	4291	4298/M	4299	4302/M	4302	4303/P	4304/P	4305/M	4307/S
		4309/M	4311	4312/S	4314/M	4315	4317/M	4318	4324
		4325/P							
ijle_p	4345	4347	4348	4348	4349/S				
ijle_p	4357	4372	4372/M	4372/P	4372/P				
ijle_p	4357	4372	4372						
ijle_p	4357	4372	4372						
ijle_p	4362	4364/M	4372/P	4373/P					
ijle_p	4388	4404/M							
ijle_p	4395	4404/P	4405	4406/P					
ijle_p	4422	4431/S	4433	4434	4435/M	4436	4439	4440	4442/M
		4443	4448	4449	4450/M	4450	4453	4454	4455/M
		4456	4459	4460	4461/M	4462	4467	4468	4469/M
		4470	4473	4474	4475/M	4475			
ijlo	3321	3326/M	3332/P						
ijlo	3771	3773/P	3821/P						
in_use	23	4085							
index_p	1166	1327	3381	3613	3618	3737	3741	3848	3879
		3917	3989	4053	4119	4160	4162	4400	4404
initial	1448	1487							
initial	3316	3333							
insert_job_in_ready_task_list	3316	3336	4007	4126	4173	4214	4373		
ioc$no_error	1102	4167							
iot$io_error	1102	133	1056						
jmc$batch	610	3620							
jmc$call_job_swapper	2089	3409/P							
jmc$dc_maximum_service_limit	393	3593							
jmc$dp_conversion	324	3490	3499	3510	3517	3532			
jmc$dsw_job_recovery	266	3743	4062						
jmc$dsw_recovery_swap_io_error	268	4064							
jmc$examine_input_queue	2097	3349/S	3542/P	3987/S					
jmc$examine_swapin_queue	2098	3350/S	3541/P	3958/S	3959/P	4010/S	4012/S	4131/S	4177/S
		4218/S							
jmc$highest_prio_age_interval	2628	2619	2629						
jmc$highest_sched_memory_level	2377	2370							
jmc$highest_service_accumulator	674	675							
jmc$highest_service_factor_valu	2652	2645							
jmc$highest_service_interval	2393	2386							
jmc$highest_working_set_size	2771	2762	2772	2774	2776	2778			
jmc$ies_entry_free	449	3742	4405						
jmc$ies_job_in_memory	452	4299							
jmc$ies_job_in_memory_non_swap	451	3777							
jmc$ies_job_swapped	454	463	3744/P	3887/P	3963	4055			
jmc$ies_job_terminating	450	4236/P							
jmc$ies_operator_force_out	455	3881	4056						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$ies_ready_task	458	4006/P	4125/P	4172/P	4213/P	4372/P			
jmc$ies_swapin_candidate	459	4347							
jmc$ies_swapin_in_progress	453	462	1360	1361	1364	1365	3744	3744	3744
		3744	3887	3887	3887	3887	4006	4006	4006
		4006	4055	4059	4059	4059	4059	4120	4125
		4125	4125	4125	4172	4172	4172	4172	4213
		4213	4213	4213	4236	4236	4236	4236	4372
		4372	4372	4372					
jmc$ies_swapped_in	462	3792							
jmc$ies_system_force_out	456	3895	4059/P	4166					
jmc$iss_idle_tasks_initiated	469	496							
jmc$iss_job_allocate_swap_file	473	4181							
jmc$iss_swapin_io_complete	494	497							
jmc$iss_swapin_requested	490	497							
jmc$iss_swapout_complete	489	496							
jmc$iss_swapped_io_cannot_init	480	507	4180						
jmc$iss_swapped_io_complete	485	3990							
jmc$iss_swapped_no_io	471	506							
jmc$job_terminated	2085	4238/P							
jmc$keyword_offset_maximum	691	2620	2763						
jmc$kjl_maximum_entries	238	231	232	626					
jmc$kol_maximum_entries	248	233							
jmc$max_active_jobs	229	2601	2609	2610					
jmc$max_ajl_ord	230	223	229						
jmc$max_completed_job_count	2073	2066							
jmc$max_dispatching_control	404	408	3590						
jmc$max_dispatching_priority	326	286	289	290	1238				
jmc$max_ijl_entries	52	1215							
jmc$max_ijl_index_count	53	1164							
jmc$max_ijl_ord	231	2701							
jmc$maximum_job_categories	2037	2034	2038						
jmc$maximum_job_classes	604	607							
jmc$maximum_job_count	245	238	2051						
jmc$maximum_output_count	255	248							
jmc$maximum_service_classes	707	710	3254						
jmc$min_dispatching_control	403	407	3590	3809	3812/S	3815/S	3817/S		
jmc$min_dispatching_priority	327	1238							
jmc$min_ecc	1770	1771	1778						
jmc$min_ecc_sch	1778	1779	1781	1783	1785	1787	1789	1791	1793
		1795	1797	1799	1801	1803	1805	1807	1809
		1811	1813	1818	1822	1825	1828	1831	1834
		1837	1840	1843	1846	1850	1853	1857	1860
		1863	1866	1869	1872	1875	1879	1882	1886
		1889	1892	1896	1899	1903	1907	1911	1915
		1918	1921	1924					
jmc$needed_memory_available	2100	4277/S							
jmc$null_service_class	700	701	3781						
jmc$priority_aging_interval_max	2619	2616							
jmc$priority_bias_maximum	2354	2350	2350						
jmc$priority_p1	340	287	1993	3446	3484	3531			
jmc$priority_p10	349	288	1993						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$priority_p14	353	288							
jmc$priority_p8	347	287	3446	3484	3531				
jmc$ready_task_in_job	2084	4008/S	4129/S	4176/S	4216/S	4374/S			
jmc$recovery_swap_io_error	2086	4065/P							
jmc$required_offset	689	2777							
jmc$reserved_ajls	234	229							
jmc$sched_profile_deadstart_id	1755	3219							
jmc$scheduler_wake_time	2099	3348/S							
jmc$scheduling_memory_level_max	2370	2367							
jmc$service_accumulator_maximum	666	663							
jmc$service_factor_value_max	2645	2642							
jmc$service_interval_maximum	2386	2383							
jmc$sr_idle_dispatching	720	4306	4310	4366/P					
jmc$sr_thrashing	715	3988							
jmc$src_change_dispatching_ctrl	2402	2425	3697						
jmc$src_class_switch	2401	2419	3291	3694					
jmc$src_cleanup_unrecovered_job	2402	2415	3700						
jmc$src_dispatching_allocation	2403	2429	3706						
jmc$src_idling_advance_swaps	2401	2417	3691						
jmc$src_operator_swap_in	2400	2415	3688						
jmc$src_process_damaged_jobs	2404	2433							
jmc$src_sched_profile_loading	2403	2427	3703						
jmc$src_swapin_recovered_jobs	2404	2431	3709						
jmc$ssn_counter_size	2731	2728							
jmc$ssn_model_number_size	2737	2734							
jmc$ssn_sequence_number_size	2696	2693							
jmc$ssn_serial_number_size	2743	2740							
jmc$subsystem_priority_change	2092	4350/P							
jmc$swap_jobs_for_lower_maxaj	2093	3828/P							
jmc$system_default_offset	690	691	2779						
jmc$system_is_thrashing	2095	4097/P							
jmc$system_service_class	702	3378	3989						
jmc$system_supplied_name_size	931	928	1370						
jmc$unlimited_offset	687	676	2630	2773					
jmc$unlimited_service_accum	675	3284	4308	4309					
jmc$unspecified_offset	688	2775							
jmc$working_set_size_maximum	2762	2759							
jme$invalid_scheduler_request	1915	3713/P							
jme$job_dead_cannot_swap	1801	3898/P							
jmk$base	3137	2788	2792	2796	2800	2804	2808	2812	2816
		2820	2824	2828	2832	2836	2840	2844	2848
		2852	2856	2860	2864	2868	2872	2876	2880
		2884	2888	2892	2896	2900	2904	2908	2912
		2916	2920	2924	2928	2932	2936	2940	2944
		2948	2952	2956	2960	2964	2968	2972	2976
		2980	2984	2988	2992	2996	3000	3004	3008
		3012	3016						
jmk$ready_task_in_swapped_job	2884	3950	4019						
jmk$set_swapout_candidate	2836	4295	4332						
jmp$activate_job_mode_swapper	3404	3411	4194						
jmp$calculate_service	1334	4303							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmp$change_dispatching_alloc	3427	3545	3707						
jmp$change_dispatching_mtr_req	3556	3640	3698						
jmp$change_ijl_entry_status	1346	1369	3744	3887	4006	4059	4125	4172	4213
		4236	4372						
jmp$decrement_swapped_job_count	3644	1366	3658	3744	3793	3887	4006	4059	4125
		4172	4213	4236	4372				
jmp$find_jsn	1370	3773							
jmp$get_ijle_p	1323	1329	3381	3618	3741	3848	3879	3917	3989
		4053	4119	4404					
jmp$increment_swapped_job_count	3662	1362	3676	3744	3795	3887	4006	4059	4125
		4172	4213	4236	4372				
jmp$mtr_job_scheduler_requests	3680	3716							
jmp$mtr_swapin_recovered_jobs	3726	3710	3754						
jmp$process_class_switch	3764	3298	3695	3834					
jmp$process_idling_adv_swaps	3838	3692	3854						
jmp$process_oper_swapin_mtr_req	3872	3689	3901						
jmp$process_unrecovered_job	3911	3701	3935						
jmp$ready_task_in_swapped_job	3939	3745	3890	4021					
jmp$recognize_job_dead	4047	4069							
jmp$recognize_thrashing	4073	4103							
jmp$reset_job_to_swapped_out	4112	4137							
jmp$resurrect_dead_jobs	4141	4197							
jmp$set_entry_status_to_rt	4206	4224							
jmp$set_job_terminated	4232	4240							
jmp$set_sched_profile_loading	4248	3704	4252						
jmp$set_scheduler_event	4256	3409	3541	3542	3828	3959	4065	4097	4238
		4268	4350						
jmp$set_scheduler_memory_event	4272	4280							
jmp$set_swapout_candidate	4284	4334	4366						
jmp$subsystem_priority_change	4344	4353							
jmp$swap_non_dispatchable_job	4357	4376							
jmp$update_serv_class_stats_req	4388	4412							
jmp$update_service_class_stats	4421	3790	4406	4479					
jmt$active_job_list	30	8							
jmt$active_job_list_entry	22	30	4289						
jmt$ajl_ordinal	223	69	1177	1194	1735	4076	4077	4285	4358
jmt$change_dispatching_list	1931	3199	3199						
jmt$completed_job_count_range	2066	2060							
jmt$cpu_dispatching_allocation	2148	2120							
jmt$delayed_swapin_work	270	99	274	4064					
jmt$dispatching_allocation	2151	2149							
jmt$dispatching_control	374	1938	2584	3564					
jmt$dispatching_control_changes	1936	1933	1939	3559					
jmt$dispatching_control_index	407	364	374	3563					
jmt$dispatching_controls	377	375							
jmt$dispatching_interval	2159	2122							
jmt$dispatching_priority	286	81	365	366	367	379	1244	1262	2012
		3364							
jmt$dispatching_priority_set	1262	1254	1255	1256	1292	1293	1294	1295	2007
		2008	3470	3471	3472	3473	3479	3480	3490
		3499	3510	3517	3532				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmt$dual_state_priority	2001	1997						
jmt$dual_state_priority_control	1993	2123						
jmt$dual_state_priority_entry	1996	1994						
jmt$dual_state_subpriority	2002	1998						
jmt$idle_dispatch_controls	2012	2009						
jmt$idle_dispatching_controls	2006	3201						
jmt$idle_dispatching_entry	2014	2012						
jmt$idle_dispatching_queue_time	2164	2124						
jmt$ijl_block_index	49	45	1166	3566	3615	3615	3730	3738	3738
		4152	4161	4161	4393	4402	4402		
jmt$ijl_block_number	48	44	1154	1155	3565	3729	4151	4392	
jmt$ijl_dispatching_control	363	82							
jmt$ijl_entry_status	449	68	1348	1351	2025	2025			
jmt$ijl_entry_status_statistics	2025	3203							
jmt$ijl_ordinal	43	24	88	116	946	947	1009	1048	1199
		1213	1214	1323	1372	1376	1380	1384	1389
		1694	1725	1736	2416	2700	2749	3241	3241
		3317	3567	3731	3771	3841	3843	3873	3940
		4048	4113	4153	4207	4233	4292	4361	4394
jmt$ijl_p	1152	1147							
jmt$ijl_page_fault_count	523	518	519	520					
jmt$ijl_page_stats	517	513							
jmt$ijl_service_class_stats	511	103							
jmt$ijl_statistics	556	102							
jmt$ijl_swap_count	532	528	529						
jmt$ijl_swap_counts	527	122	514						
jmt$ijl_swap_status	467	71	72	73					
jmt$initiated_job_list_block	1163	1169							
jmt$initiated_job_list_entry	65	25	973	1166	1324	1335	1347	1371	1390
		1585	1693	1724	1737	3272	3318	3367	3568
		3645	3663	3732	3770	3842	3877	3915	3941
		4051	4116	4150	4208	4234	4291	4345	4362
		4395	4422						
jmt$initiated_job_list_p	1169	1153							
jmt$input_file_location	646	641							
jmt$job_abort_disposition	655	639							
jmt$job_category	2034	2030							
jmt$job_category_set	2030	2125	2126	2189	2190				
jmt$job_class	607	127	2055	3648	3666				
jmt$job_class_count	2056	2055							
jmt$job_class_counts	2055	2045							
jmt$job_count_range	2051	2042	2043	2044	2057	2058	2059	2078	2079
jmt$job_counts	2041	3210							
jmt$job_mode	610	84							
jmt$job_priority	615	124	125	2593	2594	2595	2596		
jmt$job_recovery_disposition	658	640							
jmt$job_sched_event_selections	2105	3214	3214						
jmt$job_scheduler_event	2103	3212	3212						
jmt$job_scheduler_events	2083	2103	2105	4257					
jmt$job_scheduler_table	2108	3218	3218						
jmt$kjl_index	626	70							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jmt$mainframe_categories	2183	2131						
jmt$mainframe_entry	2186	2183						
jmt$maximum_active_jobs	2601	2578						
jmt$mtr_serv_class_stat_entry	2654	2559	4425					
jmt$priority_aging_interval	2616	2586						
jmt$priority_bias	2350	2135						
jmt$queue_file_ijl_information	638	109						
jmt$rb_sched_sub_reqcodes	2400	2414						
jmt$rb_scheduler_requests	2411	3276	3681	3765	3912			
jmt$rb_service_class_statistics	2550	4389						
jmt$sc_cp_stat	2682	2661	2662					
jmt$sc_pf_stat	2683	2666	2667	2668				
jmt$sc_swap_count	2685	2672	2673	2677	2679			
jmt$sc_swap_stat	2684	2674	2675	2676	2678			
jmt$scheduling_data	115	93						
jmt$scheduling_memory_level	2367	2142	2143					
jmt$scheduling_memory_levels	2141	2127						
jmt$scheduling_priority	2592	2585						
jmt$service_accumulator	663	117	118	119	2422	2424	2576	2577	4290
jmt$service_class_attributes	2564	2558	3277	3769					
jmt$service_class_count	2077	2076							
jmt$service_class_counts	2076	2046							
jmt$service_class_cp_time	2660	2655							
jmt$service_class_entry	2557	3254							
jmt$service_class_index	710	128	1937	2076	2421	2423	2569	2579	2689
		3229	3253	3258	3260	3275	3341	3368	3394
		3560	3561	3649	3667	3768	3945		
jmt$service_class_name	2634	2571	2572						
jmt$service_class_page_faults	2665	2656							
jmt$service_class_set	2689	3195	3195	3195	3197	3197	3197	3346	3562
		3571	3583						
jmt$service_class_swap_stats	2671	2657	3947						
jmt$service_factor_value	2642	2580							
jmt$service_factors	2638	2580							
jmt$service_interval	2383	2114	2116						
jmt$ssn_counter	2728	2722							
jmt$ssn_model_number	2734	2716							
jmt$ssn_sequence_number	2693	2720	3256						
jmt$ssn_serial_number	2740	2718							
jmt$swap_data	131	95							
jmt$swapin_candidate_q_header	2699	3261							
jmt$swapout_reasons	713	123	4286						
jmt$swapped_job_entry	728	140	974						
jmt$system_supplied_name	928	66	2420	2713					
jmt$system_supplied_name_mask	2710	3265							
jmt$task_time_slice	417	397	398						
jmt$time_slice_values	396	381							
jmt$trick_ijlo_variant_record	2746	3321	3322						
jmt$user_dispatching_priority	287	1271	2148	3434					
jmt$working_set_size	2759	3227	3227						
jmv$ajl_p	8	4085	4297	4364	4365				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmv$change_dispatching_list	3199	3579							
jmv$classes_in_maxaj_limit_wait	3195	3346/M	3346	4009	4130	4217	4324		
jmv$classes_in_resource_wait	3197	4011	4132	4219					
jmv$idle_dispatching_controls	3201	3372	3491	3492/M	3493/M	3494/M	3497/M	3498	3989
jmv$ijl_entry_status_statistics	3203	1355/M	1356	3744/M	3744	3887/M	3887	4006/M	4006
		4059/M	4059	4125/M	4125	4172/M	4172	4213/M	4213
		4236/M	4236	4372/M	4372				
jmv$ijl_p	1147	1327	3381	3612	3612	3613	3618	3736	3736
		3737	3741	3848	3879	3917	3989	4053	4119
		4159	4159	4160	4162	4237/M	4399	4399	4400
		4404							
jmv$ijl_ready_task_list	3208	3329/P	3332/P						
jmv$job_counts	3210	1362/M	1362	1362/M	1362	1366/M	1366	1366/M	1366
		3380	3380	3396	3397	3653/M	3654	3655/M	3655
		3671/M	3672	3673/M	3673	3744/M	3744	3744/M	3744
		3744/M	3744	3744/M	3744	3793/M	3793	3793/M	3793
		3795/M	3795	3795/M	3795	3800/M	3801	3803/M	3804
		3825	3826	3887/M	3887	3887/M	3887	3887/M	3887
		3887/M	3887	3921/M	3922	3924/M	3925	3926/M	3927
		3989	3989	3991	3991	4006/M	4006	4006/M	4006
		4006/M	4006	4006/M	4006	4059/M	4059	4059/M	4059
		4059/M	4059	4059/M	4059	4125/M	4125	4125/M	4125
		4125/M	4125	4125/M	4125	4172/M	4172	4172/M	4172
		4172/M	4172	4172/M	4172	4213/M	4213	4213/M	4213
		4213/M	4213	4213/M	4213	4236/M	4236	4236/M	4236
		4236/M	4236	4236/M	4236	4372/M	4372	4372/M	4372
		4372/M	4372	4372/M	4372				
jmv$job_sched_events_selected	3214	3409	3541	3542	3828	3958/M	3959	4012/M	4065
		4097	4238	4264	4350				
jmv$job_scheduler_event	3212	3348/M	3349/M	3350/M	3409	3409/M	3541	3541/M	3542
		3542/M	3828	3828/M	3959	3959/M	3987	4008/M	4010/M
		4065	4065/M	4097	4097/M	4129/M	4131/M	4176/M	4177/M
		4216/M	4218/M	4238	4238/M	4262	4263/M	4277/M	4350
		4350/M	4374/M						
jmv$job_scheduler_table	3218	3447	3448	3482	3485	3487	3505	3507	3515
jmv$max_ajl_ordinal_in_use	1177	4084							
jmv$max_service_class_in_use	3229	3378	3989						
jmv$max_think_time	3235	3966	3967						
jmv$min_think_time	3231	3968							
jmv$null_ijl_ordinal	3241	3379	3847	3847	3989				
jmv$number_free_ajl_entries	1184	3990							
jmv$prevent_activation_of_jobs	3243	3987							
jmv$sched_profile_is_loading	3247	3285	3778	4250/M					
jmv$service_class_stats_lock	3251	3976/P	3981/P	4429/P	4477/P				
jmv$service_classes	3253	3282	3288	3288	3380	3398	3585/M	3806	3977
		3989	3991	4002	4306	4312	4431		
jmv$subsystem_priority_changes	3258	4349/M							
jmv$swap_jobs_in_long_wait	1189	4300							
jmv$swapin_candidate_queue	3260	3379	3381/P	3989	3989/P				
jmv$swapped_idle_disp_count	3263	4371/M	4371						
jmv$system_ajl_ordinal	1194	4084							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmv$system_ijl_ordinal	1199	3617							
job_class	127	1362	1366	3651	3669	3744	3744	3793	3795
		3887	3887	3926/S	3927/S	4006	4006	4059	4059
		4125	4125	4172	4172	4213	4213	4236	4236
		4372	4372						
job_class	1346	1362/M	1362/S	1362/S					
job_class	1346	1366/M	1366/S	1366/S					
job_class	3648	3651/M	3653/S	3654/S					
job_class	3666	3669/M	3671/S	3672/S					
job_class	3726	3744/M	3744/S	3744/S					
job_class	3726	3744/M	3744/S	3744/S					
job_class	3764	3793/M	3793/S	3793/S					
job_class	3764	3795/M	3795/S	3795/S					
job_class	3872	3887/M	3887/S	3887/S					
job_class	3872	3887/M	3887/S	3887/S					
job_class	3939	4006/M	4006/S	4006/S					
job_class	3939	4006/M	4006/S	4006/S					
job_class	4047	4059/M	4059/S	4059/S					
job_class	4047	4059/M	4059/S	4059/S					
job_class	4112	4125/M	4125/S	4125/S					
job_class	4112	4125/M	4125/S	4125/S					
job_class	4141	4172/M	4172/S	4172/S					
job_class	4141	4172/M	4172/S	4172/S					
job_class	4206	4213/M	4213/S	4213/S					
job_class	4206	4213/M	4213/S	4213/S					
job_class	4232	4236/M	4236/S	4236/S					
job_class	4232	4236/M	4236/S	4236/S					
job_class	4357	4372/M	4372/S	4372/S					
job_class	4357	4372/M	4372/S	4372/S					
job_class_counts	2045	1362/M	1362	1366/M	1366	3653/M	3654	3671/M	3672
		3744/M	3744	3744/M	3744	3793/M	3793	3795/M	3795
		3887/M	3887	3887/M	3887	3926/M	3927	4006/M	4006
		4006/M	4006	4059/M	4059	4059/M	4059	4125/M	4125
		4125/M	4125	4172/M	4172	4172/M	4172	4213/M	4213
		4213/M	4213	4236/M	4236	4236/M	4236	4372/M	4372
		4372/M	4372						
job_is_good_swap_candidate	26	4328/M							
job_mode	84	3620							
job_mode	529	4473	4474	4475/M	4475				
job_mode	2661	4432/M	4432						
job_mode_swaps	2673	4472/M	4472						
job_scheduler_data	93	1362	1362	1366	1366	3282/S	3283	3286	3295
		3296	3330/M	3619	3624/S	3626/S	3651	3652	3669
		3670	3744	3744	3744	3744	3779	3783	3785
		3793	3793	3794/M	3795	3795	3797/M	3808/M	3887
		3887	3887	3887	3889/M	3921/S	3922/S	3924/S	3925/S
		3926/S	3927/S	3974	3988	3998	3999/M	4001/M	4002/M
		4006	4006	4006	4006	4059	4059	4059	4059
		4125	4125	4125	4125	4130	4132	4172	4172
		4172	4172	4213	4213	4213	4213	4217	4219
		4236	4236	4236	4236	4305/M	4307/S	4309/M	4311

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		4312/S	4314/M	4315	4317/M	4318	4324	4325/P	4349/S
		4372	4372	4372	4372	4431/S	4467	4470	4473
		4475							
job_swap_counts	122	4317/M	4318	4467	4470	4473	4475		
jsc$isqi_null	950	3928/P							
jsc$isqi_swapped_io_completed	951	953	3845/S						
jsc$isqi_swapped_io_not_init	950	953							
jsp$monitor_advance_swap	1376	3850							
jsp$monitor_swap_in	1380	3993							
jsp$monitor_swap_out	1384	4319							
jsp$relink_swap_queue	1389	3929							
jst$changed_asid_entry	996	987							
jst$ijl_swap_queue_id	950	945	1220	1391					
jst$ijl_swap_queue_link	944	77							
jst$ijl_swap_queue_list	1220	1204							
jst$ijl_swap_queue_list_entry	1212	1220							
jst$io_control_information	958	96							
jst$swap_file_descriptor	972	97							
jst$swapped_page_descriptor	981	979							
jst$swapped_page_descriptors	978	975							
jsv$ijl_swap_queue_list	1204	3845							
last_cp_time	2018	3494/M							
last_think_time	90	3965	3967/M	3969/M	4302				
list_head	3322	3329/P	3330	3331	3333/P				
local_set	3436	3529/M	3532/M	3533/M	3533	3533	3534/M	3534	3534
		3535							
lock	1499	1502	1505	1506/M	1506	1508/M			
lock	1701	1709	1711	1713/M	1715/M	1715			
lock	3427	3456	3456	3456/M	3456/M	3456			
lock	3427	3538	3538	3538/M	3538	3538/M			
lock	3556	3575	3575	3575/M	3575/M	3575			
lock	3556	3638	3638	3638/M	3638	3638/M			
lock	3726	3734	3734	3734/M	3734/M	3734			
lock	3726	3752	3752	3752/M	3752	3752/M			
lock	3764	3775	3775	3775/M	3775/M	3775			
lock	3764	3786	3786	3786/M	3786	3786/M	3832	3832	3832/M
		3832	3832/M						
lock	3872	3886	3886	3886/M	3886/M	3886			
lock	3872	3892	3892	3892/M	3892	3892/M			
lock	3911	3919	3919	3919/M	3919/M	3919			
lock	3911	3933	3933	3933/M	3933	3933/M			
lock	3939	3976	3976	3976/M	3976/M	3976			
lock	3939	3981	3981	3981/M	3981	3981/M			
lock	4047	4058	4058	4058/M	4058/M	4058			
lock	4047	4060	4060	4060/M	4060	4060/M			
lock	4112	4124	4124	4124/M	4124/M	4124			
lock	4112	4127	4127	4127/M	4127	4127/M			
lock	4141	4171	4171	4171/M	4171/M	4171			
lock	4141	4174	4174	4174/M	4174	4174/M			
lock	4421	4429	4429	4429/M	4429/M	4429			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	4421	4477	4477	4477/M	4477	4477/M			
locked	1314	1711	3456	3575	3734	3775	3886	3919	3976
		4058	4124	4171	4429				
long_wait	528	4317/M	4318	4467	4468	4469/M	4470		
long_wait_swaps	2672	4466/M	4466						
max_block_in_use	1154	3612	3736	4159	4399				
maximum	2153	3448	3505	3508					
maximum	2594	4003							
maximum_active_jobs	2578	3380	3398	3827	3989	3991			
maximum_time	1275	3507/M	3512/M						
maximums_defined	1255	3479/M	3509/M	3509					
maximums_exceeded	1295	3471/M							
memory_reserve_request	87	3956	3957/M						
minimum	2152	3447	3485	3488					
minimum_time	1274	3487/M	3503/M						
minimums_to_satisfy	1254	3473/M	3489/M	3489	3520				
minimums_to_satisfy	1293	3470/M	3520/M	3533/M	3533	3534			
mmc$pq_avail	745	791							
mmc$pq_free	744	803							
mmc$pq_job_fixed	785	792	804						
mmc$pq_job_working_set	787	804	805						
mmc$pq_shared_first_site	795	799							
mmc$pq_shared_num_sites	796	799							
mmc$pq_shared_other	754	794							
mmc$pq_shared_site_01	756	795							
mmc$pq_shared_site_25	780	800							
mmc$pq_shared_task_service	749	793							
mmc$pq_swapped_io_error	783	803							
mmc$pq_wired	747	790							
mmp$nudge_periodic_call	1396	1401	4100						
mmt$active_segment_table_entry	1006	984	1022	1055					
mmt$ast_index	1038	139	999						
mmt$global_page_queue_index	803	1140							
mmt$global_page_queue_list_ent	1130	1140							
mmt$job_page_queue_index	804	730	1141						
mmt$job_page_queue_list	1141	94							
mmt$link	1029	1007	1045	1046	1127				
mmt$locked_page	1067	1051							
mmt$memory_reserve_request	1108	87							
mmt$page_age	1074	1054	1078	1078					
mmt$page_frame_index	967	959	961	962	963	1031	1031	1110	1111
		3239							
mmt$page_frame_queue_id	805	960	1015	1049					
mmt$page_frame_table_entry	1044	982	1060						
mmt$page_queue_list_entry	1126	1131	1141						
mmv$reduce_jws_for_thrashing	1229	4099/M							
mmv$time_to_call_mem_mgr	1407	1398/M	4100/M						
monitor_mode	2662	4438/M	4438						
mtp$error_stop	1416	3896	4121						
mtp$set_status_abnormal	1418	1425	3713	3898					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
new_class	3275	3287/M	3288/S	3288/S	3293				
new_entry_status	1348	1355/S	1356/S	1358	1361	1365			
new_entry_status	3726	3744/S	3744/S	3744	3744	3744			
new_entry_status	3872	3887/S	3887/S	3887	3887	3887			
new_entry_status	3939	4006/S	4006/S	4006	4006	4006			
new_entry_status	4047	4059/S	4059/S	4059	4059	4059			
new_entry_status	4112	4125/S	4125/S	4125	4125	4125			
new_entry_status	4141	4172/S	4172/S	4172	4172	4172			
new_entry_status	4206	4213/S	4213/S	4213	4213	4213			
new_entry_status	4232	4236/S	4236/S	4236	4236	4236			
new_entry_status	4357	4372/S	4372/S	4372	4372	4372			
new_pages_assigned	583	4459	4462						
new_service_accumulator	2422	3294/M							
new_service_class	2421	3293/M	3794	3797	3803/S	3804/S	3806/S	3825/S	3826/S
next_ijl_ordinal	3843	3849/M	3851						
next_service_class_index	2579	3286	3287						
normal	1647	1423/M	3683/M	3713/M	3885/M	3898/M	4397/M		
normalized_interval	3437	3483/M	3487	3507					
old_class	3768	3779/M	3782	3784	3800/S	3801/S			
old_entry_status	1351	1353/M	1355/S	1356/S	1360	1364			
old_entry_status	3726	3744/M	3744/S	3744/S	3744	3744			
old_entry_status	3872	3887/M	3887/S	3887/S	3887	3887			
old_entry_status	3939	4006/M	4006/S	4006/S	4006	4006			
old_entry_status	4047	4059/M	4059/S	4059/S	4059	4059			
old_entry_status	4112	4125/M	4125/S	4125/S	4125	4125			
old_entry_status	4141	4172/M	4172/S	4172/S	4172	4172			
old_entry_status	4206	4213/M	4213/S	4213/S	4213	4213			
old_entry_status	4232	4236/M	4236/S	4236/S	4236	4236			
old_entry_status	4357	4372/M	4372/S	4372/S	4372	4372			
old_list_head	3323	3331/M	3332/P						
old_service_accumulator	2424	3296/M	3783/M	3784					
old_service_class	2423	3295/M	3781	3782/M	3784				
osc$free_running_clock_maximum	439	436	2179						
osc$invalid_ring	860	900							
osc$max_name_size	1759	1763	1766						
osc$max_number_of_processors	1524	1519							
osc$max_page_frames	810	134	135	729	731	967	1008	1128	1134
osc$max_page_size	2276	2272							
osc$max_page_table_entries	811	814							
osc$max_ring	859	900	901						
osc$max_segment_length	883	906							
osc$max_status_condition_code	1613	1609	1625						
osc$max_status_condition_number	1428	1419							
osc$max_string_size	1629	1632	1635	1640					
osc$max_tasks	1100	1097							
osc$maximum_offset	882	883	903	903	904				
osc$maximum_processors	1528	1524							
osc$maximum_segment	881	902							
osc$min_page_size	2275	2272							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$min_ring	858	901							
osc$null_name	1760	1754							
osc$pr_base_constant	1546	1502	1708	3456	3538	3575	3638	3734	3752
		3775	3786	3832	3886	3892	3919	3933	3976
		3981	4058	4060	4124	4127	4171	4174	4429
		4477							
osc$task_time_slice_maximum	428	431							
osk$base	3119	3021	3025	3029	3033	3037	3041	3045	3049
		3053	3057	3061	3065	3069	3073	3078	3081
		3084							
osk$entry	3151	3950	4295						
osk$exit	3152	4019	4332						
osk$system_class	3165	3149	3150	3151	3152	3153	3154	3155	
osp$fetch_locked_variable	1431	1445	3329						
osp$set_locked_variable	1447	1493	3332						
ost$asid	846	78	842	986	997	998	1013		
ost$cp_time	544	512	557	1239					
ost$cp_time_value	542	120	545	546					
ost$free_running_clock	436	27	89	90	91	92	126	136	137
		138	368	380	1012	1267	1274	1275	2017
		2018	2164	3223	3223	3249	3267		
ost$global_task_id	1091	83	112						
ost$key_lock_value	895	892							
ost$name	1766	2115	2570	2634					
ost$page_id	816	826							
ost$page_size	2272	2253							
ost$page_table_entry	821	830	983						
ost$page_table_index	814	830	1052						
ost$paging_statistics	580	558							
ost$processor_model_number	2202	2196							
ost$processor_serial_number	2280	2197							
ost$ring	900	912							
ost$segment	902	913							
ost$segment_offset	903	843	914						
ost$signature_lock	1948	1932							
ost$status_condition	1621	1648							
ost$status_condition_code	1625	1600	1621						
ost$string	1638	1601							
ost$string_size	1632	1639							
ost$system_virtual_address	841	1057							
ost$task_index	1097	1085	1086	1092	1739				
ost$task_time_slice	431	417	2112						
osv$cpus_logically_on	1519	1501	1707	3456	3538	3575	3638	3734	3752
		3775	3786	3832	3886	3892	3919	3933	3976
		3981	4058	4060	4124	4127	4171	4174	4429
		4477							
osv$time_to_check_asyn	1409	1399/M	4100/M						
page_faults	513	4449	4450/M	4454	4455/M	4460	4461/M		
page_faults	2656	4447/M	4447	4452/M	4452	4458/M	4458		
page_in_count	581	4448	4450						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pages_reclaimed_from_queue	582	4453	4456						
paging_statistics	558	4448	4450	4453	4456	4459	4462		
pmc$mainframe_id_size	2289	2286							
pmc$processor_model_number_size	2297	2289	2294						
pmc$processor_serial_num_size	2347	2290	2344						
pmt$binary_mainframe_id	2195	2188							
pmt$cpu_model_number	2262	2251	2258						
pmt$cpu_serial_number	2265	2252	2257						
pmt$mainframe_id	2286	2187							
priority	124	4002/M							
rb	3276	3290/M	3291/M	3292/M	3293/M	3294/M	3295/M	3296/M	3298/P
rb	3765	3773/P	3781	3782/M	3783/M	3784	3784	3794	3797
		3803/S	3804/S	3806/S	3825/S	3826/S			
rb	3912	3917/P	3929/P						
ready_task_count	561	3888							
ready_task_link	116	3330/M							
ready_tasks	1294	3532/M	3533	3534/M	3534				
reclaimed	519	4454	4455/M						
reclaimed	2667	4452/M	4452						
remove_class_from_maxaj_limit	3340	3353	4325						
reqcode	2412	3290/M							
request_block	3681	3683/M	3687	3689/P	3689/P	3695/P	3701/P	3713/P	
request_block	4389	4397/M							
requested_page_count	1110	3956	3957/M						
result	1437	1440	1441	1442					
result	1483	1487	1488	1489					
result	3316	3329	3329	3329					
result	3316	3333	3333	3333					
scan_ijl	3611	3611	3636						
scheduler_initiated_jobs	2078	3380	3397	3800/M	3801	3803/M	3804	3825	3921/M
		3923	3989	3991					
scheduling_dispatching_priority	81	3382	3810	3811/M	3989/P	3989	4348		
scheduling_priority	2585	4003							
search_ijl	4158	4158	4191						
service_accumulator	117	3283	3296	3783	3785	3808/M			
service_accumulator_since_swap	118	3999/M	4311	4315					
service_class	128	1362	1366	3282/S	3286	3295	3619	3624/S	3626/S
		3652	3670	3744	3744	3779	3793	3794/M	3795
		3797/M	3887	3887	3921/S	3922/S	3924/S	3925/S	3974
		4006	4006	4059	4059	4125	4125	4130	4132
		4172	4172	4213	4213	4217	4219	4236	4236
		4307/S	4312/S	4324	4325/P	4349/S	4372	4372	4431/S
service_class	1346	1362/M	1362/S	1362/S					
service_class	1346	1366/M	1366/S	1366/S					
service_class	3341	3347							
service_class	3363	3380/S	3380/S	3380/S					
service_class	3368	3378	3379/S	3380/P	3381/S				
service_class	3394	3396/S	3397/S	3398/S					
service_class	3649	3652/M	3655/S	3656/S					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES							
service_class	3667	3670/M	3673/S	3674/S					
service_class	3726	3744/M	3744/S	3744/S					
service_class	3726	3744/M	3744/S	3744/S					
service_class	3764	3793/M	3793/S	3793/S					
service_class	3764	3795/M	3795/S	3795/S					
service_class	3872	3887/M	3887/S	3887/S					
service_class	3872	3887/M	3887/S	3887/S					
service_class	3939	3989	3989/S	3989/P	3989/S				
service_class	3939	3989/S	3989/S	3989/S	3991/S	3991/S	3991/S		
service_class	3939	4006/M	4006/S	4006/S					
service_class	3939	4006/M	4006/S	4006/S					
service_class	3945	3974/M	3977/S	3991/P	4002/S	4009	4011		
service_class	4047	4059/M	4059/S	4059/S					
service_class	4047	4059/M	4059/S	4059/S					
service_class	4112	4125/M	4125/S	4125/S					
service_class	4112	4125/M	4125/S	4125/S					
service_class	4141	4172/M	4172/S	4172/S					
service_class	4141	4172/M	4172/S	4172/S					
service_class	4206	4213/M	4213/S	4213/S					
service_class	4206	4213/M	4213/S	4213/S					
service_class	4232	4236/M	4236/S	4236/S					
service_class	4232	4236/M	4236/S	4236/S					
service_class	4357	4372/M	4372/S	4372/S					
service_class	4357	4372/M	4372/S	4372/S					
service_class_counts	2046	1362/M	1362	1366/M	1366	3380	3380	3396	3397
		3655/M	3656	3673/M	3674	3744/M	3744	3744/M	3744
		3793/M	3793	3795/M	3795	3800/M	3801	3803/M	3804
		3825	3826	3887/M	3887	3887/M	3887	3921/M	3922
		3924/M	3925	3989	3989	3991	3991	4006/M	4006
		4006/M	4006	4059/M	4059	4059/M	4059	4125/M	4125
		4125/M	4125	4172/M	4172	4172/M	4172	4213/M	4213
		4213/M	4213	4236/M	4236	4236/M	4236	4372/M	4372
		4372/M	4372						
service_class_p	3277	3282/M	3283	3284	3286	3287			
service_class_p	3769	3806/M	3811	3814	3816	3827			
service_class_statistics	103	4434	4435/M	4440	4442/M	4449	4450/M	4454	4455/M
		4460	4461/M	4468	4469/M	4474	4475/M		
service_limit	380	3592	3595	3817					
service_remaining	368	3816/M							
service_used	3569	3621/M	3625/M	3625	3629/M	3631/P			
service_used	4293	4303/P							
set_defined	378	3591							
sft$counter	590	559	560						
statistics	102	3621	3622	3819	3820	3888	4433	4436	4439
		4443	4448	4450	4453	4456	4459	4462	
statistics	2559	3977	4431						
statistics_p	4425	4431/M	4432/M	4432	4438/M	4438	4447/M	4447	4452/M
		4452	4458/M	4458	4466/M	4466	4472/M	4472	
status	1420	1423/M	1424/M						
status	2413	3683/M	3689/P	3713/P					
status	2552	4397/M							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES						
status	3404	3409/P						
status	3427	3541/P	3542/P					
status	3680	3713/M	3713/M					
status	3764	3828/P						
status	3872	3898/M	3898/M					
status	3874	3885/M	3898/P					
status	3939	3959/P						
status	3946	4013/P						
status	4047	4065/P						
status	4073	4097/P						
status	4117	4133/P						
status	4154	4178/P						
status	4211	4220/P						
status	4232	4238/P						
status	4260	4265/P						
status	4275	4278/P						
status	4344	4350/P						
sub_reqcode	2414	3291/M	3687					
succeeded	1451	1485/M	1491/M					
succeeded	3316	3333/M	3333/M					
succeeded	3324	3333/P	3334					
swap_data	95	3972/M	3979	4167				
swap_queue_link	77	3849						
swap_stats	2657	3977	4466/M	4466	4472/M	4472		
swap_stats_p	3947	3977/M	3978/M	3978	3980/M	3980		
swap_status	71	3990	4180	4181				
swap_to_ready_count	2679	3980/M	3980					
swap_to_ready_time	2678	3978/M	3978					
swapin_candidate_queue	2700	3379	3381/P	3989	3989/P			
swapin_q_priority_timestamp	126	3889/M						
swapin_queue_empty	3363	3389	3989					
swapin_queue_empty	3364	3370/M	3373/M	3383/M	3989/M	3989/M	3989/M	
swapout_reason	123	3988	4305/M					
swapout_reason	4286	4305	4306	4310				
swapout_timestamp	138	3979						
swapouts	514	4468	4469/M	4474	4475/M			
swapped_jobs	2059	1362/M	1362	1366/M	1366	3653/M	3654	3671/M	3672
		3744/M	3744	3744/M	3744	3793/M	3793	3795/M	3795
		3887/M	3887	3887/M	3887	3926/M	3927	4006/M	4006
		4006/M	4006	4059/M	4059	4059/M	4059	4125/M	4125
		4125/M	4125	4172/M	4172	4172/M	4172	4213/M	4213
		4213/M	4213	4236/M	4236	4236/M	4236	4372/M	4372
		4372/M	4372						
swapped_jobs	2079	1362/M	1362	1366/M	1366	3380	3397	3655/M	3656
		3673/M	3674	3744/M	3744	3744/M	3744	3793/M	3793
		3795/M	3795	3826	3887/M	3887	3887/M	3887	3924/M
		3925	3989	3991	4006/M	4006	4006/M	4006	4059/M
		4059	4059/M	4059	4125/M	4125	4125/M	4125	4172/M
		4172	4172/M	4172	4213/M	4213	4213/M	4213	4236/M
		4236	4236/M	4236	4372/M	4372	4372/M	4372	
swapping_io_error	133	4167						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES						
syc$rc_job_scheduler_request	2528	3290						
system_supplied_name	66	3292						
system_supplied_name	2420	3292/M	3773/P					
syt$monitor_request_code	2445	2412	2551					
syt$monitor_status	1646	1420	1591	2413	2552	3344	3407	3874	3946
		4117	4154	4211	4260	4275			
terminated_job	1165	4237/M						
think_time	3948	3965/M	3966	3968	3969			
time_left_in_interval	1267	3481/M	3483	3513				
time_spent_in_job_mode	545	3495	3621	3819	4433	4434	4435/M	4436	
time_spent_in_mtr_mode	546	3496	3622	3820	4439	4440	4442/M	4443	
timestamp	137	3872/M							
timestamp	2017	3493/M							
tmc$fnx_job	1731	3821/P							
tmc$maximum_system_task_id	1654	1657							
tmc$stid_job_scheduler	1666	3351/P	3409/P	3541/P	3542/P	3828/P	3959/P	4013/P	4065/P
		4097/P	4133/P	4178/P	4220/P	4238/P	4265/P	4278/P	4350/P
tmc$stid_null_task	1660	1657							
tmp$calculate_dct_priority_int	1496	3527							
tmp$clear_lock	1499	1512	3538	3638	3752	3786	3832	3892	3933
		3981	4060	4127	4174	4477			
tmp$free_unrecovered_tasks	1584	3931							
tmp$monitor_ready_system_task	1590	3351	3409	3541	3542	3828	3959	4013	4065
		4097	4133	4178	4220	4238	4265	4278	4350
tmp$reset_dispatching_control	1692	3631							
tmp$set_lock	1701	1719	3456	3575	3734	3775	3886	3919	3976
		4058	4124	4171	4429				
tmp$update_job_task_environment	1724	3821							
tmt$cpu_execution_statistics	1238	1233							
tmt$dispatching_control_sets	1291	1281	3436						
tmt$dispatching_controls	1252	1247							
tmt$dispatching_prio_controls	1266	1257	1302						
tmt$dispatching_priority_time	1271	1268							
tmt$fnx_search_type	1731	1726	1734						
tmt$ptl_lock	1311	1307	1499	1701	3251	3251			
tmt$system_task_id	1657	1590							
tmt$task_queue_link	1084	1053							
tmt$time_limits	1273	1271							
tmv$cpu_execution_statistics	1233	3495	3496						
tmv$dispatch_priority_integer	1244	3535							
tmv$dispatching_control_sets	1281	3470/M	3471/M	3472/M	3520/M	3529			
tmv$dispatching_control_time	1302	3521/M							
tmv$dispatching_controls	1247	3473/M	3476/M	3478/M	3479/M	3480/M	3481/M	3483	3486/M
		3489/M	3489	3503/M	3506/M	3509/M	3509	3512/M	3513
		3516/M	3516	3520	3521				
tmv$ptl_lock	1307	3456/P	3538/P	3575/P	3638/P	3734/P	3752/P	3775/P	3786/P
		3832/P	3886/P	3892/P	3919/P	3933/P	4058/P	4060/P	4124/P
		4127/P	4171/P	4174/P					
too_many_active_jobs_for_class	3393	3380	3400	3989	3991				
too_many_active_jobs_for_class	3394	3380/M	3396/M	3989/M	3991/M				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE Job Management : job scheduler monitor mode
[XDCL] jmp$update_service_class_stats

IDENTIFIER	DEFINED ON LINE	REFERENCES	
u_second	3430	3482	
unblocked_priorities	2007	3497/M	3498
value	1432	1440	1443/M
value	3316	3329	3329/M
variable	1431	1440	
variable	1447	1487	
variable	3316	3329	
variable	3316	3333	

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper

```
 3 MODULE jsm$monitor_mode_job_swapper;
 4
 5 [
 6 {   The purpose of this module is to do the work necessary to swap jobs in and
 7 { out once it has been informed to do so.  Some work may have to be done in
 8 { job mode having to do with allocating the swap file.
 9 {
10 {   The actual swapping of the job is a serial function in a multi cpu system
11 { although the requests can be received asynchronously to request a swap or
12 { advance a swap.  These asynchronous requests are serialized by noting the
13 { event, the actual work is performed when the job swapper (jsp$swap_polling)
14 { is called from mtm$monitor_interrupt_handler asynchronous loop.  Procedures
15 { that can be entered asynchronously are marked.
16 {
17
18
19
20 {  Define compile time variable to control compilation of debug code.
21
22   ?VAR
23     debug: boolean := FALSE?;
24
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  5264
o  5265
o  5266 {  External procedures referenced by this module.
o  5267
o  5268
o  5269    PROCEDURE [XREF] dfp$fetch_page_status
o  5270       (     fde_p: gft$locked_file_desc_entry_p;
o  5271             offset: ost$segment_offset;
o  5272         VAR allocate_status: gft$page_status);
o  5273
   5276    PROCEDURE [XREF] dfp$set_task_segment_state
   5277      (     search: tmt$fnx_search_type;
   5278            ijle_p: ^jmt$initiated_job_list_entry;
   5279            ijlo: jmt$ijl_ordinal;
   5280            inhibit_access_work: dft$mainframe_set;
   5281            terminate_access_work: dft$mainframe_set);
   5282
o  5298    VAR
o  5299      dfv$file_server_debug_enabled: [XREF] boolean;
o  5300
o  5301
o  5302    PROCEDURE [XREF] dmp$allocate_file_space
o  5303      (     p_fde: gft$locked_file_desc_entry_p;
o  5304            initial_byte_address: amt$file_byte_address;
o  5305            bytes_to_allocate: amt$file_byte_address;
o  5306            file_space_limit: sft$file_space_limit_kind;
o  5307       VAR allocation_units_obtained: amt$file_byte_address;
o  5308       VAR overflow_indicator: boolean;
o  5309       VAR file_allocation_status: dmt$file_allocation_status);
o  5310
   5313
   5314    PROCEDURE [XREF] dmp$set_fau_state
   5315      (     fde_p: gft$locked_file_desc_entry_p;
   5316            byte_address: amt$file_byte_address;
   5317       VAR status: syt$monitor_status);
   5318
o  5321
o  5322    PROCEDURE [XREF] dmp$recover_job_dm_tables
o  5323      (     ijle_p: ^jmt$initiated_job_list_entry);
o  5324
   5327
   5328    PROCEDURE [XREF] dpp$display_error
   5329      (     line: string ( * <= dpc$top_line_message_size));
   5330
o  5342
o  5343    PROCEDURE [INLINE] gfp$mtr_get_fde_p (sfid: gft$system_file_identifier;
o  5344          ijle_p: ^jmt$initiated_job_list_entry;
o  5345      VAR fde_p: gft$file_desc_entry_p);
o  5346
o  5347
o  5390
o  5391    PROCEDURE [INLINE] gfp$mtr_get_locked_fde_p (sfid: gft$system_file_identifier;
o  5392          ijle_p: ^jmt$initiated_job_list_entry;
o  5393      VAR fde_p: gft$locked_file_desc_entry_p);
o  5394
```


NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  5395
o  5456
o  5457    PROCEDURE [XREF] i#build_adaptable_array_ptr (ring: 0 .. 15;
o  5458          segment: 0 .. 4095;
o  5459          offset: -80000000(16) .. 7fffffff(16);
o  5460          array_size: 1 .. 80000000(16);
o  5461          lower_bound: -80000000(16) .. 7fffffff(16);
o  5462          element_size: 1 .. 80000000(16);
o  5463          array_p: ^^cell);
o  5464
o  5465    PROCEDURE [XREF] i#real_memory_address (p: ^cell;
o  5466      VAR rma: integer);
o  5467
o  5468    PROCEDURE [XREF] iop$pager_io (
o  5469      fde_p: gft$locked_file_desc_entry_p;
o  5470      chapter_offset: ost$segment_offset;
o  5471      buffer_descriptor: mmt$buffer_descriptor;
o  5472      length: ost$byte_count;
o  5473      io_function: iot$io_function;
o  5474      io_identifier: mmt$io_identifier;
o  5475      VAR status: syt$monitor_status);
   5478
   5479 PROCEDURE [XREF] jmp$activate_job_mode_swapper;
   5480 PROCEDURE [XREF] jmp$assign_ajl_entry (asid: ost$asid,
   5481        ijl_o: jmt$ijl_ordinal;
   5482        caller: 0 .. 0l0(16);
   5483        must_assign: boolean;
   5484    VAR ajl_o: jmt$ajl_ordinal;
   5485    VAR status: syt$monitor_status);
   5486
o  5493
o  5494 PROCEDURE [XREF] jmp$assign_ajl_with_lock
o  5495    (   asid: ost$asid;
o  5496        ijl_o: jmt$ijl_ordinal;
o  5497        caller: 0 .. 0l0(16);
o  5498        must_assign: boolean;
o  5499    VAR ajl_o: jmt$ajl_ordinal;
o  5500    VAR status: syt$monitor_status);
o  5501
   5504
   5505 { PURPOSE:
   5506 {   This is the monitor mode procedure to change the entry status of a job.  The caller
   5507 {   of procedure must set the PTL lock if the entry status change is a SWAPPED/NOT SWAPPED
   5508 {   transition because the swapped job counts will be changed.
   5509
   5510    PROCEDURE [INLINE] jmp$change_ijl_entry_status
   5511      (     ijle_p: ^jmt$initiated_job_list_entry;
   5512            new_entry_status: jmt$ijl_entry_status);
   5513
   5514      VAR
   5515        old_entry_status: jmt$ijl_entry_status;
   5516
   5517      old_entry_status := ijle_p^.entry_status;
   5518
   5519      jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] :=
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
5520            jmv$ijl_entry_status_statistics [old_entry_status] [new_entry_status] + 1;
5521
5522      ijle_p^.entry_status := new_entry_status;
5523
5524      IF (old_entry_status <= jmc$ies_swapin_in_progress) AND
5525            (new_entry_status > jmc$ies_swapin_in_progress) THEN
5526        jmp$increment_swapped_job_count (ijle_p);
5527
5528      ELSEIF (old_entry_status > jmc$ies_swapin_in_progress) AND
5529            (new_entry_status <= jmc$ies_swapin_in_progress) THEN
5530        jmp$decrement_swapped_job_count (ijle_p);
5531      IFEND;
5532
5533    PROCEND jmp$change_ijl_entry_status;
5534
5535    PROCEDURE [INLINE] jmp$check_scheduler_memory_wait;
5536
o   5569    PROCEDURE [XREF] jmp$decrement_swapped_job_count(ijle_p: ^jmt$initiated_job_list_entry);
o   5570
    5573 PROCEDURE [XREF] jmp$free_ajl_entry
    5574   (     ijle_p: ^jmt$initiated_job_list_entry;
    5575         caller: 0 .. 10(16));
    5576
o   5579
o   5580 PROCEDURE [XREF] jmp$free_ajl_with_lock
o   5581   (     ijle_p: ^jmt$initiated_job_list_entry;
o   5582         caller: 0 .. 10(16));
o   5583
    5586    PROCEDURE [inline] jmp$get_ijle_p (ijl_ordinal: jmt$ijl_ordinal;
    5587      VAR ijle_p: ^jmt$initiated_job_list_entry);
    5588
o   5596
o   5597    PROCEDURE [XREF]  jmp$increment_swapped_job_count (ijle_p: ^jmt$initiated_job_list_entry);
o   5598
    5601    PROCEDURE [XREF] jmp$recognize_job_dead (ijl_ordinal: jmt$ijl_ordinal);
    5602
o   5605
o   5606    PROCEDURE [XREF] jmp$reset_job_to_swapped_out (ijl_o: jmt$ijl_ordinal);
o   5607
    5610
    5611    PROCEDURE [XREF] jmp$set_entry_status_to_rt
    5612      (     ijl_ordinal: jmt$ijl_ordinal;
    5613            ijle_p: ^jmt$initiated_job_list_entry);
    5614
o   5617
o   5618    PROCEDURE [XREF] jmp$set_scheduler_event (event: jmt$job_scheduler_events);
o   5619
    5622
    5623    PROCEDURE [XREF] jsp$initiate_swapout_io (pages_needed: mmt$page_frame_index);
    5624
o   5627
o   5628    PROCEDURE [XREF] mmp$asid (asti: mmt$ast_index;
o   5629      VAR asid: ost$asid);
o   5630
    5633
```


NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
5634    PROCEDURE [XREF] mmp$claim_pages_for_swapin (swapped_job_entry: jmt$swapped_job_entry;
5635          aste_p: ^mmt$active_segment_table_entry;
5636          ijl_ordinal: jmt$ijl_ordinal;
5637      VAR job_page_queue_list: mmt$job_page_queue_list);
5638
o   5641
o   5642    PROCEDURE [XREF] mmp$dump_shared_queue
o   5643      (     total_pages_needed: mmt$page_frame_index);
o   5644
5647
5648    PROCEDURE [XREF] mmp$free_memory_in_job_queues (VAR job_page_queue_list: mmt$job_page_queue_list;
5649          increment_now: boolean;
5650          decrement_soon: boolean;
5651          job_termination: boolean);
5652
o   5655
o   5656    PROCEDURE [XREF] mmp$replenish_free_queues (asid: ost$asid);
o   5657
o   5658
o   5659
o   5660 { This procedure verifies that the asti stored in the file descriptor entry is still being used by
o   5661 { the same job for the same file.  If the asti is ok, it is returned; otherwise 0 is returned.
o   5662
o   5663    PROCEDURE [INLINE] mmp$get_verify_asti_in_fde
o   5664      (     fde_p: gft$locked_file_desc_entry_p;
o   5665            sfid: gft$system_file_identifier;
o   5666            ijlo: jmt$ijl_ordinal;
o   5667       VAR asti: mmt$ast_index);
o   5668
5680
5681    PROCEDURE [INLINE] mmp$sva_purge_all_page_map (sva: ost$system_virtual_address);
5682
5683      IF mmv$multiple_page_maps THEN
5684        mmp$purge_all_map_proc;
5685      ELSE
5686        #purge_buffer (osc$sva_purge_all_page_map, sva);
5687      IFEND;
5688
5689    PROCEND;
o   5711
o   5712    PROCEDURE [XREF] mmp$remove_swapped_shared_pages
o   5713      (     ijle_p: ^jmt$initiated_job_list_entry);
o   5714
5717
5718    PROCEDURE [XREF] mmp$age_job_working_set (ijle_p: ^jmt$initiated_job_list_entry;
5719          jcb_p: ^jmt$job_control_block);
5720
o   5723    PROCEDURE [XREF] mmp$remove_stale_pages (VAR pqle: mmt$page_queue_list_entry;
o   5724          age_limit: integer;
o   5725          jcb_p: ^jmt$job_control_block;
o   5726          ijle_p: ^jmt$initiated_job_list_entry;
o   5727          queue_id: mmt$page_frame_queue_id;
o   5728          minimum_working_set: 0 .. 0ffff(16);
o   5729      VAR modified_pages_removed: integer;
o   5730      VAR total_pages_removed: integer);
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  5731
   5734
   5735   PROCEDURE [XREF] mmp$assign_asid (VAR asid: ost$asid;
   5736     VAR asti: mmt$ast_index;
   5737     VAR aste_p: ^mmt$active_segment_table_entry);
   5738
o  5741   PROCEDURE [XREF] mmp$assign_specific_asid (aste_p: ^mmt$active_segment_table_entry);
o  5742
   5745   PROCEDURE [XREF] mmp$assign_page_to_monitor (p: ^cell;
   5746         page_count: integer;
   5747         preset: boolean;
   5748     VAR status: syt$monitor_status);
   5749
o  5752   PROCEDURE [XREF] mmp$asti (asid: ost$asid;
o  5753     VAR asti: mmt$ast_index);
o  5754
   5757
   5758   PROCEDURE [INLINE] mmp$conditional_purge_all_map (time: integer);
   5759
   5760     VAR
   5761       null_sva: 0 .. 0ffffffffffff(16);
   5762
   5763     IF mmv$multiple_page_maps THEN
   5764       IF time > mmv$time_map_last_purged THEN
   5765         mmp$purge_all_map_proc;
   5766       IFEND;
   5767     ELSE
   5768       #purge_buffer (osc$purge_all_page_seg_map, null_sva);
   5769     IFEND;
   5770
   5771   PROCEND;
o  5779   PROCEDURE [XREF] mmp$delete_page_from_monitor (p: ^cell;
o  5780         page_count: integer;
o  5781     VAR status: syt$monitor_status);
o  5782
   5785
   5786   PROCEDURE [XREF] mmp$free_asid (asid: ost$asid;
   5787         aste_p: ^mmt$active_segment_table_entry);
   5788
o  5791
o  5792   PROCEDURE [XREF] mmp$delete_pt_entry
o  5793     (     pfti: mmt$page_frame_index;
o  5794           unlink_page_from_segment: boolean);
o  5795
   5798
   5799   PROCEDURE [INLINE] mmp$get_max_sdt_sdtx_pointer
   5800     (     xcb_p: ^ost$execution_control_block;
   5801       VAR sdt_p: mmt$max_sdt_p;
   5802       VAR sdtx_p: mmt$max_sdtx_p);
   5803
o  5916
o  6917   PROCEDURE [XREF] mmp$make_pt_entry (sva: ost$system_virtual_address;
o  6918         pfti: mmt$page_frame_index;
o  6919         aste_p: ^mmt$active_segment_table_entry;
o  6920         pfte_p: ^mmt$page_frame_table_entry;
```


NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  6921     VAR mpt_status: mmt$make_pt_entry_status);
o  6922
   6925
   6926   PROCEDURE [INLINE] mmp$nudge_periodic_call;
   6927
   6928     mmv$time_to_call_mem_mgr := 0;
   6929     osv$time_to_check_asyn := 0;
   6930
   6931   PROCEND mmp$nudge_periodic_call;
   6932
o  6942   PROCEDURE [XREF] mmp$process_page_table_full (sva: ost$system_virtual_address;
o  6943     VAR new_asid: ost$asid;
o  6944     VAR new_asti: mmt$ast_index;
o  6945     VAR new_aste_p: ^mmt$active_segment_table_entry;
o  6946     VAR pt_full_status: mmt$pt_full_status);
o  6947
   6956
   6957   PROCEDURE [XREF] mmp$relink_page_frame (pfti: mmt$page_frame_index;
   6958         queue_id: mmt$page_frame_queue_id);
   6959
o  6962
o  6963   PROCEDURE [XREF] mmp$trim_job_working_set
o  6964     (     ijle_p: ^jmt$initiated_job_list_entry;
o  6965           jcb_p: ^jmt$job_control_block;
o  6966           trim_to_swap_size: boolean );
o  6967
o  6968
   6971   PROCEDURE [XREF] mmp$write_page_to_disk
   6972     (    fde_p: gft$locked_file_desc_entry_p;
   6973         pfti: mmt$page_frame_index;
   6974         iotype: iot$io_function;
   6975         io_id: mmt$io_identifier;
   6976         multiple_page_req: boolean;
   6977     VAR write_status: mmt$write_page_to_disk_status);
   6978
o  6981   PROCEDURE [INLINE] mtp$cst_p (VAR cst_p: ^ost$cpu_state_table);
   7196
   7197   PROCEDURE [INLINE] mtp$set_status_abnormal (identifier: string (2);
   7198         condition: osc$max_status_condition_number + 1 .. 0fffffffff(16);
   7199     VAR status: syt$monitor_status);
   7200
o  7210
o  7211 PROCEDURE [INLINE] tmp$clear_lock (VAR lock: tmt$ptl_lock);
o  7212
o  7213     IF osv$cpus_logically_on > 1 THEN
o  7214       IF lock.id <> #READ_REGISTER (osc$pr_base_constant) THEN
o  7215         i#program_error; (Interlock failure - no message passed for performance reasons)
o  7216       IFEND;
o  7217       IF lock.count > 0 THEN
o  7218         lock.count := lock.count - 1;
o  7219       ELSE
o  7220         lock.clear := 0;
o  7221       IFEND;
o  7222     IFEND;
o  7223
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  7224 PROCEND   tmp$clear_lock;
o  7225
   7234
   7235   PROCEDURE [XREF] tmp$find_next_xcb (search: tmt$fnx_search_type;
   7236         ijle_p: ^jmt$initiated_job_list_entry;
   7237         ijl_ordinal: jmt$ijl_ordinal;
   7238      VAR state: tmt$find_next_xcb_state;
   7239      VAR xcb_p: ^ost$execution_control_block);
   7240
o  7243
o  7244   PROCEDURE [XREF] tmp$idle_tasks_in_job
o  7245     (    ajl_ordinal: jmt$ajl_ordinal;
o  7246          swapout_reason: jmt$swapout_reasons;
o  7247       VAR status: syt$monitor_status);
o  7248
   7251
   7252   PROCEDURE [XREF] tmp$monitor_flag_job_tasks
   7253     (    monitor_flag_id: syt$monitor_flag;
   7254          ijle_p: ^jmt$initiated_job_list_entry);
   7255
   7256
o  7259   PROCEDURE [XREF] tmp$restart_idled_tasks (ajl_ordinal: jmt$ajl_ordinal);
o  7260
   7263
   7264   PROCEDURE [INLINE] tmp$set_lock (VAR lock: tmt$ptl_lock);
   7265
   7266     VAR
   7267       b: boolean,
   7268       bc: integer;
   7269
   7270     IF osv$cpus_logically_on > 1 THEN
   7271       bc := #read_register (osc$pr_base_constant);
   7272       IF lock.id <> bc  THEN
   7273         REPEAT
   7274          #TEST_SET (lock.locked, b);
   7275         UNTIL NOT b;
   7276         lock.id := bc;
   7277       ELSE
   7278         lock.count := lock.count + 1;
   7279       IFEND;
   7280     IFEND;
   7281
   7282 PROCEND tmp$set_lock;
   7283
o  7286
o  7287   PROCEDURE [XREF] tmp$set_monitor_flag (task_id: ost$global_task_id;
o  7288         flag_id: syt$monitor_flag;
o  7289      VAR status: syt$monitor_status);
   7292   PROCEDURE [XREF] tmp$set_up_debug_registers (ptlo: ost$task_index;
   7293         ijle_p: ^jmt$initiated_job_list_entry;
   7294         xcb_p: ^ost$execution_control_block);
o  7297
o  7298   PROCEDURE [XREF] tmp$update_job_task_environment (ijle_p: ^jmt$initiated_job_list_entry;
o  7299         ijl_ordinal: jmt$ijl_ordinal;
o  7300         xcb_search: tmt$fnx_search_type);
```


NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
o  7301
   7304
   7305 {  Global variables referenced by this module.
   7306
   7307
   7308   VAR
   7309     dmv$p_active_volume_table: [XREF, oss$mainframe_wired] ^dmt$active_volume_table;
   7310
o  7621 {Pointer to the AJL.}
o  7622   VAR
o  7623     jmv$ajl_p: [XREF] ^jmt$active_job_list;
   7648
   7649   VAR
   7650     jmv$ijl_entry_status_statistics: [XREF] jmt$ijl_entry_status_statistics;
   7651
o  7661 {Define pointer to Initiated Job List (IJL).
o  7662
o  7663   VAR
o  7664     jmv$ijl_p: [XREF]  jmt$ijl_p;
   7692
   7693   VAR
   7694     jmv$long_wait_swap_threshold: [XREF] integer;
   7695
   7696   VAR
   7697     jmv$null_ijl_ordinal: [XREF] jmt$ijl_ordinal;
   7698
o  7701   VAR
o  7702     jmv$service_classes: [XREF, oss$mainframe_wired]
o  7703           array [jmt$service_class_index] of ^jmt$service_class_entry;
o  7704
   7747
   7748
   7749   VAR
   7750     jmv$service_class_stats_lock: [XREF] tmt$ptl_lock;
   7751
o  7754 {Define value of AJL ORDINAL used by the system job
o  7755
o  7756   VAR
o  7757     jmv$system_ajl_ordinal: [XREF]jmt$ajl_ordinal;
o  7758
   7761   VAR
   7762     jmv$system_ijl_ordinal: [XREF] jmt$ijl_ordinal;
   7763
o  7766
o  7767   VAR
o  7768     jmv$system_job_ssn: [XREF] jmt$system_supplied_name;
o  7769
   7772 {  Time for next periodic call to job swapper from
   7773 {mtm$monitor_interrupt_handler.
   7774
   7775   VAR
   7776     jsv$time_to_call_job_swapper: [XREF] integer;
   7777
   7778 {Define minimum number of pages that must be kept in the free + available page
   7779 {queues. If the actual number drops below this value, memory manager begins
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
          7780 {an aggressive aging policy. If the number of page frames drops below mmv$aggressive_aging_level_2
          7781 {then only critical system tasks are assigned memory. User tasks are put into a memory wait queue.
          7782
          7783   VAR
          7784     mmv$aggressive_aging_level: [XREF] integer,
          7785     mmv$aggressive_aging_level_2: [XREF] integer;
          7786
          7787 {The following variable defines the agining algorithm that is used by memory manager.
          7788 {    0 - no swapping active
          7789 {    1 - swapping active
          7790 {  > 1 - to be defined
          7791   VAR
          7792     mmv$aging_algorithm: [XREF] integer;
          7793 {Pointer to the Active Segment Table - (AST).}
          7794
          7795   VAR
          7796     mmv$ast_p: [XREF] ^mmt$active_segment_table;
          7797
      o   7800
      o   7801 { Global Page Queue List array.
      o   7802
      o   7803   VAR
      o   7804     mmv$gpql: [XREF] mmt$global_page_queue_list;
          7807
          7808 {Define template for an AST entry for a job fixed segment. This is used by the job swapper to
          7809 {create an AST entry for job fixed of a job being swapped in.
          7810
          7811   VAR
          7812     mmv$initial_job_fixed_ast_entry: [XREF] mmt$active_segment_table_entry;
          7813
      o   7816
      o   7817   VAR
      o   7818     mmv$max_working_set_size: [XREF] integer;
      o   7819
      o   7820 { The following variable contains the maximum segment number of a global template segment.
      o   7821
      o   7822   VAR
      o   7823     mmv$max_template_segment_number: [XREF] integer;
      o   7824
      o   7825   VAR
      o   7826     mmv$min_avail_pages: [XREF] integer;
      o   7827
      o   7828 {The following variable indicates if the configuration consists of multiple
      o   7829 {page MAPS that are not hardware connected for unified map purging - ie,
      o   7830 {if a page map purge is required each processor must purge its own map.
      o   7831
      o   7832   VAR
      o   7833     mmv$multiple_page_maps: [XREF] boolean;
      o   7834
      o   7835 {Pointer to the 'PAGE FRAME TABLE' (PFT)
      o   7836
      o   7837   VAR
      o   7838     mmv$pft_p: [XREF] ^mmt$page_frame_table;
      o   7839
          7842
```


NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
          7843   VAR
          7844     mmv$reserved_page_count: [XREF] integer;
          7845
          7846
          7847 {  Define a variable to contain the index of the last shared site queue that is actually being used.
          7848
          7849   VAR
          7850     mmv$last_active_shared_queue: [XREF] mmt$global_page_queue_index;
      o   7853
      o   7854
      o   7855   VAR
      o   7856     mmv$swapping_aic: [XREF] integer;
      o   7857
      o   7858 { Timestamp that contains the free-running-clock value when a global ASID was last changed.
      o   7859
      o   7860   VAR
      o   7861     mmv$time_changed_global_asid: [XREF] ost$free_running_clock;
      o   7862
          7865
          7866 { Timestamp that contains the free-running-clock value when a global ASID was last changed.
          7867
          7868   VAR
          7869     mmv$time_changed_template_asid: [XREF] ost$free_running_clock;
          7870
      o   7873
      o   7874 {Pointer to the system PAGE TABLE (PT).
      o   7875
      o   7876   VAR
      o   7877     mmv$pt_p: [XREF] ^ost$page_table;
      o   7878
          7881 {The following variable contains a count of the number of page frames that can be reassigned to be
          7882 {used for another purpose. The count represents the number of pages that are in the free + available
          7883 {queues. The count is broken into two parts - pages with no IO active, and pages with IO active.
          7884
          7885   VAR
          7886     mmv$reassignable_page_frames: [XREF] mmt$reassignable_page_frames;
      o   7896 {Monitor segment table.}
      o   7897
      o   7898   VAR
      o   7899     mtv$monitor_segment_table: [XREF] record
      o   7900       st: ALIGNED [0 MOD 8] array [0 .. 4095] of mmt$segment_descriptor,
      o   7901     recend;
      o   7902
          7905 {Define SMU Communications Block (SCB).
          7906
          7907   VAR
          7908     mtv$scb: [XREF] mtt$smu_communications_block;
      o   8044
      o   8045  VAR
      o   8046     mtv$system_job_monitor_xcb_p: [XREF] ^ost$execution_control_block;
      o   8047
          8050 {System page size.}
          8051
          8052   VAR
          8053     osv$page_size: [XREF] ost$page_size;
```

NOS/VE js : monitor mode job swapper
Global Declarations Referenced by This Module

```
        8054
      o 8057 VAR
      o 8058     tmv$ptl_lock: [XREF] tmt$ptl_lock;
      o 8061
      o 8062   VAR
      o 8063     tmv$swapin_in_progress: [XREF] integer;
      o 8064
```


NOS/VE js : monitor mode job swapper
Global Declarations Declared by This Module

```
      o 8067
      o 8068 {  Global constants defined by this module.
      o 8069
      o 8070 ?? FMT (FORMAT := OFF) ??
      o 8071
      o 8072   CONST
      o 8073     reassigned_asid_list_length = 20,
      o 8074
      o 8075 {  Define trace indexes for swap trace buffer.  JSC$TI_UNUSED_XX identifies free indexes.
      o 8076
      o 8077     jsc$ti_min_index = 0,
      o 8078     jsc$ti_no_memory_for_swap_in = 1,
      o 8079     jsc$ti_new_job_fixed_asid = 2,
      o 8080     jsc$ti_reuse_job_fixed_asid = 3,
      o 8081     jsc$ti_reuse_job_fixed_asid_as = 4,     { Reassign old ASID to job fixed.}
      o 8082     jsc$ti_no_pages_for_sfd_on_si = 5,
      o 8083     jsc$ti_sfd_freed = 6,
      o 8084     jsc$ti_free_memory_si_aborted = 7,
      o 8085     jsc$ti_free_memory = 8,
      o 8086     jsc$ti_pager_io_error = 10,
      o 8087     jsc$ti_move_am_back_to_am = 11,
      o 8088     jsc$ti_move_am_back_to_am_pc = 12,      { Page count of pages moved back to available modified.}
      o 8089     jsc$ti_flush_am_pc = 13,                { Page count of pages in am that were flushed.}
      o 8090     jsc$ti_flush_am_relink = 14,            { Move am back to jws--write to disk reject.}
      o 8091     jsc$ti_flush_am_ready = 15,             { Task ready after flush.}
      o 8092     jsc$ti_swapping_queue_and_exec = 16,    { Swap status of executing and swap direction of in.}
      o 8093     jsc$ti_allocate_swap_file = 17,         { Call DM to allocate swap file in monitor mode.}
      o 8094     jsc$ti_allocate_swap_file_jm = 18,      { Allocate swap file in job mode.}
      o 8095     jsc$ti_dm_transient_error = 19,         { Device management transient error.}
      o 8096     jsc$ti_change_asid_again = 20,
      o 8097     jsc$ti_change_asid = 21,
      o 8098     jsc$ti_change_asid_sfd = 22,            { Update changed ASID's in swap file descriptor.}
      o 8099
      o 8100 {  Trace indexes for events during reset to memory manager tables.
      o 8101
      o 8102     jsc$ti_rmmt_no_change = 24,             { No change in ASID.}
      o 8103     jsc$ti_rmmt_pf = 25,                    { ASID change of page belonging to a permanent file.}
      o 8104     jsc$ti_rmmt_pf_rec_ptm = 26,            { Assign new ASID on job recovery and modified.}
      o 8105     jsc$ti_rmmt_pf_rec_ptu = 27,            { Job recovery, relink unmodified page into free queue.}
      o 8106     jsc$ti_rmmt_pf_assign_asid = 28,        { Not job recovery, assign new ASID.}
      o 8107     jsc$ti_rmmt_pf_reuse_asid = 29,         { Not job recovery, reuse ASID.}
      o 8108     jsc$ti_rmmt_lf_assign_asid = 30,        { Assign ASID for page assigned to local file.}
      o 8109     jsc$ti_rmmt_lf_reuse_asid = 31,         { Reuse ASID for page assigned to local file.}
      o 8110     jsc$ti_rmmt_pt_done = 32,
      o 8111     jsc$ti_rmmt_pt_full = 33,
      o 8112     jsc$ti_rmmt_pt_full_failed = 34,
      o 8113     jsc$ti_rmmt_pt_full_succ = 35,          { Succeeded in recovering from page table full.}
      o 8114     jsc$ti_rmmt_pte_exists_pf = 36,         { Permanent file page is now in shared queue.}
      o 8115     jsc$ti_rmmt_pte_exists_am = 37,         { Local file page is still in Avail modeified queue.}
      o 8116     jsc$ti_rmmt_pte_exists_a = 38,          { Local file page found in Avail queue.}
      o 8117     jsc$ti_rmmt_pte_exists_err = 39,        { Local file page found in Swapped error queue.}
      o 8118
      o 8119 {  Trace buffer indexes for reset xcb and sdt tables.
      o 8120
      o 8121     jsc$ti_rxcb_temp_asids_changed = 40,
```

```
      jsc$ti_rxcb_job_asids_changed = 41,
      jsc$ti_rxcb_glob_asids_changed = 42,
      jsc$ti_rxcb_fix_xcb_sdt = 43,
      jsc$ti_rxcb_fix_asids = 44,
      jsc$ti_rxcb_fix_templ_asid = 45,
      jsc$ti_pt_full_reassign_jf = 46,
      jsc$ti_rxcb_fix_jf_asid = 47,
      jsc$ti_rxcb_fix_job_asid = 48,
      jsc$ti_rxcb_zero_job_asid = 49,
      jsc$ti_rxcb_recovery = 53,
      jsc$ti_rxcb_zero_asid = 54,             { Reset tables zeroed out an ASID in a segment table.}

      jsc$ti_lwa = 55,                        { Long wait aging called}
      jsc$ti_lwa_cp_age = 56,                 { called cp aging}
      jsc$ti_lwa_stale_pages_rem = 57,        { total number of pages removed}
      jsc$ti_lwa_stale_mod_pages_rem = 58,    { number modified pages removed}
      jsc$ti_lwa_ready_task = 59,             { long wait aging caused task to go ready.}

      jsc$ti_swapin_io_error = 60,
      jsc$ti_swapout_io_error = 61,
      jsc$ti_sif_idle_tasks_init = 63,        { Swap in from idle tasks initiated.}
      jsc$ti_sif_wait_state = 64,             { Swap in from a wait state.}
      jsc$ti_sif_swapout_io_init = 65,        { Swap in from swap out io initiated or completed.}
      jsc$ti_swapout_int_by_swapin = 67,      { Swap in requested on job being swapped out.}
      jsc$ti_swapin_int_by_swapout = 68,      { Swap out requested on job being swapped in.}
      jsc$ti_no_ajl_ord_for_swap_in = 69,     { Swap in aborted, could not assign AJL ordinal.}
      jsc$ti_swapout_from_job_mode = 76,      { Swapout request from job mode}
      jsc$ti_swapout_from_mtr_mode = 77,      { Swapout from monitor mode.}
      jsc$ti_swapin_from_job_mode = 78,       { Swapin from job mode.}
      jsc$ti_swapin_from_mtr_mode = 79,       { Swapin from monitor mode.}
      jsc$ti_swapin_mtr_direct = 80,          { Swapin from monitor mode - S0 to R.}
      jsc$ti_swapin_req_status_bad = 81,      { Swapin from job mode--advance_swap got bad status.}
      jsc$ti_cd_idle_task_complete = 82,      { Change direction to in detected in idld task complete.}
      jsc$ti_sif_idled_tasks_comp = 83,       { Swap in from idle tasks complete.}
      jsc$ti_cd_idle_task_complete_2 = 86,    { Change direction to IN in idle task complete--2nd check.}
      jsc$ti_reserve_memory_failed = 87,      { Memory no longer available for reserve request on swapin.}
      jsc$ti_cd_to_in_at_s2 = 88,             { Changed direction to in at swapped_io_complete.}
      jsc$ti_cd_to_in_at_s = 89,              { Changed direction to in at swapout_complete.}
      jsc$ti_init_swapin_io_error = 90,       { IO error discovered upon swapin io complete.}
      jsc$ti_init_swapout_io_error = 91,      { IO error discovered upon swapout io complete.}
      jsc$ti_swapout_disk_down = 93,          { Disk down discovered upon swapout io complete.}
      jsc$ti_swapin_disk_down = 94,           { Disk down discovered upn swapin io compelete.}
      jsc$ti_zero_out_pages_for_sfd_1 = 95,   { Abort swapout at wait alloc sfd--swapin req--0 out pages needed.}
      jsc$ti_zero_out_pages_for_sfd_2 = 96,   { Adv swapout from wait alloc sfd--polling--0 out pages needed.}
      jsc$ti_no_ajlo_swapin_before_io = 97,   { No ajlo available in swapin_before_io. }
      jsc$ti_dump_shared_q_for_sfd = 98,      { Dump the shared queue to get pages for an SFD.}
      jsc$ti_dump_shared_queue = 99,          { Dump the shared queue to claim enough pages to swap a job in.}
      jsc$ti_free_readied_s2_job = 100,       { Free an S2 job that has been readied in order to use its memory.}
      jsc$ti_no_ajlo_swapin_after_io = 101,   { No ajlo available in swapin_after_io. }
      jsc$ti_advance_from_cannot_init = 103,  { Advance swap state from io_cannot_init to io_not_init. }
      jsc$ti_page_q_counts_different = 104,   { Page q counts different at job_io_complete. }
      jsc$ti_mtr_req_adv_from_aj = 105,       { Mtr request to advance from job_allocate_swap_file. }
      jsc$ti_mtr_req_adv_from_sd = 106,       { Mtr request to advance from swapped_io_cannot_init. }
      jsc$ti_recalculate_sje = 107,           { Recalculate the sje after removing a job shared page. }
      jsc$ti_recal_sje_s0 = 108,              { Recalculate the sje -- S0 state. }
```



```
      jsc$ti_recal_sje_s2 = 109,              { Recalculate the sje -- S2 state. }

      jsc$ti_riop_relinked = 110,             { Pages relinked into JWS from swapped error queue.}
      jsc$ti_riop_mem_freed = 111,            { IO error page belonging to a job that had freed memory.}
      jsc$ti_riop_m_bit_reset = 112,          { Pages that needed to have M bit set.}
      jsc$ti_riop_init = 113,                 { IO error occurred on an initial write.}

      jsc$ti_no_ajlo_mtr_swapin = 114,




      jsc$ti_max_index = 150;

?? FMT (FORMAT := ON) ??

{  Global variables defined by this module.

  VAR
    osv$debug: [XREF] array [0 .. 15] of integer,
    jsv$swap_trace: [XDCL] array [jsc$ti_min_index .. jsc$ti_max_index] of integer,

{  Define variable used to serialize job swapper access to the IJL when necessary.  Check calls
{  to tmp$set_lock and tmp$clear_lock for these cases.

    jsv$ijl_serial_lock: [XDCL] tmt$ptl_lock := [FALSE, 0],
    jsv$write_stale_pages: [XDCL, #GATE] boolean := FALSE,
    jsv$swap_file_page_count: [XDCL, #GATE] jst$swap_file_page_count,
    jsv$ijl_swap_queue_list: [XDCL, #GATE] jst$ijl_swap_queue_list :=
          [[[0, 0], [0, 0], 0], [[0, 0], [0, 0], 0], [[0, 0], [0, 0], 0], [[0, 0], [0, 0], 0], [[0, 0],
          [0, 0], 0], [[0, 0], [0, 0], 0]],
    jsv$enable_swap_file_statistics: [XDCL, #GATE] boolean := FALSE,
    jsv$enable_swap_resident: [XDCL, #GATE] boolean := TRUE,
    jsv$enable_swap_resident_no_io: [XDCL, #GATE] boolean := TRUE,
    jsv$enable_debug_code: [XDCL, #GATE] boolean := FALSE,
    jsv$halt_on_swapin_failure: [XDCL, #GATE] boolean := FALSE,
    jsv$free_working_set_on_swapout: [XDCL, #GATE] boolean := FALSE,
    jsv$max_pages_first_swap_task: [XDCL, #GATE] integer := 65536,
    jsv$maximum_pages_to_swap: [XDCL, #GATE] integer := 65536,
    jsv$pages_needed_for_sfd: [XDCL] integer := 0,
    jsv$swap_state_statistics: [XDCL, #GATE] jst$swap_state_statistics,
    jsv$swap_file_statistics: [XDCL, #GATE] jst$swap_file_statistics :=
          [[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 0],
    jsv$swapped_page_entry_size: [XDCL, #GATE] 0 .. 0ff(16) := 82, {deadstart init resets exactly}
    syv$allow_jr_test: [XDCL, #GATE] boolean := FALSE,
    syv$test_jr_system: [XDCL, #GATE] syt$test_jr_set := $syt$test_jr_set [];


{
{  Global type definitions defined by this module.
{

  TYPE
    cybil_pointer_trick = record
```

NOS/VE js : monitor mode job swapper
Global Declarations Declared by This Module

```
0  8232        case pointer_type: 0 .. 1 of
0  8233        = 0 =
0  8234          sfd_p: ^jst$swap_file_descriptor,
0  8235        = 1 =
0  8236          pva: ost$pva,
0  8237        casend,
0  8238      recend;
0  8239
0  8240    VAR
0  8241      kt: packed record
0  8242        case boolean of
0  8243        = TRUE =
0  8244          s: string (5),
0  8245        = FALSE =
0  8246          f1: 0 .. 0fffff(16),
0  8247          f2: 0 .. 0fff(16),
0  8248        casend,
0  8249      recend;
0  8250
0  8251
0  8252    PROCEDURE [INLINE] trace
0  8253      (     trace_index: jsc$ti_min_index .. jsc$ti_max_index;
0  8254            j: integer);
0  8255
0  8256      jsv$swap_trace [trace_index] := jsv$swap_trace [trace_index] + j;
0  8257    PROCEND trace;
0  8258
0  8259
0  8260
```


NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

```
0  8262
0  8263    PROCEDURE advance_swap
4  8264      (     ijl_ordinal: jmt$ijl_ordinal;
4  8265            ijle_p: ^jmt$initiated_job_list_entry;
4  8266        VAR set_polling_event: boolean;
4  8267        VAR status: syt$monitor_status);
4  8268
4  8269 {
4  8270 {       The purpose of this procedure is to advance the swap as far as it can go without
4  8271 {   waiting.  The swap is advanced until abnormal status is returned or a wait to complete
4  8272 {   condition is encountered.  If next_swap_status <> jmc$iss_null then that is moved
4  8273 {   to swap_status and another cycle is taken through the advance swap, current swap status
4  8274 {   is processed first however.  NEXT_SWAP_STATUS is used to indicate that a swap wait state
4  8275 {   has completed and advancing the swap should continue.  NEXT_SWAP_STATUS is set in the
4  8276 {   procedures that can be entered asynchronously in this module.
4  8277 {
4  8278 {   NOTE:
4  8279 {       Abnormal status is returned only for those conditions that abort the swap.
4  8280 {
4  8281 {       Mmv$reassignable_page_frames must be maintained.  Swapped_io_not_initiated and
4  8282 {       swapped_io_cannot_initiate contains the job queues page count.  Soon includes the
4  8283 {       job queues plus the SFD page count.
4  8284 {
4  8285
4  8286      VAR
4  8287        change_swap_direction: boolean,
4  8288        initiate_swapout_io: boolean,
4  8289        job_page_count: mmt$page_frame_index,
4  8290        last_swap_status: jmt$ijl_swap_status,
4  8291        pages_removed: mmt$page_frame_index,
4  8292        queue_id: mmt$job_page_queue_index,
4  8293        total_swapped_page_count: 0 .. osc$max_page_frames;
4  8294
4  8295
4  8296      IF ijle_p^.swap_queue_link.queue_id <> jsc$isqi_swapping THEN
E  8297        mtp$error_stop ('JS - advance_swap called for job not in swapping queue.');
2E 8298      IFEND;
2E 8299
2E 8300      status.normal := TRUE;
2E 8301      set_polling_event := FALSE;
2E 8302      last_swap_status := ijle_p^.swap_status;
2E 8303
2E 8304      WHILE status.normal DO
44 8305        CASE ijle_p^.swap_status OF
D8 8306
D8 8307        = jmc$iss_executing = {  R }
D8 8308
D8 8309          IF ijle_p^.entry_status > jmc$ies_swapped_in THEN
E2 8310            mtp$error_stop ('JS -- bad swap status - swapout executing job');
106 8311          ELSE
106 8312
106 8313 {  Cover the case where may go through the advance swap loop one time after job has been swapped in.
106 8314
106 8315            trace (jsc$ti_swapping_queue_and_exec, 1);
106 8316            RETURN;
```

NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

```
116  8317          IFEND;
11A  8318
11A  8319        = jmc$iss_job_idle_tasks_complete = { TJ }
11A  8320
11A  8321          IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
124  8322            trace (jsc$ti_sif_idled_tasks_comp, 1);
124  8323            ijle_p^.next_swap_status := jmc$iss_null;
124  8324            restart_idled_tasks (ijl_ordinal, ijle_p);
29A  8325            RETURN;
2A0  8326          ELSE
2A0  8327            jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
2B4  8328            calculate_swapped_pages (ijle_p);
2E4  8329            jsv$swap_file_page_count.swap_count := jsv$swap_file_page_count.swap_count + 1;
2E4  8330            jsv$swap_file_page_count.page_count := jsv$swap_file_page_count.page_count +
2E4  8331                 ijle_p^.swap_data.swapped_job_page_count;
2E4  8332
2E4  8333            initiate_swapout_io := ((mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon) <=
340  8334                 jmv$long_wait_swap_threshold) OR NOT jsv$enable_swap_resident_no_io;
340  8335
340  8336            IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
34A  8337              trace (jsc$ti_cd_idle_task_complete, 1);
34A  8338              swapin_before_io (ijl_ordinal, ijle_p);
364  8339              RETURN;
36A  8340            ELSEIF NOT initiate_swapout_io THEN
36E  8341              jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_not_init);
388  8342              advance_swap_state (ijle_p, jmc$iss_swapped_no_io);
39C  8343
39C  8344 { Recheck swap direction.  There is a timing problem here; direction can change just after it is checked
39C  8345 { above, and the job sits in the SO queue for two minutes before advancing.
39C  8346
39C  8347              IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
3A6  8348                jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
3BE  8349                trace (jsc$ti_cd_idle_task_complete_2, 1);
3CC  8350              ELSE
3CC  8351                RETURN;
3CE  8352              IFEND;
3D4  8353
3D4  8354            ELSE
3D4  8355              advance_swap_state (ijle_p, jmc$iss_flush_am_pages);
3E8  8356            IFEND;
3EC  8357          IFEND;
3F4  8358
3F4  8359        = jmc$iss_swapped_no_io = { SO }
3F4  8360
3F4  8361          IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
3FE  8362            swapin_before_io (ijl_ordinal, ijle_p);
40E  8363            RETURN;
414  8364
414  8365          ELSE
414  8366            jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
42C  8367            advance_swap_state (ijle_p, jmc$iss_flush_am_pages);
44C  8368          IFEND;
454  8369
454  8370        = jmc$iss_flush_am_pages = { FA }
454  8371
```


NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

```
454  8372          flush_am_pages_to_disk (ijl_ordinal, ijle_p);
464  8373          calculate_sfd_length (ijle_p);
472  8374*         advance_swap_state (ijle_p, jmc$iss_allocate_swap_file);
48A  8375
48A  8376        = jmc$iss_allocate_swap_file = { AF }
48A  8377
48A  8378          IF ijle_p^.swap_data.swapping_io_error <= ioc$allocate_file_space THEN
492  8379            allocate_swap_file (ijle_p, status);
4A6  8380            IF NOT status.normal THEN
4AE  8381              IF status.condition = dme$transient_error THEN
4BE  8382                advance_swap_state (ijle_p, jmc$iss_wait_allocate_swap_file);
4D2  8383                set_polling_event := TRUE;
4DC  8384              ELSE
4DC  8385                ijle_p^.swap_data.swapping_io_error := ioc$allocate_file_space;
4DC  8386                advance_swap_state (ijle_p, jmc$iss_job_allocate_swap_file);
4F6  8387                jmp$activate_job_mode_swapper;
4FE  8388              IFEND;
4FE  8389              status.normal := TRUE;
4FE  8390              RETURN;
508  8391            ELSE
508  8392              mmv$reassignable_page_frames.swapout_io_not_initiated :=
508  8393                   mmv$reassignable_page_frames.swapout_io_not_initiated -
508  8394                   ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
508  8395              mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon +
508  8396                   ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
508  8397
508  8398              ijle_p^.notify_swapper_when_io_complete := TRUE;
508  8399              IF ijle_p^.inhibit_swap_count <> 0 THEN
53A  8400                advance_swap_state (ijle_p, jmc$iss_wait_job_io_complete);
54E  8401                RETURN;
554  8402              ELSE
554  8403                ijle_p^.notify_swapper_when_io_complete := FALSE;
554  8404                advance_swap_state (ijle_p, jmc$iss_job_io_complete);
56C  8405              IFEND;
56E  8406            IFEND;
572  8407          ELSE
572  8408
572  8409 { The swap file encountered an error on a previous swapout.  Call job mode swapper to try to
572  8410 { reassign or reallocate the swap file.
572  8411
572  8412            advance_swap_state (ijle_p, jmc$iss_job_allocate_swap_file);
586  8413            jmp$activate_job_mode_swapper;
58E  8414            RETURN;
590  8415          IFEND;
59A  8416
59A  8417        = jmc$iss_job_io_complete = { JC }
59A  8418
59A  8419 { Verify that page queue counts are the same; if io completed abnormally the page queue counts
59A  8420 { may be differrent.  The swap file descriptor needs to be re-allocated. Swapout_io_not_initiated
59A  8421 { and soon needs to be updated.
59A  8422
59A  8423          IF (ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_io_error] <>
5B2  8424               ijle_p^.job_page_queue_list [mmc$pq_job_io_error].count) OR
5B2  8425               (ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] <>
5B2  8426               ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count) THEN
```

```
NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

     5B2  8427
     5B2  8428            trace (jsc$ti_page_q_counts_different, 1);
     5B2  8429            mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
     5B2  8430                  ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
     5B2  8431            calculate_swapped_pages (ijle_p);
     608  8432            calculate_sfd_length (ijle_p);
     62A  8433            advance_swap_state (ijle_p, jmc$iss_allocate_swap_file);
     642  8434          ELSE
     642  8435            advance_swap_state (ijle_p, jmc$iss_allocate_sfd);
     664  8436          IFEND;
     66C  8437
     66C  8438        = jmc$iss_allocate_sfd = { AD }
     66C  8439
     66C  8440          assign_pages_for_sfd (ijle_p, ijl_ordinal, jsc$sd_out, status);
     688  8441          IF NOT status.normal AND (status.condition = mme$no_free_pages) THEN
     6A0  8442
     6A0  8443 { Try freeing enough pages from the shared queue for the sfd and try to allocate the sfd again.  If there
     6A0  8444 { still are not enough free pages then cause mmp$periodic_call to be called to do some aging.
     6A0  8445
     6A0  8446            status.normal := TRUE;
     6A0  8447            trace (jsc$ti_dump_shared_q_for_sfd, 1);
     6A0  8448            mmp$dump_shared_queue (ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count);
     6C8  8449            assign_pages_for_sfd (ijle_p, ijl_ordinal, jsc$sd_out, status);
     6DC  8450            IF NOT status.normal THEN
     6E4  8451              status.normal := TRUE;
     6E4  8452              jsv$pages_needed_for_sfd := jsv$pages_needed_for_sfd +
     6E4  8453                    ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
     6E4  8454              mmp$nudge_periodic_call;
*WARN*    8455              advance_swap_state (ijle_p, jmc$iss_wait_allocate_sfd);
     71E  8456              set_polling_event := TRUE;
     71E  8457              RETURN;
     726  8458            IFEND;
     72A  8459          ELSEIF NOT status.normal THEN
     732  8460            status.normal := TRUE;
     732  8461            advance_swap_state (ijle_p, jmc$iss_wait_allocate_sfd);
     746  8462            set_polling_event := TRUE;
     746  8463            RETURN;
     750  8464          IFEND;
     750  8465
     750  8466 { When the job was last swapped in and the old swap file descriptor freed, the IJL.PURGE_MAP_TIMESTAMP
     750  8467 { was set equal to the value of the free running clock. The page map must be purged if it has not been
     750  8468 { purged since that time. If the map is NOT purged, references to the SFD may use the OLD page frames
     750  8469 { that were assigned at the PREVIOUS swapin. Purging of the map has been delayed since it will usually
     750  8470 { NOT be required at this point since something else will have purged the map.
     750  8471
     750  8472          mmp$conditional_purge_all_map (ijle_p^.sfd_purge_timestamp);
     782  8473
     782  8474 { XCB access will be inhibited from now on.  Set the timestamp now for reassigning ASIDs.
     782  8475
     782  8476          ijle_p^.swap_data.asid_reassigned_timestamp := #FREE_RUNNING_CLOCK (0);
     788  8477          advance_swap_state (ijle_p, jmc$iss_initiate_swapout_io);
     7AA  8478
     7AA  8479        = jmc$iss_swapped_io_cannot_init = { SD }
     7AA  8480
     7AA  8481          mmv$reassignable_page_frames.swapout_io_cannot_initiate :=
```



```
NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

     7AA  8482                mmv$reassignable_page_frames.swapout_io_cannot_initiate -
     7AA  8483                ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
     7AA  8484          IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
     7CC  8485            swapin_after_io (ijl_ordinal, ijle_p);
     7DC  8486            RETURN;
     7E2  8487          ELSE
     7E2  8488            mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon +
     7E2  8489                  ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
     7E2  8490            advance_swap_state (ijle_p, jmc$iss_allocate_sfd);
     802  8491          IFEND;
     802  8492          trace (jsc$ti_advance_from_cannot_init, 1);
     818  8493
     818  8494        = jmc$iss_initiate_swapout_io = { OS }
     818  8495
     818  8496          total_swapped_page_count := ijle_p^.swap_data.swapped_job_page_count +
     818  8497                ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
     818  8498          job_swapping_io (ijl_ordinal, ijle_p, ijle_p^.swap_data.swap_file_sfid, ioc$swap_out,
     848  8499                total_swapped_page_count, ijle_p^.swap_io_control, status);
     848  8500
     848  8501          IF NOT status.normal THEN
     852  8502            IF status.condition = ioe$unit_disabled THEN
     862  8503              trace (jsc$ti_init_swapout_io_error, 1);
     862  8504              ijle_p^.swap_data.swapping_io_error := ioc$unrecovered_error_unit_down;
     862  8505              process_io_error_on_swapout (ijl_ordinal, ijle_p, set_polling_event);
     88E  8506
     88E  8507            ELSE
     88E  8508              set_polling_event := TRUE;
     88E  8509              advance_swap_state (ijle_p, jmc$iss_wait_swapout_io_init);
     8A4  8510            IFEND;
     8A4  8511
     8A4  8512            status.normal := TRUE;
     8A4  8513            RETURN;
     8AE  8514          ELSE
     8AE  8515            advance_swap_state (ijle_p, jmc$iss_swapout_io_initiated);
     8C0  8516          IFEND;
     8CE  8517
     8CE  8518        = jmc$iss_swapout_io_complete = { OC }
     8CE  8519
     8CE  8520          IF ijle_p^.swap_data.swapping_io_error <> ioc$no_error THEN
     8D6  8521            ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
     8D6  8522            IF ijle_p^.swap_data.swapping_io_error = ioc$unrecovered_error_unit_down THEN
     8E0  8523              trace (jsc$ti_swapout_disk_down, 1);
     8E0  8524              advance_swap_state (ijle_p, jmc$iss_initiate_swapout_io);
     904  8525            ELSE
     904  8526              trace (jsc$ti_swapout_io_error, 1);
     904  8527              process_io_error_on_swapout (ijl_ordinal, ijle_p, set_polling_event);
     924  8528              RETURN;
     926  8529            IFEND;
     92A  8530
     92A  8531          ELSE
     92A  8532            free_swap_file_descriptor (ijle_p, ijl_ordinal);
     A26  8533
     A26  8534            IF (mmv$reassignable_page_frames.now < mmv$min_avail_pages) OR NOT jsv$enable_swap_resident THEN
     A46  8535              last_swap_status := jmc$iss_swapout_io_complete;
     A46  8536              advance_swap_state (ijle_p, jmc$iss_free_swapped_memory);
```

NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

```
A64  8537            ELSE
A64  8538
A64  8539 {  Increment reassignable page frames NOW and decrement SOON.
A64  8540
A64  8541              mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
A64  8542                    ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
A64  8543              mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now +
A64  8544                    ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
A64  8545              advance_swap_state (ijle_p, jmc$iss_swapped_io_complete);
A94  8546              jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_completed);
ABO  8547
ABO  8548 { Recheck the swap direction.
ABO  8549 { On a dual CPU system, the swap direction may have changed (because a
ABO  8550 { ready task was processed in tmp$switch_task) just as the swap status
ABO  8551 { was advanced to swapped_io_completed.
ABO  8552
ABO  8553              IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
ABA  8554                jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
AD2  8555                trace (jsc$ti_cd_to_in_at_S2, 1);
AE0  8556              ELSE
AE0  8557                RETURN;
AE2  8558              IFEND;
AE6  8559            IFEND;
AEA  8560          IFEND;
AF2  8561
AF2  8562        = jmc$iss_swapped_io_complete = { S2 }
AF2  8563
AF2  8564          IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
AFC  8565            ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
AFC  8566            mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now -
AFC  8567                  ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
AFC  8568            swapin_after_io (ijl_ordinal, ijle_p);
B2A  8569            RETURN;
B30  8570          ELSE
B30  8571            last_swap_status := jmc$iss_swapped_io_complete;
B30  8572            advance_swap_state (ijle_p, jmc$iss_free_swapped_memory);
B50  8573          IFEND;
B58  8574
B58  8575        = jmc$iss_free_swapped_memory = { FM }
B58  8576
B58  8577          free_swapped_jobs_mm_resources (ijle_p, ijl_ordinal, last_swap_status);
B6A  8578          advance_swap_state (ijle_p, jmc$iss_swapout_complete);
B7A  8579          jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
BAO  8580
BAO  8581 { Do not return yet; need to loop through again to check swap direction.
BAO  8582 { On a dual CPU system, the swap direction may have changed (because a
BAO  8583 { ready task was processed in tmp$switch_task) just as the swap status
BAO  8584 { was advanced to swapout_complete.
BAO  8585
BAO  8586
BAO  8587        = jmc$iss_swapout_complete = {  S }
BAO  8588
BAO  8589          IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
BAA  8590
BAA  8591 { Check if the job is in the swapping queue; because of dual CPU timing, the
```


NOS/VE js : monitor mode job swapper
ADVANCE_SWAP

```
BAA  8592 { job may have been relinked to the swapped out queue after the job was readied
BAA  8593 { and direction set to IN.
BAA  8594
BAA  8595            IF ijle_p^.swap_queue_link.queue_id <> jsc$isqi_swapping THEN
BB2  8596              jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
BCA  8597              trace (jsc$ti_cd_to_in_at_S, 1);
BD8  8598            IFEND;
BD8  8599
BD8  8600 { Add up the swapped job page count again.  If job shared pages were removed from the
BD8  8601 { job's working set while the job was in the swapped_io_complete (S2) state, the
BD8  8602 { swapped job page count was changed to reflect the new (lower) working set size.
BD8  8603 { However, all pages that were written out need to be read back in, so the swapped
BD8  8604 { job page count needs to be reset to the total written out.
BD8  8605
BD8  8606            job_page_count := 0;
BD8  8607            FOR queue_id := LOWERVALUE (mmt$job_page_queue_index) TO UPPERVALUE (mmt$job_page_queue_index) DO
BE8  8608              job_page_count := job_page_count + ijle_p^.swap_data.swapped_job_entry.
BE8  8609                    job_page_queue_count [queue_id];
BE8  8610            FOREND;
BF6  8611            ijle_p^.swap_data.swapped_job_page_count := job_page_count;
BF6  8612            advance_swap_state (ijle_p, jmc$iss_swapin_requested);
C14  8613          ELSE
C14  8614            RETURN;
C16  8615          IFEND;
C1E  8616
C1E  8617        = jmc$iss_swapin_requested = { IR }
C1E  8618
C1E  8619          ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
C1E  8620          claim_pages_for_swap_in (ijl_ordinal, ijle_p, status);
C38  8621
C38  8622          IF NOT status.normal THEN
C40  8623            advance_swap_state (ijle_p, jmc$iss_swapout_complete);
C50  8624            jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
C6C  8625            jmp$reset_job_to_swapped_out (ijl_ordinal);
C7C  8626            RETURN;
C82  8627          ELSE
C82  8628            advance_swap_state (ijle_p, jmc$iss_swapin_resource_claimed);
C96  8629          IFEND;
CAO  8630
CAO  8631
CAO  8632        = jmc$iss_swapin_resource_claimed = { IS }
CAO  8633
CAO  8634          total_swapped_page_count := ijle_p^.swap_data.swapped_job_page_count +
CAO  8635                ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
CAO  8636          job_swapping_io (ijl_ordinal, ijle_p, ijle_p^.swap_data.swap_file_sfid, ioc$swap_in,
CD8  8637                total_swapped_page_count, ijle_p^.swap_io_control, status);
CD8  8638          IF NOT status.normal THEN
CE2  8639            IF status.condition = ioe$unit_disabled THEN
CF2  8640              trace (jsc$ti_init_swapin_io_error, 1);
CF2  8641              process_io_error_on_swapin (ijl_ordinal, ijle_p);
D18  8642              RETURN;
D1E  8643
D1E  8644            ELSE
D1E  8645              advance_swap_state (ijle_p, jmc$iss_wait_swapin_io_init);
D30  8646              set_polling_event := TRUE;
```

```
D30  8647              status.normal := TRUE;
D3A  8648            IFEND;
D3C  8649          ELSE
D3C  8650            tmv$swapin_in_progress := tmv$swapin_in_progress + 1;
D3C  8651            advance_swap_state (ijle_p, jmc$iss_swapin_io_initiated);
D5C  8652          IFEND;
D5C  8653
D5C  8654          RETURN;
D62  8655
D62  8656        = jmc$iss_swapin_io_complete = [ IC ]
D62  8657
D62  8658          tmv$swapin_in_progress := tmv$swapin_in_progress - 1;
D62  8659          IF ijle_p^.swap_data.swapping_io_error <> ioc$no_error THEN
D78  8660            IF ijle_p^.swap_data.swapping_io_error = ioc$unrecovered_error_unit_down THEN
D7E  8661              trace (jsc$ti_swapin_disk_down, 1);
D7E  8662              advance_swap_state (ijle_p, jmc$iss_swapin_resource_claimed);
D9E  8663              ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
DA8  8664            ELSE
DA8  8665              trace (jsc$ti_swapin_io_error, 1);
DA8  8666              process_io_error_on_swapin (ijl_ordinal, ijle_p);
DC6  8667              RETURN;
DC8  8668            IFEND;
DCC  8669
DCC  8670          ELSEIF ijle_p^.entry_status > jmc$ies_swapped_in THEN
DD6  8671
DD6  8672 {  Abort the swapin, received request to swap job out again.
DD6  8673
DD6  8674            trace (jsc$ti_swapin_int_by_swapout, 1);
DD6  8675            free_swapped_jobs_mm_resources (ijle_p, ijl_ordinal, jmc$iss_swapin_io_complete);
DFA  8676            ?IF debug = TRUE THEN
     8677              IF syv$allow_jr_test THEN
     8678                IF syc$tjr_mtr_fsjmmr IN syv$test_jr_system THEN
     8679                  mtp$error_stop ('JOB RECOVERY TEST');
     8680                IFEND;
     8681              IFEND;
DFA  8682            ?IFEND
DFA  8683            advance_swap_state (ijle_p, jmc$iss_swapout_complete);
E0A  8684            jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
E26  8685            jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
E3A  8686            RETURN;
E40  8687          ELSE
E40  8688
E40  8689 {  Restore memory manager tables for job image read from mass storage, update ASID's in job's
E40  8690 {  segment tables and the system file table.  Swap status is advanced to executing if successful.
E40  8691
E40  8692            reset_swapped_job_mm_tables (ijl_ordinal, ijle_p, ijle_p^.swap_data.swapped_job_entry,
E62  8693                  ijle_p^.sfd_p, status);
E62  8694            IF NOT status.normal THEN
E6C  8695              IF status.condition = jse$pt_full_on_swap_in THEN
E7C  8696                advance_swap_state (ijle_p, jmc$iss_swapout_complete);
E90  8697                jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
EAC  8698                jmp$reset_job_to_swapped_out (ijl_ordinal);
EBC  8699                jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
ED2  8700              ELSEIF status.condition = jse$bad_swap_file_data_detected THEN
EDA  8701                jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
```



```
EF6  8702                free_swapped_jobs_mm_resources (ijle_p, ijl_ordinal, jmc$iss_swapin_io_complete);
F0A  8703                advance_swap_state (ijle_p, jmc$iss_swapout_complete);
F1A  8704                jmp$recognize_job_dead (ijl_ordinal);
F2A  8705                jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
F40  8706              ELSE
F40  8707                mtp$error_stop ('JS - unexpected status on reset MM tables');
F5C  8708              IFEND;
F5C  8709            IFEND;
F5C  8710
F5C  8711            RETURN;
F5E  8712          IFEND;
F66  8713
F66  8714        ELSE
F66  8715
F66  8716 {  Process the unselected case, check if change in swap direction or if next swap status is set.
F66  8717
F66  8718          last_swap_status := ijle_p^.swap_status;
F66  8719          change_swap_direction := ((last_swap_status <= UPPERVALUE (jmt$swapout)) AND
FC6  8720                (last_swap_status >= LOWERVALUE (jmt$swapout)) AND
FC6  8721                (ijle_p^.entry_status < jmc$ies_swapped_out)) OR
FC6  8722                ((last_swap_status <= UPPERVALUE (jmt$swapin)) AND
FC6  8723                (last_swap_status >= LOWERVALUE (jmt$swapin)) AND (ijle_p^.entry_status > jmc$ies_swapped_in));
FC6  8724          IF change_swap_direction THEN
FCA  8725            IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
FD4  8726              trace (jsc$ti_swapout_int_by_swapin, 1);
FD4  8727              direction_changed_to_in (ijl_ordinal, ijle_p);
FF2  8728              RETURN;
FF8  8729            ELSE {direction is out}
FF8  8730              IF ijle_p^.next_swap_status = jmc$iss_swapin_io_complete THEN
1004  8731                advance_swap_state (ijle_p, ijle_p^.next_swap_status);
1016  8732                ijle_p^.next_swap_status := jmc$iss_null;
1020  8733              ELSE
1020  8734                mtp$error_stop ('JS--bad swap status-swapin changed direction');
1046  8735              IFEND;
104A  8736            IFEND;
104E  8737          ELSEIF (ijle_p^.next_swap_status <> jmc$iss_null) THEN
1056  8738            advance_swap_state (ijle_p, ijle_p^.next_swap_status);
1068  8739            ijle_p^.next_swap_status := jmc$iss_null;
1074  8740          ELSE
1074  8741            RETURN;
1076  8742          IFEND;
107A  8743        CASEND;
107A  8744      WHILEND;
1082  8745
1082  8746    PROCEND advance_swap;
```

NOS/VE js : monitor mode job swapper
ADVANCE_SWAP_STATE

```
        o  8748
        o  8749 {
        o  8750 {  This procedure is responsible for updating the job swap status in the IJL. In addition
        o  8751 {  to maintaining job status, this procedure also keep statistics on the total amount of
        o  8752 {  time spent in a state and the new state that was entered from the current state. This is
        o  8753 {  maintained in a 2-dimensional matrix as follows:
        o  8754 {
        o  8755 {
        o  8756 {                              new state
        o  8757 {                     xx xx xx xx xx xx xx xx        each element in the matrix contains:
        o  8758 {                     xx xx xx xx xx xx xx xx           count - number of transitions between states
        o  8759 {            old      xx xx xx xx xx xx xx xx           time -  total time spent in old state prior to
        o  8760 {            state    xx xx xx xx xx xx xx xx                   transition to new state
        o  8761 {                     xx xx xx xx xx xx xx xx
        o  8762 {
        o  8763
        o  8764
        o  8765
        o  8766    PROCEDURE advance_swap_state
        o  8767      (     ijle_p: ^jmt$initiated_job_list_entry;
        o  8768            new_swap_status: jmt$ijl_swap_status);
        o  8769
        o  8770      VAR
        o  8771        current_time: ost$free_running_clock,
        o  8772        delta_time: ost$free_running_clock,
        o  8773        old_swap_status: jmt$ijl_swap_status;
        o  8774
        o  8775      old_swap_status := ijle_p^.swap_status;
        4  8776      ijle_p^.last_swap_status := old_swap_status;
        4  8777      current_time := #FREE_RUNNING_CLOCK (0);
       14  8778      delta_time := current_time - ijle_p^.swap_data.timestamp;
       14  8779
       14  8780      jsv$swap_state_statistics [old_swap_status] [new_swap_status].
       14  8781            count := jsv$swap_state_statistics [old_swap_status] [new_swap_status].count + 1;
       14  8782
       14  8783      jsv$swap_state_statistics [old_swap_status] [new_swap_status].
       14  8784            total_time := jsv$swap_state_statistics [old_swap_status] [new_swap_status].total_time + delta_time;
       14  8785
       14  8786      IF delta_time > jsv$swap_state_statistics [old_swap_status] [new_swap_status].maximum_time THEN
       46  8787        IF delta_time > UPPERVALUE (jsv$swap_state_statistics [old_swap_status] [new_swap_status].
       4E  8788              maximum_time) THEN
       4E  8789          jsv$swap_state_statistics [old_swap_status] [new_swap_status].
       5A  8790                maximum_time := UPPERVALUE (jsv$swap_state_statistics [old_swap_status] [new_swap_status].
       5A  8791                maximum_time);
       5A  8792        ELSE
       5A  8793          jsv$swap_state_statistics [old_swap_status] [new_swap_status].maximum_time := delta_time;
       5E  8794        IFEND;
       5E  8795      IFEND;
       5E  8796
       5E  8797      ijle_p^.swap_data.timestamp := current_time;
       5E  8798      ijle_p^.swap_status := new_swap_status;
       5E  8799
       5E  8800    PROCEND advance_swap_state;
```


NOS/VE js : monitor mode job swapper
ALLOCATE_SWAP_FILE

```
        o  8802
        o  8803 { PURPOSE:
        o  8804 {    This procedure determines if the swap file is large enough, and allocates more space if necessary.
        o  8805
        o  8806    PROCEDURE allocate_swap_file
        o  8807      (     ijle_p: ^jmt$initiated_job_list_entry;
        o  8808       VAR status: syt$monitor_status);
        o  8809
        o  8810      VAR
        o  8811        fde_p: gft$locked_file_desc_entry_p,
        o  8812        file_status: dmt$file_allocation_status,
        o  8813        ignore_aus_obtained: amt$file_byte_address,
        o  8814        ignore_overflow: boolean,
        o  8815        total_swapped_page_count: 0 .. osc$max_page_frames;
        o  8816
        o  8817      status.normal := TRUE;
        4  8818
        4  8819      total_swapped_page_count := ijle_p^.swap_data.swapped_job_page_count +
        4  8820            ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
        4  8821      IF total_swapped_page_count > ijle_p^.swap_data.swap_file_length_in_pages THEN
       1C  8822        gfp$mtr_get_locked_fde_p (ijle_p^.swap_data.swap_file_sfid, ijle_p, fde_p);
       9C  8823        dmp$allocate_file_space (fde_p, 0, total_swapped_page_count * osv$page_size - 1, sfc$no_limit,
       E2  8824              ignore_aus_obtained, ignore_overflow, file_status);
       E2  8825        trace (jsc$ti_allocate_swap_file, 1);
       E2  8826
       E2  8827        CASE file_status OF
      108  8828        = dmc$fas_file_allocated =
      108  8829          ijle_p^.swap_data.swap_file_length_in_pages := total_swapped_page_count;
      108  8830          fde_p^.eoi_byte_address := total_swapped_page_count * osv$page_size;
      108  8831          fde_p^.flags.eoi_modified := TRUE;
      122  8832
      122  8833        = dmc$fas_job_mode_work_required =
      122  8834          trace (jsc$ti_allocate_swap_file_jm, 1);
      122  8835          mtp$set_status_abnormal ('JS', jse$swap_file_not_allocated, status);
      13E  8836
      13E  8837        = dmc$fas_temp_reject =
      13E  8838          trace (jsc$ti_dm_transient_error, 1);
      13E  8839          mtp$set_status_abnormal ('DM', dme$transient_error, status);
      15A  8840
      15A  8841        ELSE
      15A  8842          mtp$error_stop ('JS - unexpected status from dmp$allocate_file_space');
      17A  8843        CASEND;
      17A  8844      IFEND;
      17A  8845
      17A  8846    PROCEND allocate_swap_file;
```

NOS/VE js : monitor mode job swapper
ASSIGN PAGES FOR SFD

```
      o  8848
      o  8849    PROCEDURE assign_pages_for_sfd
      o  8850      (     ijle_p: ^jmt$initiated_job_list_entry;
      o  8851            ijl_ordinal: jmt$ijl_ordinal;
      o  8852            direction: jst$swap_direction;
      o  8853       VAR status: syt$monitor_status);
      o  8854
      o  8855 {
      o  8856 {     This procedure assigns the pages for the swap file descriptor in job fixed of the job being
      o  8857 {  swapped in or out.
      o  8858 {
      o  8859
      o  8860      VAR
      o  8861        ajlo: jmt$ajl_ordinal,
      o  8862        jcb_p: ^jmt$job_control_block,
      o  8863        ptr_to_sfd: ^^cell,
      o  8864        rma: integer,
      o  8865        sfd_cell_p: ^cell,
      o  8866        sfd_offset: integer,
      o  8867        sfd_page_count: 0 .. osc$max_page_frames,
      o  8868        total_swapped_page_count: 0 .. osc$max_page_frames,
      o  8869        try: integer;
      o  8870
      o  8871      sfd_page_count := ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
      4  8872      total_swapped_page_count := ijle_p^.swap_data.swapped_job_page_count + sfd_page_count;
      4  8873      sfd_offset := osv$page_size * 3713 + 10000000(16);
      4  8874      ajlo := ijle_p^.ajl_ordinal;
      4  8875
      4  8876      IF direction = jsc$sd_out THEN
     2E  8877        jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$lock_ajl, TRUE {must assign} , ajlo,
     5E  8878              status);
     5E  8879      IFEND;
     5E  8880
     5E  8881 { Allocate pages at the end JOB FIXED for the SFD. If the request is rejected because of page
     5E  8882 {table full, try several times before giving up - use a different PVA on each try.
     5E  8883
     5E  8884      try := 10;
     5E  8885
     5E  8886      REPEAT
     62  8887        sfd_offset := sfd_offset + osv$page_size * 471;
     62  8888        sfd_cell_p := #ADDRESS (1, ajlo + mtc$job_fixed_segment, sfd_offset);
     62  8889        mmp$assign_page_to_monitor (sfd_cell_p, sfd_page_count, FALSE, status);
     9E  8890        IF NOT status.normal AND ((status.condition <> mme$page_table_full) OR (try = 0)) THEN
     BA  8891          IF direction = jsc$sd_out THEN
     BE  8892            jmp$free_ajl_entry (ijle_p, jmc$lock_ajl);
     D6  8893          IFEND;
     D6  8894          RETURN;
     D8  8895        IFEND;
     D8  8896        try := try - 1;
     D8  8897      UNTIL status.normal;
     E2  8898
     E2  8899 { Update the IJL with SFD descriptive information. Set up the swap io control block
     E2  8900 { with the information required for build_lock_rma_list.
     E2  8901
     E2  8902      ptr_to_sfd := #LOC (ijle_p^.sfd_p);
```


NOS/VE js : monitor mode job swapper
ASSIGN PAGES FOR SFD

```
     E6  8903      i#build_adaptable_array_ptr (1, ajlo + mtc$job_fixed_segment, sfd_offset,
    11E  8904            #SIZE (jst$swapped_page_descriptor) * total_swapped_page_count, 0,
    11E  8905            #SIZE (jst$swapped_page_descriptor), #LOC (ptr_to_sfd^));
    11E  8906      i#real_memory_address (sfd_cell_p, rma);
    136  8907      ijle_p^.swap_io_control.swap_file_descriptor_pfti := rma DIV osv$page_size;
    136  8908      IF direction = jsc$sd_out THEN
    148  8909
    148  8910 { Set up jcb with the swapped_job_entry.  It is used by job recovery.
    148  8911
    148  8912        jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ajlo, 0);
    148  8913        jcb_p^.swapped_job_entry := ijle_p^.swap_data.swapped_job_entry;
    168  8914
    168  8915        ijle_p^.sfd_p^.swapped_job_entry := ijle_p^.swap_data.swapped_job_entry;
    176  8916        ijle_p^.sfd_p^.ijl_entry := ijle_p^;
    18C  8917
    18C  8918        jmp$free_ajl_entry (ijle_p, jmc$lock_ajl);
    1A4  8919      IFEND;
    1A4  8920
    1A4  8921      mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon + sfd_page_count;
    1A4  8922
    1A4  8923    PROCEND assign_pages_for_sfd;
      o  8924
```

NOS/VE js : monitor mode job swapper
CALCULATE_SFD_LENGTH

```
        o  8926
        o  8927    PROCEDURE calculate_sfd_length
        o  8928      (     ijle_p: ^jmt$initiated_job_list_entry);
        o  8929
        o  8930  {
        o  8931  {  This procedure calculates the swap file descriptor length and set the value in the IJL.
        o  8932  {
        o  8933
        o  8934      VAR
        o  8935        job_page_count: 0 .. osc$max_page_frames,
        o  8936        sfd_p: ^jst$swap_file_descriptor,
        o  8937        sfd_page_count: 0 .. osc$max_page_frames;
        o  8938
        o  8939
        o  8940      job_page_count := ijle_p^.swap_data.swapped_job_page_count;
        4  8941
        4  8942  { Determine number of pages to allocate for the swap file descriptor.  The following algorithm makes a
        4  8943  { guess that 1 page is required, then iterates until the size is correct.
        4  8944
        4  8945      PUSH sfd_p: [0 .. 0]; {used to get size of a sfd with 1 entry}
       1A  8946      sfd_page_count := 1 + (#SIZE (sfd_p^) + #SIZE (jst$swapped_page_descriptor) * job_page_count - 1) DIV
       1A  8947            osv$page_size;
       1A  8948      WHILE ((job_page_count + sfd_page_count - 1) * #SIZE (jst$swapped_page_descriptor) + #SIZE (sfd_p^)) >
       48  8949            (sfd_page_count * osv$page_size) DO
       48  8950        sfd_page_count := sfd_page_count + 1;
       48  8951      WHILEND;
       66  8952
       66  8953      ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count := sfd_page_count;
       66  8954
       66  8955    PROCEND calculate_sfd_length;
```


NOS/VE js : monitor mode job swapper
CALCULATE_SWAPPED_PAGES

```
        o  8958
        o  8959    PROCEDURE [INLINE] calculate_swapped_pages
        o  8960      (     ijle_p: ^jmt$initiated_job_list_entry);
        o  8961
        o  8962      VAR
        o  8963        job_page_count: mmt$page_frame_index,
        o  8964        job_queue_id: mmt$job_page_queue_index;
        o  8965
        o  8966      job_page_count := 0;
        o  8967
        o  8968      FOR job_queue_id := LOWERVALUE (mmt$job_page_queue_index) TO UPPERVALUE (mmt$job_page_queue_index) DO
        o  8969        job_page_count := job_page_count + ijle_p^.job_page_queue_list [job_queue_id].count;
        o  8970        ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [job_queue_id] :=
        o  8971              ijle_p^.job_page_queue_list [job_queue_id].count;
        o  8972      FOREND;
        o  8973
        o  8974      ijle_p^.swap_data.swapped_job_page_count := job_page_count;
        o  8975
        o  8976      mmv$reassignable_page_frames.swapout_io_not_initiated :=
        o  8977            mmv$reassignable_page_frames.swapout_io_not_initiated + ijle_p^.swap_data.swapped_job_page_count -
        o  8978            ijle_p^.job_fixed_contiguous_pages;
        o  8979
        o  8980    PROCEND calculate_swapped_pages;
        o  8981
```

NOS/VE js : monitor mode job swapper
CLAIM_PAGES_FOR_SWAP_IN

```
      0  8984
      0  8985    PROCEDURE claim_pages_for_swap_in
      0  8986      (     ijl_ordinal: jmt$ijl_ordinal;
      0  8987            ijle_p: ^jmt$initiated_job_list_entry;
      0  8988       VAR status: syt$monitor_status);
      0  8989
      0  8990  {
      0  8991  {   The purpose of this procedure is to claim the number of pages needed to
      0  8992  { swap the job in.  The pages are linked in the proper queues at this time
      0  8993  { except the available modified pages are linked into the job working set
      0  8994  { queue.
      0  8995  {
      0  8996
      0  8997      VAR
      0  8998        ajl_ordinal: jmt$ajl_ordinal,
      0  8999        ast_index: mmt$ast_index,
      0  9000        temp_asti: mmt$ast_index,
      0  9001        asid: ost$asid,
      0  9002        aste_p: ^mmt$active_segment_table_entry,
      0  9003        queue: mmt$global_page_queue_index,
      0  9004        sum_shared: integer,
      0  9005        total_swapped_page_count: 0 .. osc$max_page_frames,
      0  9006        update_segnum_sfd_p: cybil_pointer_trick;
      0  9007
      0  9008      status.normal := TRUE;
      4  9009      total_swapped_page_count := ijle_p^.swap_data.swapped_job_page_count +
      4  9010            ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
      4  9011
      4  9012  {  Check if there is enough memory in the free and availble queues to swap this job in.
      4  9013
      4  9014      IF ((total_swapped_page_count >= mmv$reassignable_page_frames.now) OR
     34  9015            (total_swapped_page_count >= (mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon -
     34  9016            mmv$aggressive_aging_level_2))) THEN
     34  9017
     34  9018  { Raid the shared queue if there are enough pages in it to swapin the job.
     34  9019
     34  9020        sum_shared := 0;
     34  9021        FOR queue := mmc$pq_shared_first TO mmv$last_active_shared_queue DO
     48  9022          sum_shared := sum_shared + mmv$gpql [queue].pqle.count;
     48  9023        FOREND;
     5A  9024        IF (mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon + sum_shared -
     74  9025              mmv$aggressive_aging_level_2) > total_swapped_page_count THEN
     74  9026          trace (jsc$ti_dump_shared_queue, 1);
     74  9027          mmp$dump_shared_queue (total_swapped_page_count);
     92  9028        IFEND;
     92  9029
     92  9030  { If there is still not enough memory, RETURN bad status.
     92  9031
     92  9032        IF (total_swapped_page_count >= mmv$reassignable_page_frames.now) OR
     AA  9033              (total_swapped_page_count >= (mmv$reassignable_page_frames.now +
     AA  9034              mmv$reassignable_page_frames.soon - mmv$aggressive_aging_level_2)) THEN
     AA  9035          trace (jsc$ti_no_memory_for_swap_in, 1);
     AA  9036          mtp$set_status_abnormal ('JS', jse$not_enough_mem_for_swap_in, status);
     AA  9037          RETURN;
     CA  9038        IFEND;
```


NOS/VE js : monitor mode job swapper
CLAIM_PAGES_FOR_SWAP_IN

```
     CA  9039      IFEND;
     CA  9040
     CA  9041
     CA  9042  {   Reclaim the old job fixed ASID or assign a new one if the old one is in use.
     CA  9043
     CA  9044      asid := ijle_p^.job_fixed_asid;
     CA  9045
     CA  9046      mmp$asti (asid, ast_index);
     EE  9047      aste_p := ^mmv$ast_p^ [ast_index];
     EE  9048      IF (jmc$dsw_job_recovery IN ijle_p^.delayed_swapin_work) OR (aste_p^.ijl_ordinal <> ijl_ordinal) THEN
    122  9049        trace (jsc$ti_new_job_fixed_asid, 1);
    122  9050        mmp$assign_asid (asid, temp_asti, aste_p);
    154  9051        ijle_p^.job_fixed_asid := asid;
    160  9052      ELSE
    160  9053        trace (jsc$ti_reuse_job_fixed_asid, 1);
    160  9054        IF NOT aste_p^.in_use THEN
    17A  9055          trace (jsc$ti_reuse_job_fixed_asid_as, 1);
    17A  9056          mmp$assign_specific_asid (aste_p);
    194  9057        IFEND;
    194  9058      IFEND;
    194  9059      aste_p^ := mmv$initial_job_fixed_ast_entry;
    1A6  9060      aste_p^.ijl_ordinal := ijl_ordinal;
    1A6  9061
    1A6  9062  { Assign an ajl entry to the job.
    1A6  9063
    1A6  9064      jmp$assign_ajl_entry (asid, ijl_ordinal, jmc$swapping_ajl, FALSE (must assign) , ajl_ordinal, status);
    1D8  9065      IF NOT status.normal THEN
    1E0  9066        trace (jsc$ti_no_ajl_ord_for_swap_in, 1);
    1E0  9067        RETURN;
    1EC  9068      IFEND;
    1EC  9069
    1EC  9070      IF syv$perf_keypoints_enabled.swapping_keypoints THEN
    1F8  9071        kt.s := ijle_p^.system_supplied_name (16, 4);
    202  9072        #KEYPOINT (osk$performance, osk$m * kt.f1, ptk$swapin_job_name_1);
    216  9073        #KEYPOINT (osk$performance, osk$m * ((kt.f2 * 256) + ajl_ordinal), ptk$swapin_job_name_2);
    230  9074      IFEND;
    230  9075
    230  9076  { Assign new page frames for the job and swap file descriptor.
    230  9077
    230  9078      mmp$claim_pages_for_swapin (ijle_p^.swap_data.swapped_job_entry, aste_p, ijl_ordinal,
    25C  9079            ijle_p^.job_page_queue_list);
    25C  9080      assign_pages_for_sfd (ijle_p, ijl_ordinal, jsc$sd_in, status);
    26E  9081      IF NOT status.normal THEN
    278  9082        trace (jsc$ti_no_pages_for_sfd_on_si, 1);
    278  9083        mmp$free_memory_in_job_queues (ijle_p^.job_page_queue_list, TRUE, FALSE, FALSE);
    2A2  9084        jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
    2B8  9085        mmp$free_asid (asid, aste_p);
    2D4  9086        mtp$set_status_abnormal ('JS', jse$not_enough_mem_for_swap_in, status);
    2E4  9087      IFEND;
    2E4  9088
    2E4  9089    PROCEND claim_pages_for_swap_in;
```

NOS/VE js : monitor mode job swapper
COMPLETE_SWAPIN

```
        0  9092
        0  9093   PROCEDURE complete_swapin
        0  9094     (     ijl_ordinal: jmt$ijl_ordinal;
        0  9095           ijle_p: ^jmt$initiated_job_list_entry;
        0  9096           available_modified_page_count: 0 .. osc$max_page_frames);
        0  9097
        0  9098 {
        0  9099 {     The purpose of this procedure is to perform the tasks to complete the swapin of a job after
        0  9100 {   the memory manager tables have been restored.  This procedure sets the proper swap status and
        0  9101 {   relinks the job into the null swapping queue.
        0  9102 {
        0  9103
        0  9104     VAR
        0  9105       jcb_p: ^jmt$job_control_block;
        0  9106
        0  9107
        0  9108 {   Move pages back to the available modified queue if they belong there.
        0  9109
        0  9110     IF available_modified_page_count > 0 THEN
        8  9111       move_am_to_am (ijle_p, available_modified_page_count);
       20  9112     IFEND;
       20  9113
       20  9114     jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, 0);
       20  9115     jcb_p^.next_cyclic_aging_time := #FREE_RUNNING_CLOCK (0) + jcb_p^.next_cyclic_aging_time;
       3E  9116
       3E  9117     restart_idled_tasks (ijl_ordinal, ijle_p);
      1A8  9118
      1A8  9119     IF (available_modified_page_count > 0) THEN
      1AC  9120       mmp$replenish_free_queues (0);
      1BC  9121     IFEND;
      1BC  9122
      1BC  9123   PROCEND complete_swapin;
```


NOS/VE js : monitor mode job swapper
FLUSH_AM_PAGES_TO_DISK

```
        0  9125
        0  9126   PROCEDURE flush_am_pages_to_disk
        0  9127     (     ijl_ordinal: jmt$ijl_ordinal;
        0  9128           ijle_p: ^jmt$initiated_job_list_entry);
        0  9129
        0  9130 {   This prcedure will initiate IO to disk to write out the pages in the available modified
        0  9131 {   queue that belong to the specified job.  If IO fails for any reason the page will be moved
        0  9132 {   to the job working set.
        0  9133
        0  9134     VAR
        0  9135       ajlo: jmt$ajl_ordinal,
        0  9136       fde_p: gft$locked_file_desc_entry_p,
        0  9137       io_id: mmt$io_identifier,
        0  9138       modified_pages_removed: 0 .. osc$max_page_frames,
        0  9139       next_pfti: mmt$page_frame_index,
        0  9140       pfti: mmt$page_frame_index,
        0  9141       status: syt$monitor_status,
        0  9142       write_status: mmt$write_page_to_disk_status;
        0  9143
        0  9144     #KEYPOINT (osk$entry, 0, jsk$flush_am_pages_to_disk);
        8  9145
        8  9146 {   Set up an AJL ordinal for use by mmp$write_page_to_disk.
        8  9147
        8  9148     jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$lock_ajl, TRUE (must assign) , ajlo,
       3E  9149           status);
       3E  9150     modified_pages_removed := 0;
       3E  9151     pfti := mmv$gpql [mmc$pq_avail_modified].pqle.link.bkw;
       3E  9152
       3E  9153     io_id.specified := FALSE;
       3E  9154
       3E  9155   /scan_available_modified_queue/
       3E  9156     WHILE pfti <> 0 DO
       56  9157       next_pfti := mmv$pft_p^ [pfti].link.bkw;
       56  9158       IF (mmv$pft_p^ [pfti].aste_p^.ijl_ordinal = ijl_ordinal) AND mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m THEN
       A2  9159         gfp$mtr_get_locked_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijle_p, fde_p);
      124  9160         mmp$write_page_to_disk (fde_p, pfti, ioc$write_page, io_id, FALSE, write_status);
      150  9161         trace (jsc$ti_flush_am_pc, 1);
      150  9162         IF mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m THEN
      18E  9163
      18E  9164 {   Write_status <> ws_ok.
      18E  9165
      18E  9166           mmp$relink_page_frame (pfti, mmc$pq_job_working_set);
      1A6  9167           modified_pages_removed := modified_pages_removed + 1;
      1A6  9168           trace (jsc$ti_flush_am_relink, 1);
      1B2  9169         IFEND;
      1B2  9170       IFEND;
      1B2  9171       pfti := next_pfti;
      1B2  9172     WHILEND /scan_available_modified_queue/;
      1BA  9173
      1BA  9174     jmp$free_ajl_entry (ijle_p, jmc$lock_ajl);
      1D2  9175
      1D2  9176     IF modified_pages_removed <> 0 THEN
      1D6  9177       ijle_p^.swap_data.swapped_job_entry.available_modified_page_count :=
      1D6  9178             ijle_p^.swap_data.swapped_job_entry.available_modified_page_count + modified_pages_removed;
      1D6  9179       mmv$reassignable_page_frames.swapout_io_not_initiated :=
```

NOS/VE js : monitor mode job swapper
FLUSH_AM_PAGES_TO_DISK

```
1D6  9180              mmv$reassignable_page_frames.swapout_io_not_initiated - ijle_p^.swap_data.swapped_job_page_count +
1D6  9181              ijle_p^.job_fixed_contiguous_pages;
1D6  9182        calculate_swapped_pages (ijle_p);
23A  9183      IFEND;
23A  9184
23A  9185      IF ijle_p^.statistics.ready_task_count > 0 THEN
242  9186        trace (jsc$ti_flush_am_ready, 1);
250  9187      IFEND;
250  9188
250  9189      #KEYPOINT (osk$exit, 0, jsk$flush_am_pages_to_disk);
254  9190
254  9191    PROCEND flush_am_pages_to_disk;
  0  9192
```


NOS/VE js : monitor mode job swapper
FREE_SWAPPED_JOBS_MM_RESOURCES

```
  0  9195
  0  9196    PROCEDURE free_swapped_jobs_mm_resources
  4  9197      (     ijle_p: ^jmt$initiated_job_list_entry;
  4  9198            ijl_ordinal: jmt$ijl_ordinal;
  4  9199            last_swap_status: jmt$ijl_swap_status);
  4  9200
  4  9201  {
  4  9202  {   The purpose of this procedure is to free the memory manager resources
  4  9203  {  of each page in memory of the job being swapped out.  The swap file
  4  9204  {  descriptor is freed if it has not already been.
  4  9205  {
  4  9206  {  NOTE:
  4  9207  {       last_swap_status                swap_status                      reason/routine
  4  9208  {    jmc$iss_swapin_io_complete      jmc$iss_swapin_resource_claimed     process_io_error_on_swapin
  4  9209  {                                    jmc$iss_swapin_io_complete          process_io_error_on_swapin
  4  9210  {                                    jmc$iss_swapin_io_complete          direction change
  4  9211  {                                    jmc$iss_swapin_io_complete          page table full
  4  9212  {                                    jmc$iss_swapin_io_complete          bad swap file data
  4  9213  {    jmc$iss_swapout_io_complete     jmc$iss_free_swapped_memory         advance_swap
  4  9214  {    jmc$iss_swapped_io_complete     jmc$iss_free_swapped_memory         advance_swap
  4  9215  {
  4  9216
  4  9217      #KEYPOINT (osk$entry, 0, jsk$free_swapped_jobs_mm_resour);
  8  9218
  8  9219  {  The swap file descriptor has not been freed if last_swap_status is jmc$iss_swapin_io_complete.
  8  9220
  8  9221      IF ijle_p^.sfd_p <> NIL THEN
 16  9222        free_swap_file_descriptor (ijle_p, ijl_ordinal);
110  9223        trace (jsc$ti_sfd_freed, 1);
122  9224      IFEND;
122  9225
122  9226      IF ijle_p^.swap_status >= jmc$iss_swapin_resource_claimed THEN
12E  9227
12E  9228  {  Swapin aborted.  Free the pages we claimed.
12E  9229
12E  9230        mmp$free_memory_in_job_queues (ijle_p^.job_page_queue_list, TRUE {increment now} , FALSE
156  9231              {decrement soon} , FALSE);
156  9232        trace (jsc$ti_free_memory_si_aborted, 1);
168  9233      ELSEIF last_swap_status = jmc$iss_swapped_io_complete THEN
170  9234
170  9235  {  NOW and SOON were updated when we went from OC to S2.
170  9236
170  9237        mmp$free_memory_in_job_queues (ijle_p^.job_page_queue_list, FALSE {increment now} , FALSE
196  9238              {decrement soon} , FALSE);
196  9239        trace (jsc$ti_free_memory, 1);
1A8  9240      ELSE {last_swap_status = jmc$iss_swapout_io_complete}
1A8  9241
1A8  9242  { Going directly from OC to FM to S.  Update NOW and SOON.
1A8  9243
1A8  9244        mmp$free_memory_in_job_queues (ijle_p^.job_page_queue_list, TRUE {increment now} ,
1D0  9245              TRUE {decrement soon}, FALSE);
1D0  9246        trace (jsc$ti_free_memory, 1);
1DE  9247      IFEND;
1DE  9248
1DE  9249      #KEYPOINT (osk$exit, 0, jsk$free_swapped_jobs_mm_resour);
```

NOS/VE js : monitor mode job swapper
FREE_SWAPPED_JOBS_MM_RESOURCES

```
      1E2  9250
      1E2  9251    PROCEND free_swapped_jobs_mm_resources;
```


NOS/VE js : monitor mode job swapper
FREE_SWAP_FILE_DESCRIPTOR

```
        o  9253
        o  9254    PROCEDURE [INLINE] free_swap_file_descriptor
        o  9255      (     ijle_p: ^jmt$initiated_job_list_entry;
        o  9256            ijl_ordinal: jmt$ijl_ordinal);
        o  9257
        o  9258  {
        o  9259  {       The purpose of this procedure is to free the swap file descriptor from monitor's address
        o  9260  {   space.
        o  9261  {
        o  9262
        o  9263      VAR
        o  9264        ajlo: jmt$ajl_ordinal,
        o  9265        need_ajl: boolean,
        o  9266        status: syt$monitor_status,
        o  9267        update_segnum_sfd_p: cybil_pointer_trick;
        o  9268
        o  9269      need_ajl := (ijle_p^.ajl_ordinal = jmc$null_ajl_ordinal);
        o  9270      IF need_ajl THEN
        o  9271        jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$lock_ajl, TRUE (must assign) , ajlo,
        o  9272              status);
        o  9273        update_segnum_sfd_p.sfd_p := ijle_p^.sfd_p;
        o  9274        update_segnum_sfd_p.pva.seg := ajlo + mtc$job_fixed_segment;
        o  9275        ijle_p^.sfd_p := update_segnum_sfd_p.sfd_p;
        o  9276      IFEND;
        o  9277
        o  9278      mmp$delete_page_from_monitor (ijle_p^.sfd_p, ijle_p^.swap_data.swapped_job_entry.
        o  9279            swap_file_descriptor_page_count, status);
        o  9280
        o  9281      IF need_ajl THEN
        o  9282        jmp$free_ajl_entry (ijle_p, jmc$lock_ajl);
        o  9283      IFEND;
        o  9284
        o  9285      IF NOT status.normal THEN
        o  9286        mtp$error_stop ('JS - unable to free SFD');
        o  9287      IFEND;
        o  9288      ijle_p^.sfd_p := NIL;
        o  9289
        o  9290  {   Decrement reassignable page frames SOON. NOW was incremented when the swap file descriptor
        o  9291  {   pages were deleted from monitor's address space above.
        o  9292
        o  9293      mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
        o  9294            ijle_p^.swap_data.swapped_job_entry.swap_file_descriptor_page_count;
        o  9295
        o  9296  { Update the MAP_PURGE_TIMESTAMP. Pages assigned to the SFD were just deleted. Before the job next swaps
        o  9297  { out or in and attempts to reference the SFD, the page map must be purged. The timestamp is used to
        o  9298  { remember the time the SFD was freed.
        o  9299
        o  9300      ijle_p^.sfd_purge_timestamp := #FREE_RUNNING_CLOCK (0);
        o  9301
        o  9302    PROCEND free_swap_file_descriptor;
        o  9303
```

NOS/VE js : monitor mode job swapper
JOB_MODE_SWAPOUT

```
o  9305
o  9306 { PURPOSE:
o  9307 {   This procedure processes the swap_job_out or the special_swapout monitor swapping requests.
o  9308 { DESIGN:
o  9309 {   The caller must have the PTL lock set so that entry status is not changing through the task switch/
o  9310 {   monitor swap path, and because the entry status change done by this procedure will cause the swapped
o  9311 {   job count to change.
o  9312 {   NOTE:  The caller has verified that the job's entry status is either in memory or swapin in progress.
o  9313
o  9314   PROCEDURE [INLINE] job_mode_swapout
o  9315     (     ijl_ordinal: jmt$ijl_ordinal;
o  9316           ijle_p: ^jmt$initiated_job_list_entry;
o  9317           swap_reason: jmt$swapout_reasons;
o  9318      VAR poll_swapping: boolean;
o  9319      VAR status: syt$monitor_status);
o  9320
o  9321     VAR
o  9322       job_page_count: mmt$page_frame_index,
o  9323       old_entry_status: jmt$ijl_entry_status,
o  9324       queue_id: mmt$job_page_queue_index;
o  9325
o  9326     old_entry_status := ijle_p^.entry_status;
o  9327
o  9328     IF swap_reason = jmc$sr_operator_request THEN
o  9329       jmp$change_ijl_entry_status (ijle_p, jmc$ies_operator_force_out);
o  9330     ELSEIF swap_reason = jmc$sr_job_damaged THEN
o  9331       jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_damaged);
o  9332     ELSE
o  9333       IF ijle_p^.statistics.ready_task_count > 0 THEN
o  9334         jmp$set_entry_status_to_rt (ijl_ordinal, ijle_p);
o  9335       ELSE
o  9336         jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_swapped);
o  9337       IFEND;
o  9338     IFEND;
o  9339
o  9340     IF old_entry_status = jmc$ies_swapin_in_progress THEN
o  9341
o  9342 { If the swap status is an end state, the job must have been made a swapin candidate and relinked to the
o  9343 { swapping queue just before this monitor request got the PTL lock.  The job needs to be relinked back
o  9344 { to the proper swap queue.  Otherwise, the job must be in a blocked state (waiting for I/O, etc.).
o  9345 { Leave the job in the swapping queue. Advance_swap will advance it to the next end state.
o  9346
o  9347       IF ijle_p^.swap_status = jmc$iss_swapped_io_cannot_init THEN
o  9348         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_cannot_init);
o  9349       ELSEIF ijle_p^.swap_status = jmc$iss_swapped_io_complete THEN
o  9350         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_completed);
o  9351       ELSEIF ijle_p^.swap_status = jmc$iss_swapout_complete THEN
o  9352         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
o  9353       IFEND;
o  9354
o  9355     ELSE
o  9356
o  9357 { Old_entry_status = jmc$ies_in_memory, so swap the job out.
o  9358
o  9359       sched_trace (jsc$sc_swapout_job_mode, ijl_ordinal);
```


NOS/VE js : monitor mode job swapper
JOB_MODE_SWAPOUT

```
o  9360
o  9361       ijle_p^.job_scheduler_data.swapout_reason := swap_reason;
o  9362       ijle_p^.job_scheduler_data.job_swap_counts.job_mode :=
o  9363             ijle_p^.job_scheduler_data.job_swap_counts.job_mode + 1;
o  9364
o  9365       IF ijle_p^.swap_status = jmc$iss_executing THEN
o  9366         trace (jsc$ti_swapout_from_job_mode, 1);
o  9367         IF syv$perf_keypoints_enabled.swapping_keypoints THEN
o  9368           #KEYPOINT (osk$performance, osk$m * ijle_p^.ajl_ordinal, ptk$ajl_for_swap_out);
o  9369         IFEND;
o  9370         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
o  9371
o  9372 {  Set close approximation of swapped job page count for job mode job scheduler.  The count is also
o  9373 {  used for the service class statistics.
o  9374
o  9375         job_page_count := 0;
o  9376         FOR queue_id := LOWERVALUE (mmt$job_page_queue_index) TO UPPERVALUE (mmt$job_page_queue_index) DO
o  9377           job_page_count := job_page_count + ijle_p^.job_page_queue_list [queue_id].count;
o  9378         FOREND;
o  9379
o  9380         ijle_p^.swap_data.swapped_job_page_count := job_page_count;
o  9381         ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
o  9382
o  9383 { Swap_data.timestamp is still the time when the job completed swapin.  Swapin to swapout is residence time.
o  9384
o  9385         ijle_p^.swap_data.swapout_timestamp := #FREE_RUNNING_CLOCK (0);
o  9386
o  9387         tmp$set_lock (jmv$service_class_stats_lock);
o  9388         jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
o  9389               residence_time := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.
o  9390               swap_stats.residence_time + (ijle_p^.swap_data.swapout_timestamp - ijle_p^.swap_data.timestamp);
o  9391         jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.swapped_pages :=
o  9392               jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
o  9393               swapped_pages + ijle_p^.swap_data.swapped_job_page_count;
o  9394         tmp$clear_lock (jmv$service_class_stats_lock);
o  9395
o  9396         tmp$idle_tasks_in_job (ijle_p^.ajl_ordinal, ijle_p^.job_scheduler_data.swapout_reason, status);
o  9397         IF status.normal THEN
o  9398           ijle_p^.delayed_swapin_work := $jmt$delayed_swapin_work [];
o  9399
o  9400 { Dont clear inhibit - let it be cleared by either server job recovery
o  9401 { or by the job when it detects that the server is not longer inactive.
o  9402
o  9403           ijle_p^.terminate_access_work := $dft$mainframe_set [];
o  9404           advance_swap_state (ijle_p, jmc$iss_job_idle_tasks_complete);
o  9405           set_swapping_event (jsc$se_immediate);
o  9406           poll_swapping := FALSE;
o  9407         ELSEIF status.condition = jse$unable_to_idle_all_tasks THEN
o  9408           status.normal := TRUE;
o  9409           advance_swap_state (ijle_p, jmc$iss_idle_tasks_initiated);
o  9410         ELSE
o  9411           mtp$error_stop ('JS - UNEXPECTED CONDITION FROM IDLE TASKS');
o  9412         IFEND;
o  9413
o  9414         ?IF debug = TRUE THEN
```

NOS/VE js : monitor mode job swapper
JOB_MODE_SWAPOUT

```
          9415            IF syv$allow_jr_test THEN
          9416              IF syc$tjr_mtr_mvamjws IN syv$test_jr_system THEN
          9417                mtp$error_stop ('JOB RECOVERY TEST');
          9418              IFEND;
          9419            IFEND;
        o 9420          ?IFEND
        o 9421        IFEND;
        o 9422      IFEND;
        o 9423
        o 9424    PROCEND job_mode_swapout;
        o 9425
```


NOS/VE js : monitor mode job swapper
JOB_SWAPPING_IO

```
        o 9428
        o 9429 { PURPOSE:
        o 9430 {   This procedure performs the io necessary to swap a job in or out.
        o 9431
        o 9432     PROCEDURE job_swapping_io
        o 9433       (     ijl_ordinal: jmt$ijl_ordinal;
        o 9434             ijle_p: ^jmt$initiated_job_list_entry;
        o 9435             sfid: dmt$system_file_id;
        o 9436             io_function: iot$io_function;
        o 9437             total_swapped_page_count: 0 .. osc$max_page_frames;
        o 9438        VAR io_control_information: jst$io_control_information;
        o 9439        VAR status: syt$monitor_status);
        o 9440
        o 9441
        o 9442       VAR
        o 9443         ajlo: jmt$ajl_ordinal,
        o 9444         buffer_descriptor: mmt$buffer_descriptor,
        o 9445         fde_p: gft$file_desc_entry_p,
        o 9446         io_id: mmt$io_identifier,
        o 9447         jcb_p: ^jmt$job_control_block,
        o 9448         page_count: mmt$page_frame_index,
        o 9449         page_status: gft$page_status,
        o 9450         update_segnum_sfd_p: cybil_pointer_trick;
        o 9451
        o 9452
        o 9453       io_id.specified := FALSE;
        4 9454       io_id.ijl_ordinal := ijl_ordinal;
        4 9455
        4 9456       IF io_function = ioc$swap_out THEN
       18 9457
       18 9458 {  Add a temporary segment table entry to monitor's segment table for the job fixed segment of the job
       18 9459 {  being swapped.  Update the sfd_p in the IJL entry too.
       18 9460
       18 9461         jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$lock_ajl, TRUE {must assign} , ajlo,
       4A 9462               status);
       4A 9463         update_segnum_sfd_p.sfd_p := ijle_p^.sfd_p;
       54 9464         update_segnum_sfd_p.pva.seg := ajlo + mtc$job_fixed_segment;
       54 9465         ijle_p^.sfd_p := update_segnum_sfd_p.sfd_p;
       72 9466         ijle_p^.sfd_p^.ijl_entry := ijle_p^;
       8C 9467         jcb_p := #ADDRESS (1, update_segnum_sfd_p.pva.seg, 0);
       8C 9468         jcb_p^.swapped_job_entry := ijle_p^.swap_data.swapped_job_entry;
       AC 9469       IFEND;
       AC 9470
       AC 9471 {  Issue the necessary IO requests to swap job out.
       AC 9472
       AC 9473       buffer_descriptor.buffer_descriptor_type := mmc$bd_job_swapping_io;
       AC 9474       buffer_descriptor.ijl_ordinal := ijl_ordinal;
       AC 9475
       AC 9476     /initiate_swap_io/
       AC 9477       BEGIN
       AC 9478         gfp$mtr_get_locked_fde_p (sfid, ijle_p, fde_p);
      130 9479         REPEAT
      130 9480           page_count := (total_swapped_page_count - io_control_information.spd_index);
      130 9481           IF page_count > fde_p^.allocation_unit_size DIV osv$page_size THEN
      14E 9482             page_count := fde_p^.allocation_unit_size DIV osv$page_size;
```

NOS/VE js : monitor mode job swapper
JOB_SWAPPING_IO

```
      152  9483          IFEND;
      152  9484
      152  9485          buffer_descriptor.page_count := page_count;
      152  9486          iop$pager_io (fde_p, io_control_information.spd_index * osv$page_size, buffer_descriptor,
      192  9487                page_count * osv$page_size, io_function, io_id, status);
      192  9488          IF NOT status.normal THEN
      19A  9489            trace (jsc$ti_pager_io_error, 1);
      19A  9490            EXIT /initiate_swap_io/;
      1AC  9491          IFEND;
      1AC  9492        UNTIL io_control_information.spd_index >= total_swapped_page_count;
      1B4  9493      END /initiate_swap_io/;
      1B4  9494
      1B4  9495      IF io_function = ioc$swap_out THEN
      1BA  9496        jmp$free_ajl_entry (ijle_p, jmc$lock_ajl);
      1D2  9497      IFEND;
      1D2  9498
      1D2  9499  { Both callers of job_swapping_io check only for condition = ioe$unit_disabled.  All other 'bad'
      1D2  9500  { statuses are assumed to be a transient error--the job is advanced to a wait_io_init state;
      1D2  9501  { swapper will try to initiate the io again shortly.
      1D2  9502
      1D2  9503      IF NOT status.normal THEN
      1DA  9504        IF status.condition = ioe$unit_disabled THEN
      1EA  9505
      1EA  9506  { Reset spd_index--if io is initiated again the io will start at the beginning.
      1EA  9507
      1EA  9508          ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
      1EA  9509          ijle_p^.swap_data.swapping_io_error := ioc$unrecovered_error_unit_down;
      1EA  9510          IF ijle_p^.active_io_page_count > 0 THEN
      1FE  9511            status.normal := TRUE;
      204  9512          IFEND;
      204  9513        IFEND;
      204  9514      IFEND;
      204  9515
      204  9516      IF status.normal THEN
      20C  9517        ijle_p^.notify_swapper_when_io_complete := TRUE;
      212  9518      IFEND;
      212  9519
      212  9520    PROCEND job_swapping_io;
```


NOS/VE js : monitor mode job swapper
MOVE_AM_TO_AM

```
        0  9523
        0  9524    PROCEDURE move_am_to_am
        0  9525      (     ijle_p: ^jmt$initiated_job_list_entry;
        0  9526            available_modified_page_count: 0 .. osc$max_page_frames);
        0  9527
        0  9528  {
        0  9529  {   The purpose of this procedure is to move pages back to the  available modified
        0  9530  { queue that belong to specified job. This procedure is used if a swapout request is aborted.
        0  9531  {
        0  9532
        0  9533      VAR
        0  9534        pfti: mmt$page_frame_index,
        0  9535        i: integer;
        0  9536
        0  9537
        0  9538      trace (jsc$ti_move_am_back_to_am, 1);
        4  9539      trace (jsc$ti_move_am_back_to_am_pc, available_modified_page_count);
        4  9540      pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.fwd;
        4  9541      WHILE (pfti <> 0) AND (NOT mmv$pt_p^ [mmv$pft_p^ [pfti].pti].v) AND
       68  9542            (mmv$pft_p^ [pfti].locked_page = mmc$lp_not_locked) DO
       68  9543        mmp$relink_page_frame (pfti, mmc$pq_avail_modified);
       7C  9544        pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.fwd;
       7C  9545      WHILEND;
       BE  9546
       BE  9547      ijle_p^.swap_data.swapped_job_entry.available_modified_page_count := 0;
       BE  9548
       BE  9549    PROCEND move_am_to_am;
```

NOS/VE js : monitor mode job swapper
PROCESS_IO_ERROR_ON_SWAPIN

```
      0  9551
      0  9552   PROCEDURE process_io_error_on_swapin
      4  9553     (     ijl_ordinal: jmt$ijl_ordinal;
      4  9554           ijle_p: ^jmt$initiated_job_list_entry);
      4  9555
      4  9556 {  IO completed abnormally, free resources, put the job in swapped-out state and tell the scheduler.
      4  9557
      4  9558     free_swapped_jobs_mm_resources (ijle_p, ijl_ordinal, jmc$iss_swapin_io_complete);
     18  9559     advance_swap_state (ijle_p, jmc$iss_swapout_complete);
     2A  9560     jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_out);
     46  9561     jmp$recognize_job_dead (ijl_ordinal);
     56  9562     jmp$free_ajl_entry (ijle_p, jmc$swapping_ajl);
     6C  9563
     6C  9564   PROCEND process_io_error_on_swapin;
```


NOS/VE js : monitor mode job swapper
PROCESS_IO_ERROR_ON_SWAPOUT

```
      0  9566
      0  9567   PROCEDURE process_io_error_on_swapout
      4  9568     (     ijl_ordinal: jmt$ijl_ordinal;
      4  9569           ijle_p: ^jmt$initiated_job_list_entry;
      4  9570       VAR set_polling_event: boolean);
      4  9571
      4  9572     advance_swap_state (ijle_p, jmc$iss_swapped_io_cannot_init);
     18  9573     mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
     18  9574           ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
     18  9575     mmv$reassignable_page_frames.swapout_io_cannot_initiate :=
     18  9576           mmv$reassignable_page_frames.swapout_io_cannot_initiate + ijle_p^.swap_data.swapped_job_page_count -
     18  9577           ijle_p^.job_fixed_contiguous_pages;
     18  9578     jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_cannot_init);
     58  9579     free_swap_file_descriptor (ijle_p, ijl_ordinal);
    154  9580
    154  9581 { Recheck swap direction before returning to prevent timing problems with a task of the job going ready.
    154  9582
    154  9583     IF ijle_p^.entry_status < jmc$ies_swapped_out THEN
    162  9584       jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
    17C  9585       set_polling_event := TRUE;
    182  9586     ELSE
    182  9587       jmp$activate_job_mode_swapper;
    18A  9588     IFEND;
    18A  9589
    18A  9590   PROCEND process_io_error_on_swapout;
      0  9591
```

NOS/VE js : monitor mode job swapper
RECLAIM_IO_ERROR_PAGES

```
        0  9593
        0  9594    PROCEDURE reclaim_io_error_pages
        0  9595      (     ijl_ordinal: jmt$ijl_ordinal;
        0  9596            ijle_p: ^jmt$initiated_job_list_entry);
        0  9597
        0  9598      VAR
        0  9599        boffset: integer,
        0  9600        eoffset: integer,
        0  9601        fde_p: gft$file_desc_entry_p,
        0  9602        next_pfti: mmt$page_frame_index,
        0  9603        pfte_p: ^mmt$page_frame_table_entry,
        0  9604        pfti: mmt$page_frame_index,
        0  9605        status: syt$monitor_status,
        0  9606        tu_pfte_p: ^mmt$page_frame_table_entry,
        0  9607        tu_pfti: mmt$page_frame_index;
        0  9608
        0  9609
        0  9610      pfti := mmv$gpql [mmc$pq_swapped_io_error].pqle.link.bkw;
        4  9611
        4  9612      WHILE pfti <> 0 DO
       10  9613        pfte_p := ^mmv$pft_p^ [pfti];
       10  9614        next_pfti := pfte_p^.link.bkw;
       10  9615        IF (pfte_p^.aste_p^.ijl_ordinal = ijl_ordinal) THEN
       42  9616          trace (jsc$ti_riop_relinked, 1);
       42  9617          mmp$relink_page_frame (pfti, mmc$pq_job_io_error);
       68  9618
       68  9619 {  Reset the modified bit for all pages in this TU if memory was freed.  If the io error
       68  9620 {  occurred after the job was in the JC state and there was a page in the JWS or Job IO
       68  9621 {  error queue in the write request (due to multiple page write), the page was not moved
       68  9622 {  to an error queue and the modified bit is no longer set.  Unlock rma list resets the
       68  9623 {  modified bit while processing the error but it is lost if memory is freed.
       68  9624
       68  9625          IF ijle_p^.last_swap_status > jmc$iss_swapped_io_complete (S2) THEN
       74  9626            trace (jsc$ti_riop_mem_freed, 1);
       74  9627            gfp$mtr_get_locked_fde_p (pfte_p^.aste_p^.sfid, ijle_p, fde_p);
      108  9628            boffset := pfte_p^.sva.offset DIV fde_p^.allocation_unit_size * fde_p^.allocation_unit_size;
      108  9629            eoffset := boffset + fde_p^.allocation_unit_size;
      108  9630            tu_pfti := pfte_p^.aste_p^.pft_link.fwd;
      108  9631
      108  9632            WHILE tu_pfti <> 0 DO
      12C  9633              tu_pfte_p := ^mmv$pft_p^ [tu_pfti];
      12C  9634              IF (tu_pfte_p^.sva.offset >= boffset) AND (tu_pfte_p^.sva.offset < eoffset) AND
      164  9635                    (tu_pfte_p^.queue_id >= mmc$pq_job_base) THEN
      164  9636                trace (jsc$ti_riop_m_bit_reset, 1);
      164  9637                mmv$pt_p^ [tu_pfte_p^.pti].m := TRUE;
      186  9638              IFEND;
      186  9639              tu_pfti := mmv$pft_p^ [tu_pfti].segment_link.fwd;
      186  9640            WHILEND;
      1A6  9641          IFEND;
      1A6  9642
      1A6  9643 {  If the io error occured on an initial write, reset the fau state.
      1A6  9644
      1A6  9645          IF (pfte_p^.io_error = ioc$error_on_init) OR (pfte_p^.io_error = ioc$unit_down_on_init) THEN
      1B6  9646            trace (jsc$ti_riop_init, 1);
      1B6  9647            dmp$set_fau_state (fde_p, pfte_p^.sva.offset, status);
```


NOS/VE js : monitor mode job swapper
RECLAIM_IO_ERROR_PAGES

```
      1EA  9648          IFEND;
      1EA  9649
      1EA  9650        IFEND;
      1EA  9651        pfti := next_pfti;
      1EA  9652      WHILEND;
      1F2  9653
      1F2  9654    PROCEND reclaim_io_error_pages;
        0  9655
```

NOS/VE js : monitor mode job swapper
RECOVER_JOB_DM_TABLES

```
  0  9657
  0  9658   PROCEDURE recover_job_dm_tables
  0  9659     (     ijle_p: ^jmt$initiated_job_list_entry;
  0  9660           ijl_ordinal: jmt$ijl_ordinal;
  0  9661           system_job_monitor_sdtx_p: ^mmt$segment_descriptor_table_ex);
  0  9662
  0  9663 {
  0  9664 {  This procedure is called to update job information if the swapin is the FIRST swapin of the job
  0  9665 {  that has occurred since a system recovery. This procedure does the following:
  0  9666 {    . reset some info in the SDTX dealing with locked pages/segments.
  0  9667 {    . modifies the SFIDs in the SDTXs of each task to show the segment is waiting for recovery.
  0  9668 {    . sets the dispatching priority in the XCB to the system job priority.
  0  9669 {    . clears the read/write count in the FDE for each job file.
  0  9670 {
  0  9671
  0  9672     VAR
  0  9673       sdt_p: mmt$max_sdt_p,
  0  9674       sdtx_p: mmt$max_sdtx_p,
  0  9675       segment_number: ost$segment,
  0  9676       sfid: gft$system_file_identifier,
  0  9677       status: syt$monitor_status,
  0  9678       system_fde_p: gft$file_desc_entry_p,
  0  9679       system_ijle_p: ^jmt$initiated_job_list_entry,
  0  9680       xcb_p: ^ost$execution_control_block,
  0  9681       xcb_state: tmt$find_next_xcb_state;
  0  9682
  0  9683
  0  9684     jmp$get_ijle_p (jmv$system_ijl_ordinal, system_ijle_p);
  4  9685
  4  9686     tmp$find_next_xcb (tmc$fnx_swapping_job, ijle_p, ijl_ordinal, xcb_state, xcb_p);
 58  9687
 58  9688     WHILE xcb_p <> NIL DO
 6C  9689
 6C  9690       tmp$set_monitor_flag (xcb_p^.global_task_id, syc$mf_cause_job_recovery, status);
 90  9691       IF NOT status.normal THEN
 98  9692         mtp$error_stop ('JS - cant set job recovery flag');
 B8  9693       IFEND;
 B8  9694
 B8  9695       xcb_p^.keypoint_enable := FALSE;
 B8  9696       mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
 B8  9697       FOR segment_number := 0 TO xcb_p^.xp.segment_table_length DO
102  9698         IF sdt_p^.st [segment_number].ste.v1 <> osc$v1_invalid_entry THEN
116  9699           sdtx_p^.sdtx_table [segment_number].assign_active := osc$max_segment_length;
116  9700           sdtx_p^.sdtx_table [segment_number].segment_lock := mmc$lss_none;
116  9701
116  9702 { If the segment is a template segment (open_validating_ring is 0), copy the sfid from the system job
116  9703 { monitor.  Otherwise, if the segment is for a permanent file, mark the file as waiting for recovery.
116  9704
116  9705           IF sdtx_p^.sdtx_table [segment_number].open_validating_ring_number = 0 THEN
138  9706             sdtx_p^.sdtx_table [segment_number].sfid := system_job_monitor_sdtx_p^.
148  9707                   sdtx_table [segment_number].sfid;
148  9708           ELSEIF sdtx_p^.sdtx_table [segment_number].sfid.residence = gfc$tr_system THEN
150  9709             sdtx_p^.sdtx_table [segment_number].sfid.residence := gfc$tr_system_wait_recovery;
156  9710           IFEND;
156  9711
```


NOS/VE js : monitor mode job swapper
RECOVER_JOB_DM_TABLES

```
156  9712           IF (sdtx_p^.sdtx_table [segment_number].shadow_info.shadow_segment_kind <> mmc$ssk_none) AND
17A  9713                (sdtx_p^.sdtx_table [segment_number].shadow_info.shadow_segment_kind <>
17A  9714                mmc$ssk_segment_number) AND (sdtx_p^.sdtx_table [segment_number].shadow_info.shadow_sfid.
17A  9715                residence = gfc$tr_system) THEN
17A  9716             sdtx_p^.sdtx_table [segment_number].shadow_info.shadow_sfid.residence :=
180  9717                   gfc$tr_system_wait_recovery;
180  9718           IFEND;
180  9719         IFEND;
180  9720       FOREND;
184  9721
184  9722       xcb_p^.dispatching_priority := jmc$priority_system_job;
184  9723
184  9724       tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
1BE  9725
1BE  9726     WHILEND;
1D2  9727
1D2  9728     dmp$recover_job_dm_tables (ijle_p);
1E2  9729
1E2  9730   PROCEND recover_job_dm_tables;
  0  9731
```

NOS/VE js : monitor mode job swapper
RELINK_SWAP_QUEUE

```
 0  9734
 0  9735    PROCEDURE [XDCL] jsp$relink_swap_queue
 0  9736      (     ijl_ordinal: jmt$ijl_ordinal;
 0  9737            ijle_p: ^jmt$initiated_job_list_entry;
 0  9738            new_queue: jst$ijl_swap_queue_id);
 0  9739
 0  9740 {
 0  9741 {     The purpose of this procedure is to move and IJL entry from one swap queue
 0  9742 { to the end of another and maintain queue counts.  Process must be serialized for
 0  9743 { multiple processors.
 0  9744 {
 0  9745
 0  9746      VAR
 0  9747        backward_ijle_p: ^jmt$initiated_job_list_entry,
 0  9748        current_queue: jst$ijl_swap_queue_id,
 0  9749        forward_ijle_p: ^jmt$initiated_job_list_entry,
 0  9750        last_entry_in_queue: jmt$ijl_ordinal,
 0  9751        last_ijle_p: ^jmt$initiated_job_list_entry;
 0  9752
 0  9753
 0  9754      tmp$set_lock (jsv$ijl_serial_lock);
42  9755
42  9756      last_entry_in_queue := jsv$ijl_swap_queue_list [new_queue].backward_link;
42  9757      current_queue := ijle_p^.swap_queue_link.queue_id;
42  9758      IF current_queue = new_queue THEN
60  9759        IF new_queue <> jsc$isqi_swapping THEN
64  9760          mtp$error_stop ('JS - relink_swap_queue called to relink to same queue.');
88  9761        ELSE
88  9762          tmp$clear_lock (jsv$ijl_serial_lock);
BC  9763          RETURN;
BE  9764        IFEND;
BE  9765      IFEND;
BE  9766
BE  9767 {  Remove entry from old swap queue if it is not in the null queue.
BE  9768
BE  9769      IF current_queue <> jsc$isqi_null THEN
C2  9770        IF ijle_p^.swap_queue_link.backward_link <> jmv$null_ijl_ordinal THEN
D6  9771          jmp$get_ijle_p (ijle_p^.swap_queue_link.backward_link, backward_ijle_p);
D6  9772          backward_ijle_p^.swap_queue_link.forward_link := ijle_p^.swap_queue_link.forward_link;
104 9773        ELSE
104 9774          jsv$ijl_swap_queue_list [current_queue].forward_link := ijle_p^.swap_queue_link.forward_link;
112 9775        IFEND;
112 9776
112 9777        IF ijle_p^.swap_queue_link.forward_link <> jmv$null_ijl_ordinal THEN
122 9778          jmp$get_ijle_p (ijle_p^.swap_queue_link.forward_link, forward_ijle_p);
122 9779          forward_ijle_p^.swap_queue_link.backward_link := ijle_p^.swap_queue_link.backward_link;
152 9780        ELSE
152 9781          jsv$ijl_swap_queue_list [current_queue].backward_link := ijle_p^.swap_queue_link.backward_link;
160 9782        IFEND;
160 9783
160 9784        jsv$ijl_swap_queue_list [current_queue].count := jsv$ijl_swap_queue_list [current_queue].count - 1;
170 9785      IFEND;
170 9786
170 9787      IF jsv$ijl_swap_queue_list [current_queue].backward_link = jmv$null_ijl_ordinal THEN
186 9788        IF jsv$ijl_swap_queue_list [current_queue].forward_link <> jmv$null_ijl_ordinal THEN
```


NOS/VE js : monitor mode job swapper
RELINK_SWAP_QUEUE

```
192  9789          mtp$error_stop ('JS - swap queue linkage error.');
1B2  9790        IFEND;
1B6  9791      ELSE
1B6  9792        IF jsv$ijl_swap_queue_list [current_queue].forward_link = jmv$null_ijl_ordinal THEN
1BE  9793          mtp$error_stop ('JS - swap queue linkage error.');
1DE  9794        IFEND;
1DE  9795      IFEND;
1DE  9796
1DE  9797 {  Add entry to the end of the new queue unless it is the null queue.  If it is the null queue just change
1DE  9798 {  the queue id.  Entries in the null queue are not linked.
1DE  9799
1DE  9800      IF new_queue <> jsc$isqi_null THEN
1E6  9801        IF last_entry_in_queue <> jmv$null_ijl_ordinal THEN
1F6  9802          jmp$get_ijle_p (last_entry_in_queue, last_ijle_p);
1F6  9803          last_ijle_p^.swap_queue_link.forward_link := ijl_ordinal;
1F6  9804          ijle_p^.swap_queue_link.backward_link := last_entry_in_queue;
22A  9805        ELSE
22A  9806          ijle_p^.swap_queue_link.backward_link := jmv$null_ijl_ordinal;
22A  9807          jsv$ijl_swap_queue_list [new_queue].forward_link := ijl_ordinal;
244  9808        IFEND;
244  9809
244  9810        ijle_p^.swap_queue_link.forward_link := jmv$null_ijl_ordinal;
244  9811        jsv$ijl_swap_queue_list [new_queue].backward_link := ijl_ordinal;
244  9812        jsv$ijl_swap_queue_list [new_queue].count := jsv$ijl_swap_queue_list [new_queue].count + 1;
268  9813      IFEND;
268  9814
268  9815 {  Check queue links for correctness.
268  9816
268  9817      IF jsv$ijl_swap_queue_list [new_queue].backward_link = jmv$null_ijl_ordinal THEN
27E  9818        IF jsv$ijl_swap_queue_list [new_queue].forward_link <> jmv$null_ijl_ordinal THEN
28A  9819          mtp$error_stop ('JS - swap queue linkage error.');
2AA  9820        IFEND;
2AE  9821      ELSE
2AE  9822        IF jsv$ijl_swap_queue_list [new_queue].forward_link = jmv$null_ijl_ordinal THEN
2B6  9823          mtp$error_stop ('JS - swap queue linkage error.');
2D6  9824        IFEND;
2D6  9825      IFEND;
2D6  9826
2D6  9827      ijle_p^.swap_queue_link.queue_id := new_queue;
2D6  9828
2D6  9829      tmp$clear_lock (jsv$ijl_serial_lock);
312  9830
312  9831    PROCEND jsp$relink_swap_queue;
```

NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES

```
        o  9833
        o  9834 {
        o  9835 {   The purpose of this procedure is restore the memory manager tables so that the job being swapped
        o  9836 {   in may proceed with execution from the point at which it was interrupted when swapped out.  The
        o  9837 {   page frame table, page table and AST table are updated for the page frames swapped
        o  9838 {   out.  If an asid is reassigned the asid is updated in each task's segment table and the system
        o  9839 {   file table.  The segment table address in each task's exchanges package is also updated.
        o  9840 {
        o  9841
        o  9842
        o  9843   PROCEDURE reset_swapped_job_mm_tables
        o  9844     (     ijl_ordinal: jmt$ijl_ordinal;
        o  9845           ijle_p: ^jmt$initiated_job_list_entry;
        o  9846           swapped_job_entry: jmt$swapped_job_entry;
        o  9847      VAR sfd_p: ^jst$swap_file_descriptor;
        o  9848      VAR status: syt$monitor_status);
        o  9849
        o  9850
```


NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES
change_asids_in_sfd

```
        o  9853
        o  9854 { PURPOSE:
        o  9855 {   This procedure is called whenever an ASID is reassigned during swapin.
        o  9856 { DESIGN:
        o  9857 {     The procedure does the following:
        o  9858 {     . Scan the rest of the SPD array. Remaining entries that used the old ASID are updated to
        o  9859 {         reflect the new ASID.
        o  9860
        o  9861     PROCEDURE change_asids_in_sfd
        o  9862       (     starting_spd_index: 0 .. osc$max_page_frames;
        o  9863             new_asid: ost$asid;
        o  9864             new_asti: mmt$ast_index;
        o  9865             new_aste_p: ^mmt$active_segment_table_entry;
        o  9866             ijle_p: ^jmt$initiated_job_list_entry;
        o  9867             changing_jf_asid: boolean);
        o  9868
        o  9869       VAR
        o  9870         existing_entry: boolean,
        o  9871         fde_p: gft$locked_file_desc_entry_p,
        o  9872         old_asid: ost$asid,
        o  9873         spd_index: 0 .. osc$max_page_frames;
        o  9874
        o  9875
        o  9876 {  Change the ASIDs in the rest of the SFD for each that used the ASID that was just reassigned.
        o  9877 {  The entries in the SFD prior to the current dont have to be changed since they will never be referenced
        o  9878 {  again.
        o  9879 {  Pages can have their ASID changed more than once; the entry_updated flag helps to differentiate those
        o  9880 {  pages from other pages.  For example, if asid AAAA changes to BBBB, and later BBBB changes to CCCC, the
        o  9881 {  the entry_updated flag differentiates BBBB pages that had been AAAA pages from pages that happened
        o  9882 {  to be using asid BBBB when they swapped out.
        o  9883
        o  9884       reset_changed_asid := TRUE;
        8  9885       old_asid := sfd_p^.swapped_page_descriptors [starting_spd_index].pft_entry.sva.asid;
        8  9886       existing_entry := sfd_p^.swapped_page_descriptors [starting_spd_index].entry_updated;
        8  9887       IF existing_entry THEN
       2A  9888         trace (jsc$ti_change_asid_again, 1);
       3C  9889       ELSE
       3C  9890         trace (jsc$ti_change_asid, 1);
       4A  9891       IFEND;
       4A  9892       FOR spd_index := starting_spd_index TO UPPERBOUND (sfd_p^.swapped_page_descriptors) DO
       64  9893         IF (existing_entry = sfd_p^.swapped_page_descriptors [spd_index].entry_updated) AND
       84  9894               (old_asid = sfd_p^.swapped_page_descriptors [spd_index].pft_entry.sva.asid) THEN
       84  9895           trace (jsc$ti_change_asid_sfd, 1);
       84  9896           sfd_p^.swapped_page_descriptors [spd_index].page_table_entry.pageid.asid := new_asid;
       84  9897           sfd_p^.swapped_page_descriptors [spd_index].pft_entry.sva.asid := new_asid;
       84  9898           sfd_p^.swapped_page_descriptors [spd_index].pft_entry.aste_p := new_aste_p;
       84  9899           sfd_p^.swapped_page_descriptors [spd_index].entry_updated := TRUE;
       AE  9900         IFEND;
       AE  9901       FOREND;
       B2  9902
       B2  9903       IF (new_aste_p^.sfid.residence <> gfc$tr_system_wait_recovery) AND (NOT changing_jf_asid) THEN
       C0  9904         gfp$mtr_get_locked_fde_p (new_aste_p^.sfid, ijle_p, fde_p);
      13E  9905         fde_p^.asti := new_asti;
      146  9906       IFEND;
```

NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES
change_asids_in_sfd

```
      146  9907
      146  9908      PROCEND change_asids_in_sfd;
```


NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES

```
        0  9911
        0  9912      VAR
        0  9913        changing_jf_asid: boolean,
        0  9914        count: integer,
        0  9915        current_queue_id: mmt$page_frame_queue_id,
        0  9916        found_sva: boolean,
        0  9917        existing_pfti: mmt$page_frame_index,
        0  9918        existing_pfte_p: ^mmt$page_frame_table_entry,
        0  9919        fde_p: gft$file_desc_entry_p,
        0  9920        jf_asid: ost$asid,
        0  9921        jf_asid_changed: boolean,
        0  9922        jf_aste_p: ^mmt$active_segment_table_entry,
        0  9923        jf_asti: mmt$ast_index,
        0  9924        jf_sfid: gft$system_file_identifier,
        0  9925        live_aste_p: ^mmt$active_segment_table_entry,
        0  9926        mpt_count: integer,
        0  9927        mpt_status: mmt$make_pt_entry_status,
        0  9928        msg: string (70),
        0  9929        new_asid: ost$asid,
        0  9930        new_aste_p: ^mmt$active_segment_table_entry,
        0  9931        new_asti: mmt$ast_index,
        0  9932        next_pfti: mmt$page_frame_index,
        0  9933        pfti: mmt$page_frame_index,
        0  9934        pt_full_status: mmt$pt_full_status,
        0  9935        pti: integer,
        0  9936        recovery: boolean,
        0  9937        reset_changed_asid: boolean,
        0  9938        spd_index: 0 .. osc$max_page_frames,
        0  9939        spd_p: ^jst$swapped_page_descriptor;
        0  9940
        0  9941  { When the job was last swapped out and the old swap file descriptor freed, the IJL.PURGE_MAP_TIMESTAMP
        0  9942  { was set equal to the value of the free running clock. The page map must be purged if it has not been
        0  9943  { purged since that time. If the map is NOT purged, references to the SFD may use the OLD page frames
        0  9944  { that were assigned at the PREVIOUS swapout. Purging of the map has been delayed since it will usually
        0  9945  { NOT be required at this point since something else will have purged the map.
        0  9946
        0  9947      mmp$conditional_purge_all_map (ijle_p^.sfd_purge_timestamp);
       3A  9948
       3A  9949
       3A  9950  { The following code will verify the swap file descriptor. The action the system will take
       3A  9951  { upon finding corrupted data in the swap file descriptor depends on the value of the
       3A  9952  { system attribute-HALT_ON_SWAPIN_FAILURE.
       3A  9953
       3A  9954      IF sfd_p^.ijl_entry.system_supplied_name <> ijle_p^.system_supplied_name THEN
       50  9955        IF jsv$halt_on_swapin_failure THEN
       5C  9956          mtp$error_stop ('Bad swap file descriptor data detected.');
       80  9957        ELSE
       80  9958          msg := ' Job XXXXXXXXXXXXXXXXXX is dead. Bad swap data detected.';
       8E  9959          msg (6, 19) := ijle_p^.system_supplied_name;
       98  9960          dpp$display_error (msg);
       B4  9961          ijle_p^.hung_task_in_job := TRUE;
       B4  9962          IF ijle_p^.queue_file_information.job_abort_disposition = jmc$restart_on_abort THEN
       C2  9963            ijle_p^.queue_file_information.job_recovery_disposition := jmc$restart_on_recovery;
       CA  9964          ELSE { jmc$terminate_on_abort
       CA  9965            ijle_p^.queue_file_information.job_recovery_disposition := jmc$terminate_on_recovery;
```

```
      DO  9966          IFEND;
      DO  9967          status.normal := FALSE;
      DO  9968          status.condition := jse$bad_swap_file_data_detected;
      DO  9969          RETURN;
      EO  9970        IFEND;
      EO  9971      IFEND;
      EO  9972
      EO  9973      current_queue_id := LOWERVALUE (mmt$job_page_queue_index);
      EO  9974      pfti := ijle_p^.job_page_queue_list [current_queue_id].link.bkw;
      EO  9975      spd_index := LOWERBOUND (sfd_p^.swapped_page_descriptors);
      EO  9976
      EO  9977 { If this is the first swapin since a system recovery, the old AST entry cannot be referenced since
      EO  9978 { the AST may have moved. Set a flag for subsequent use to indicate if this is a recovery swapin.
      EO  9979
      EO  9980      recovery := jmc$dsw_job_recovery IN ijle_p^.delayed_swapin_work;
      EO  9981      jf_asid_changed := FALSE;
      EO  9982      reset_changed_asid := FALSE;
      EO  9983
      EO  9984 { Restore the SFID in the ASTE for the job fixed segment.  (The sfid was unknown when the aste was assigned
      EO  9985 { in claim_pages_for_swapin.  Pick up the sfid from the first page of the swapped page descripto, which is
      EO  9986 { a job fixed page.  The job fixed sfid will not change.)
      EO  9987
      EO  9988      jf_asid := ijle_p^.job_fixed_asid;
      EO  9989      mmp$asti (jf_asid, jf_asti);
     122  9990      jf_aste_p := ^mmv$ast_p^ [jf_asti];
     122  9991      jf_sfid := sfd_p^.swapped_page_descriptors [spd_index].ast_entry.sfid;
     122  9992      jf_aste_p^.sfid := jf_sfid;
     122  9993
     122  9994 {  If the ASID of job fixed has changed, update the ASIDs in the swap file descriptor.
     122  9995 {  NOTE:  The swapped page descriptor entry_updated field was set to TRUE by mmp$build_lock_rma_list
     122  9996 {  for all job fixed pages.  This was done to differentiate job fixed pages from other fixed pages.
     122  9997 {  When scanning each page in the swap file descriptor to see if the ASID needs to be reclaimed/
     122  9998 {  reassigned, nothing will need to be done for job fixed pages, because they have been updated
     122  9999 {  here, if necessary.
     122 10000
     122 10001      IF (ijle_p^.job_fixed_asid <> sfd_p^.swapped_page_descriptors [spd_index].pft_entry.sva.asid) OR
     156 10002          (recovery) THEN
*WARN*   10003        change_asids_in_sfd (spd_index, jf_asid, jf_asti, jf_aste_p, ijle_p, TRUE);
     184 10004        jf_asid_changed := TRUE;
     194 10005      IFEND;
     194 10006
     194 10007
     194 10008 { Loop through each page in the swap file descriptor.
     194 10009 { Reclaim the old ASID if it is still available (may still be assigned) or assign a new ASID
     194 10010 { if the old ASID has been reused for something else. Make PT entries for each page.
     194 10011
     194 10012      WHILE pfti <> 0 DO
     198 10013        next_pfti := mmv$pft_p^ [pfti].link.bkw;
     198 10014        spd_p := ^sfd_p^.swapped_page_descriptors [spd_index];
     198 10015        live_aste_p := spd_p^.pft_entry.aste_p;
     198 10016
     198 10017 {    If the SPD entry has already been updated (as a result of reassigning the ASID and updating the SPD
     198 10018 {    array), skip the following blocks of code that assign/reclaim the AST entry.
     198 10019 {    Note: 'entry_updated' is reset by mmp$build_lock_rma_list on swapout.
     198 10020
```



```
     198 10021        IF spd_p^.entry_updated THEN
     1D2 10022
     1D2 10023 [nothing needs to be done
     1D2 10024
     1D2 10025          trace (jsc$ti_rmmt_no_change, 1);
     1E4 10026
     1E4 10027 {    If the page belongs to a permanent file the ASID can be reclaimed only if the AST entry is still
     1E4 10028 {    assigned to the same SFID. (AST is not actually reclaimed - its still assigned)
     1E4 10029 {    If the AST is not still assigned, check with DM to see if another ASID has been assigned.
     1E4 10030 {    If this is a recovery swapin, throw the page away unless it's been modified; if the page has been
     1E4 10031 {    modified, set the AST entry to indicate the page is awaiting recovery.
     1E4 10032 {
     1E4 10033 {    NOTE:  After a new asid is assigned, the ast entry information is copied from the swapped page
     1E4 10034 {    descriptor ast entry.  Because the spd ast entry contains stale information with respect to
     1E4 10035 {    pages_in_memory and pft_link, those fields must be zeroed out in the new entry.  (This occurs
     1E4 10036 {    after each call to mmp$assign_asid.
     1E4 10037
     1E4 10038        ELSEIF (spd_p^.ast_entry.sfid.residence = gfc$tr_system) THEN
     1EE 10039          trace (jsc$ti_rmmt_pf, 1);
     1EE 10040          IF recovery THEN
     200 10041            IF spd_p^.page_table_entry.m THEN
     20A 10042              trace (jsc$ti_rmmt_pf_rec_ptm, 1);
     20A 10043              mmp$assign_asid (new_asid, new_asti, new_aste_p);
     238 10044              spd_p^.ast_entry.sfid.residence := gfc$tr_system_wait_recovery;
     238 10045              new_aste_p^ := spd_p^.ast_entry;
     24C 10046              new_aste_p^.pages_in_memory := 0;
     24C 10047              new_aste_p^.pft_link.bkw := 0;
     24C 10048              new_aste_p^.pft_link.fwd := 0;
*WARN*   10049              change_asids_in_sfd (spd_index, new_asid, new_asti, new_aste_p, ijle_p, FALSE);
     294 10050            ELSE
     294 10051              trace (jsc$ti_rmmt_pf_rec_ptu, 1);
     294 10052              mmp$relink_page_frame (pfti, mmc$pq_free);
     2B2 10053              pfti := 0; {prevent making PT entry}
     2C6 10054            IFEND;
     2CA 10055          ELSEIF (spd_p^.ast_entry.sfid <> live_aste_p^.sfid) OR NOT live_aste_p^.in_use THEN
     2DE 10056            gfp$mtr_get_fde_p (spd_p^.ast_entry.sfid, ijle_p, fde_p);
     348 10057            new_asti := fde_p^.asti;
     348 10058            IF new_asti = 0 THEN
     35C 10059              trace (jsc$ti_rmmt_pf_assign_asid, 1);
     35C 10060              mmp$assign_asid (new_asid, new_asti, new_aste_p);
     38C 10061              new_aste_p^ := spd_p^.ast_entry;
     39A 10062              new_aste_p^.pages_in_memory := 0;
     39A 10063              new_aste_p^.pft_link.bkw := 0;
     39A 10064              new_aste_p^.pft_link.fwd := 0;
     3AC 10065            ELSE
     3AC 10066              trace (jsc$ti_rmmt_pf_reuse_asid, 1);
     3AC 10067              mmp$asid (new_asti, new_asid);
     3D0 10068              new_aste_p := ^mmv$ast_p^ [new_asti];
     3E6 10069            IFEND;
*WARN*   10070            change_asids_in_sfd (spd_index, new_asid, new_asti, new_aste_p, ijle_p, FALSE);
     454 10071          IFEND;
     466 10072
     466 10073 {    If the segment is a local file or transient segment, the ASID can NOT be reclaimed if some other job
     466 10074 {    has used the AST entry since the current job used it OR the AST entry has already been assigned to be
     466 10075 {    used for another segment of current job.
```

```
NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES

     466 10076
     466 10077         ELSEIF recovery OR (live_aste_p^.ijl_ordinal <> ijl_ordinal) OR
     48E 10078               (live_aste_p^.in_use AND (live_aste_p^.sfid <> spd_p^.ast_entry.sfid)) THEN
     48E 10079           trace (jsc$ti_rmmt_lf_assign_asid, 1);
     48E 10080           mmp$assign_asid (new_asid, new_asti, new_aste_p);
     4C0 10081           new_aste_p^ := spd_p^.ast_entry;
     4CE 10082           new_aste_p^.pages_in_memory := 0;
     4CE 10083           new_aste_p^.pft_link.bkw := 0;
     4CE 10084           new_aste_p^.pft_link.fwd := 0;
*WARN*    10085           change_asids_in_sfd (spd_index, new_asid, new_asti, new_aste_p, ijle_p, FALSE);
     524 10086
     524 10087 {    The same ASID can be used. If the AST entry is not currently assigned it must be reclaimed. The AST
     524 10088 {    might still be assigned if pages of the segment remained in the AVAIL queue while the job was swapped out.
     524 10089 {    Preserve the live ast entry pages_in_memory and pft_link fields.  (The spd ast_entry contains stale
     524 10090 {    information in those two fields.)
     524 10091
     524 10092         ELSEIF NOT live_aste_p^.in_use THEN
     52C 10093           trace (jsc$ti_rmmt_lf_reuse_asid, 1);
     52C 10094           mmp$assign_specific_asid (live_aste_p);
     54A 10095           spd_p^.ast_entry.pages_in_memory := live_aste_p^.pages_in_memory;
     54A 10096           spd_p^.ast_entry.pft_link.bkw := live_aste_p^.pft_link.bkw;
     54A 10097           spd_p^.ast_entry.pft_link.fwd := live_aste_p^.pft_link.fwd;
     54A 10098           live_aste_p^ := spd_p^.ast_entry;
     56C 10099         IFEND;
     56C 10100
     56C 10101
     56C 10102
     56C 10103 {  Create and reserve the page table entry. (If the page has been discarded, PFTI is zero.)
     56C 10104
     56C 10105         IF pfti <> 0 THEN
     570 10106           mpt_count := 0;
     570 10107           REPEAT
     576 10108
     576 10109 { Zero out the segment link in the swapped page descriptor pft_entry;  the links in the entry are
     576 10110 { left over from when the job was running before.  (Non-zero links in make_pt_entry will cause a failure.)
     576 10111
     576 10112             spd_p^.pft_entry.segment_link.bkw := 0;
     576 10113             spd_p^.pft_entry.segment_link.fwd := 0;
     576 10114             mmp$make_pt_entry (spd_p^.pft_entry.sva, pfti, spd_p^.pft_entry.aste_p, ^spd_p^.pft_entry,
     5AA 10115                   mpt_status);
     5AA 10116
     5AA 10117 {     If the page table entry was made sucessfully, restore the PFT entry and the page table V C M bits.
     5AA 10118 {     Zero out the pft.active_io_count in case PFTS io was active when the swapped page descriptor
     5AA 10119 {     information was captured.
     5AA 10120
     5AA 10121             CASE mpt_status OF
     5BA 10122             = mmc$mpt_done = [ Normal return
     5BA 10123               trace (jsc$ti_rmmt_pt_done, 1);
     5BA 10124               spd_p^.pft_entry.link := mmv$pft_p^ [pfti].link;
     5BA 10125               mmv$pft_p^ [pfti] := spd_p^.pft_entry;
     608 10126               mmv$pft_p^ [pfti].task_queue.head := 0;
     608 10127               mmv$pft_p^ [pfti].task_queue.tail := 0;
     608 10128               mmv$pft_p^ [pfti].active_io_count := 0;
     608 10129               pti := spd_p^.pft_entry.pti;
     608 10130               mmv$pt_p^ [pti].u := spd_p^.page_table_entry.u;
```



```
NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES

     608 10131               mmv$pt_p^ [pti].m := spd_p^.page_table_entry.m;
*WARN*    10132               mmv$pt_p^ [pti].v := spd_p^.page_table_entry.v;
     664 10133
     664 10134 {     If a page table full reject occurred, call MM to process the PT full condition. If still not successful
     664 10135 {     abort the swappin and free the resources assigned to the job. If page table full processing was
     664 10136 {     successful and the ASID was changed, update the CHANGED ASID list.
     664 10137
     664 10138             = mmc$mpt_page_table_full =
     664 10139               changing_jf_asid := (spd_p^.pft_entry.sva.asid = jf_asid);
     664 10140               mmp$process_page_table_full (spd_p^.pft_entry.sva, new_asid, new_asti, new_aste_p,
     6A4 10141                     pt_full_status);
     6A4 10142               trace (jsc$ti_rmmt_pt_full, 1);
     6A4 10143               mpt_count := mpt_count + 1;
     6A4 10144               IF (pt_full_status = mmc$pfs_failed) OR (mpt_count > 20) THEN
     6C8 10145                 IF spd_p^.pft_entry.aste_p^.pages_in_memory = 0 THEN
     6D6 10146                   mmp$free_asid (spd_p^.pft_entry.sva.asid, spd_p^.pft_entry.aste_p);
     6EE 10147                 IFEND;
     6EE 10148                 trace (jsc$ti_rmmt_pt_full_failed, 1);
 WARN*    10149                 free_swapped_jobs_mm_resources (ijle_p, ijl_ordinal, jmc$iss_swapin_io_complete);
     716 10150                 mtp$set_status_abnormal ('JS', jse$pt_full_on_swap_in, status);
     716 10151                 RETURN;
     72E 10152               ELSEIF pt_full_status = mmc$pfs_input_asid_reassigned THEN
     734 10153                 trace (jsc$ti_rmmt_pt_full_succ, 1);
*WARN*    10154                 change_asids_in_sfd (spd_index, new_asid, new_asti, new_aste_p, ijle_p, changing_jf_asid);
     782 10155                 IF changing_jf_asid THEN
     786 10156                   jf_asid_changed := TRUE;
     786 10157                   jf_asid := new_asid;
     786 10158                   jf_asti := new_asti;
     786 10159                   trace (jsc$ti_pt_full_reassign_jf, 1);
     7A4 10160                 IFEND;
     7C2 10161               IFEND;
     7D4 10162
     7D4 10163 {     If an entry already exists, it better belong to a permanent file that is now in
     7D4 10164 {     a shared queue or to a local file in one of the invalid page table queues or the
     7D4 10165 {     io error while swapped queue.
     7D4 10166
     7D4 10167             = mmc$mpt_page_already_exists =
     7D4 10168               #HASH_SVA (spd_p^.pft_entry.sva, pti, count, found_sva);
     7DA 10169               IF NOT found_sva THEN
     7F0 10170                 mtp$error_stop ('JS - cannot find existing job_shared page.');
     812 10171               IFEND;
     812 10172               existing_pfti := mmv$pt_p^ [pti].rma * 512 DIV osv$page_size;
     812 10173               existing_pfte_p := ^mmv$pft_p^ [existing_pfti];
     812 10174
     812 10175 {   IF a page in the jws had io active when memory was freed, it was put into the available modified
     812 10176 {   queue.  If IO has not yet completed, the page is still there.  We will delete the new page coming in
     812 10177 {   incase the IO completes with an error and we need to reset the modified bit.  IO completed normally
     812 10178 {   if the existing page is in the available queue and we can just delete it.  If an io error occurred,
     812 10179 {   the existing page is in the swapped io error queue.  We will delete the new page coming in and
     812 10180 {   reclaim the io error page later in swapin.
     812 10181
     812 10182               IF (existing_pfte_p^.aste_p^.sfid.residence = gfc$tr_job) THEN
     85C 10183                 IF (existing_pfte_p^.queue_id = mmc$pq_avail) THEN
     868 10184                   trace (jsc$ti_rmmt_pte_exists_a, 1);
     868 10185                   mmp$delete_pt_entry (existing_pfti, TRUE);
```

NOS/VE js : monitor mode job swapper
RESET_SWAPPED_JOB_MM_TABLES

```
      88A 10186                 mmp$relink_page_frame (existing_pfti, mmc$pq_free);
      8A4 10187               ELSEIF ((existing_pfte_p^.queue_id = mmc$pq_swapped_io_error) OR
      8B2 10188                    (existing_pfte_p^.queue_id = mmc$pq_avail_modified)) THEN
      8B2 10189                 IF (existing_pfte_p^.queue_id = mmc$pq_swapped_io_error) THEN
      8BE 10190                   trace (jsc$ti_rmmt_pte_exists_err, 1);
      8D0 10191                 ELSE
      8D0 10192                   trace (jsc$ti_rmmt_pte_exists_am, 1);
      8DE 10193                 IFEND;
      8DE 10194                 mmp$relink_page_frame (pfti, mmc$pq_free);
      8F6 10195                 mpt_status := mmc$mpt_done;
      8FE 10196               ELSE
*WARN*    10197                 mtp$error_stop ('JS - Page table entry already exists on swap in (reset tables).');
      924 10198               IFEND;
      928 10199             ELSEIF (existing_pfte_p^.aste_p^.queue_id >= mmc$pq_shared_first) AND
      93C 10200                  (existing_pfte_p^.aste_p^.queue_id <= mmc$pq_shared_last) THEN
      93C 10201               trace (jsc$ti_rmmt_pte_exists_pf, 1);
      93C 10202               mmp$relink_page_frame (pfti, mmc$pq_free);
      962 10203               mpt_status := mmc$mpt_done;
      96A 10204             ELSE
      96A 10205               mtp$error_stop ('JS - Page table entry already exists on swap in (reset tables).');
      98C 10206             IFEND;
      994 10207           CASEND;
      994 10208
      994 10209         UNTIL mpt_status = mmc$mpt_done;
      9A0 10210       IFEND;
      9A0 10211
      9A0 10212 {  Get next page frame index.
      9A0 10213
      9A0 10214       WHILE ((next_pfti = 0) OR (next_pfti = ijle_p^.swap_io_control.swap_file_descriptor_pfti)) AND
      9B4 10215             (current_queue_id < UPPERVALUE (mmt$job_page_queue_index)) DO
      9B4 10216         current_queue_id := SUCC (current_queue_id);
      9B4 10217         next_pfti := ijle_p^.job_page_queue_list [current_queue_id].link.bkw;
      9B4 10218       WHILEND;
      9DA 10219
      9DA 10220       pfti := next_pfti;
      9DA 10221
      9DA 10222       spd_index := spd_index + 1;
      9DA 10223
      9DA 10224     WHILEND;
      9E4 10225
      9E4 10226     IF jf_asid_changed THEN
      9E8 10227       gfp$mtr_get_locked_fde_p (jf_sfid, ijle_p, fde_p);
      A64 10228       fde_p^.asti := jf_asti;
      A70 10229     IFEND;
      A70 10230
      A70 10231     reset_sdt_xcb_tables (ijl_ordinal, ijle_p, TRUE, reset_changed_asid);
      A84 10232
      A84 10233   PROCEND reset_swapped_job_mm_tables;
```


NOS/VE js : monitor mode job swapper
RESET_SDT_XCB_TABLES

```
        0 10235
        0 10236 {
        0 10237 { PURPOSE:
        0 10238 {   This procedure is called at the end of swapin to reset XCB and SDT information that may
        0 10239 {   have changed while the job was swapped out.
        0 10240 { DESIGN:
        0 10241 {   The segment tables RMAs are fixed, if necessary.  If any ASIDs changed while the job was
        0 10242 {   swapped out, the old ASIDs must be zeroed out in the segment tables of all tasks of the job.
        0 10243 {   On the next page fault for a page of a segment with a zeroed out ASID, the ASID will be
        0 10244 {   obtained from the FDE.
        0 10245
        0 10246   PROCEDURE reset_sdt_xcb_tables
        0 10247     (     ijl_ordinal: jmt$ijl_ordinal;
        0 10248           ijle_p: ^jmt$initiated_job_list_entry;
        0 10249           reset_sdt_addresses: boolean;
        0 10250           reset_changed_asid: boolean);
        0 10251
        0 10252     VAR
        0 10253       asid: ost$asid,
        0 10254       aste_p: ^mmt$active_segment_table_entry,
        0 10255       asti: mmt$ast_index,
        0 10256       fde_p: gft$locked_file_desc_entry_p,
        0 10257       fix_asid: boolean,
        0 10258       global_asids_changed: boolean,
        0 10259       jf_asti: mmt$ast_index,
        0 10260       job_asids_changed: boolean,
        0 10261       max_segnum: integer,
        0 10262       max_segnum_to_update: integer,
        0 10263       recovery: boolean,
        0 10264       rma: integer,
        0 10265       segment_number: ost$segment,
        0 10266       sdt_p: mmt$max_sdt_p,
        0 10267       sdtx_p: mmt$max_sdtx_p,
        0 10268       system_job_monitor_sdt_p: mmt$max_sdt_p,
        0 10269       system_job_monitor_sdtx_p: mmt$max_sdtx_p,
        0 10270       template_asids_changed: boolean,
        0 10271       timestamp: integer,
        0 10272       xcb_p: ^ost$execution_control_block,
        0 10273       xcb_state: tmt$find_next_xcb_state;
        0 10274
        0 10275     mmp$get_max_sdt_sdtx_pointer (mtv$system_job_monitor_xcb_p, system_job_monitor_sdt_p,
        4 10276           system_job_monitor_sdtx_p);
        4 10277
        4 10278     recovery := jmc$dsw_job_recovery IN ijle_p^.delayed_swapin_work;
        4 10279
        4 10280 {  If this is the first swapin of this job since job recovery occurred, device management tables
        4 10281 {  need to be recovered.
        4 10282
        4 10283     IF recovery THEN
       4C 10284       trace (jsc$ti_rxcb_recovery, 1);
       4C 10285       recover_job_dm_tables (ijle_p, ijl_ordinal, system_job_monitor_sdtx_p);
       84 10286     IFEND;
       84 10287
       84 10288 {  Determine the kinds of updates that have to be made to the ASIDs in the segment tables of tasks in the
       84 10289 {  job. GLOBAL_ASIDS_HAVE_CHANGED means an ASID of a shared/sharable segment has changed since the job was
```

```
      84 10290 {  was swapped. JOB_ASIDS_HAVE_CHANGED means a job local ASID was changed on swapin OR a job local ASID that
      84 10291 {  belonged to the job was reassigned while the job was swapped out but no pages of the segment were in
      84 10292 {  in the swap file.
      84 10293
      84 10294      timestamp := ijle_p^.swap_data.asid_reassigned_timestamp;
      84 10295      global_asids_changed := (mmv$time_changed_global_asid > timestamp) OR
      A8 10296            (jmc$dsw_job_shared_asid_changed IN ijle_p^.delayed_swapin_work);
      A8 10297      job_asids_changed := (reset_changed_asid) OR (jmc$dsw_job_asid_changed IN ijle_p^.delayed_swapin_work);
      B8 10298      template_asids_changed := mmv$time_changed_template_asid > timestamp;
      B8 10299      IF template_asids_changed THEN
      CC 10300        trace (jsc$ti_rxcb_temp_asids_changed, 1);
      DA 10301      IFEND;
      DA 10302      IF job_asids_changed THEN
      DE 10303        trace (jsc$ti_rxcb_job_asids_changed, 1);
      EC 10304      IFEND;
      EC 10305      IF global_asids_changed THEN
      F0 10306        trace (jsc$ti_rxcb_glob_asids_changed, 1);
      FE 10307      IFEND;
      FE 10308
      FE 10309 { Determine the maximum segment number that may have to be updated. If ONLY template ASIDs have changed
      FE 10310 { the max segnum is determined by the largest template segment number in use. Otherwise all segments have
      FE 10311 { to be examined.
      FE 10312
      FE 10313      IF global_asids_changed OR job_asids_changed THEN
     105 10314        max_segnum_to_update := 4096;
     10E 10315      ELSEIF template_asids_changed THEN
     112 10316        max_segnum_to_update := mmv$max_template_segment_number;
     120 10317      ELSE
     120 10318        max_segnum_to_update := 0;
     124 10319      IFEND;
     124 10320
     124 10321 { Update the tables in job fixed. Fix the segment table RMA in each XCB. Update the ASIDS in
     124 10322 { the segment tables if necessary.
     124 10323
     124 10324      tmp$find_next_xcb (tmc$fnx_swapping_job, ijle_p, ijl_ordinal, xcb_state, xcb_p);
     14E 10325
     14E 10326      IF (max_segnum_to_update > 0) OR reset_sdt_addresses THEN
     156 10327        WHILE xcb_p <> NIL DO
     168 10328          trace (jsc$ti_rxcb_fix_xcb_sdt, 1);
     168 10329          mmp$get_max_sdt_sdtx_pointer (xcb_p, sdt_p, sdtx_p);
     168 10330
     168 10331          IF reset_sdt_addresses THEN
     1AC 10332            i#real_memory_address (sdt_p, rma);
     1C4 10333            xcb_p^.xp.segment_table_address_1 := rma DIV 10000(16);
     1C4 10334            xcb_p^.xp.segment_table_address_2 := rma MOD 10000(16);
     1EA 10335          IFEND;
     1EA 10336
     1EA 10337          IF max_segnum_to_update > 0 THEN
     1EE 10338            trace (jsc$ti_rxcb_fix_asids, 1);
     1EE 10339            max_segnum := max_segnum_to_update;
     1EE 10340            IF max_segnum = 4096 THEN
     202 10341              max_segnum := xcb_p^.xp.segment_table_length;
     20E 10342            IFEND;
     20E 10343            FOR segment_number := 0 TO max_segnum DO
     218 10344              IF (sdt_p^.st [segment_number].ste.v1 <> osc$v1_invalid_entry) AND
```



```
     234 10345                  (sdt_p^.st [segment_number].ste.asid <> 0) THEN
     234 10346
     234 10347                aste_p := ^mmv$ast_p^ [sdt_p^.st [segment_number].asti];
     234 10348                IF (NOT aste_p^.in_use) OR (aste_p^.sfid <> sdtx_p^.sdtx_table [segment_number].sfid) OR
     284 10349                      ((aste_p^.sfid.residence = gfc$tr_job) AND (ijl_ordinal <> aste_p^.ijl_ordinal)) THEN
     284 10350
     284 10351                  IF (sdtx_p^.sdtx_table [segment_number].open_validating_ring_number = 0) AND
     2AA 10352                        (sdtx_p^.sdtx_table [segment_number].sfid = system_job_monitor_sdtx_p^.
     2AA 10353                        sdtx_table [segment_number].sfid) THEN
     2AA 10354                    sdt_p^.st [segment_number] := system_job_monitor_sdt_p^.st [segment_number];
     2AA 10355                    trace (jsc$ti_rxcb_fix_templ_asid, 1);
     2CC 10356
     2CC 10357                  ELSEIF segment_number = osc$segnum_job_fixed_heap THEN
     2D2 10358                    sdt_p^.st [segment_number].ste.asid := ijle_p^.job_fixed_asid;
     2D2 10359                    mmp$asti (ijle_p^.job_fixed_asid, jf_asti);
     2FE 10360                    sdt_p^.st [segment_number].asti := jf_asti;
     2FE 10361                    trace (jsc$ti_rxcb_fix_jf_asid, 1);
     318 10362
     318 10363                  ELSEIF (sdtx_p^.sdtx_table [segment_number].sfid.residence = gfc$tr_system) OR
     334 10364                        (sdtx_p^.sdtx_table [segment_number].sfid.residence = gfc$tr_job) THEN
     334 10365                    gfp$mtr_get_locked_fde_p (sdtx_p^.sdtx_table [segment_number].sfid, ijle_p, fde_p);
     3C8 10366                    IF (sdtx_p^.sdtx_table [segment_number].sfid.residence = gfc$tr_job) THEN
*WARN*    10367                      mmp$get_verify_asti_in_fde (fde_p, sdtx_p^.sdtx_table [segment_number].sfid, ijl_ordinal,
     44A 10368                            asti);
     44A 10369                    ELSE
     44A 10370                      asti := fde_p^.asti;
     45C 10371                    IFEND;
     45C 10372                    IF asti <> 0 THEN
     464 10373                      mmp$asid (asti, asid);
     47E 10374                      sdt_p^.st [segment_number].ste.asid := asid;
     47E 10375                      sdt_p^.st [segment_number].asti := asti;
     47E 10376                      trace (jsc$ti_rxcb_fix_job_asid, 1);
     4A4 10377                    ELSE
     4A4 10378                      sdt_p^.st [segment_number].ste.asid := 0;
     4A4 10379                      trace (jsc$ti_rxcb_zero_job_asid, 1);
     4BC 10380                    IFEND;
     4CC 10381                  ELSE
     4CC 10382                    sdt_p^.st [segment_number].ste.asid := 0;
     4CC 10383                    trace (jsc$ti_rxcb_zero_asid, 1);
     4E2 10384                  IFEND;
     4E4 10385                IFEND;
     4E4 10386
     4E4 10387              IFEND;
     4E4 10388            FOREND;
     4E8 10389          IFEND;
     4E8 10390
     4E8 10391          tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
     51A 10392
     51A 10393        WHILEND;
     52E 10394      IFEND;
     52E 10395
     52E 10396      IF jmc$dsw_adjust_cpu_selections IN ijle_p^.delayed_swapin_work THEN
     53A 10397        update_processor_selections (ijle_p, ijl_ordinal);
     54E 10398      IFEND;
     54E 10399
```

NOS/VE js : monitor mode job swapper
RESET_SDT_XCB_TABLES

```
54E 10400 { Debug lists need to be updated on the first swapin for job recovery.  Update the debug lists in each XCB.
54E 10401
54E 10402     IF jmc$dsw_update_debug_lists IN ijle_p^.delayed_swapin_work THEN
55A 10403       ijle_p^.system_breakpoint_selected := FALSE;
55A 10404       tmp$find_next_xcb (tmc$fnx_swapping_job, ijle_p, ijl_ordinal, xcb_state, xcb_p);
58A 10405       WHILE xcb_p <> NIL DO
59C 10406         tmp$set_up_debug_registers (xcb_p^.global_task_id.index, ijle_p, xcb_p);
5BC 10407         tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
5E8 10408       WHILEND;
5FE 10409     IFEND;
5FE 10410
5FE 10411     IF jmc$dsw_update_server_files IN ijle_p^.delayed_swapin_work THEN
60A 10412       update_server_files (ijle_p, ijl_ordinal);
61E 10413     IFEND;
61E 10414
61E 10415 {  The swap file descriptor has not been freed if we are swapping in from disk.
61E 10416
61E 10417     IF ijle_p^.sfd_p <> NIL THEN
62C 10418       free_swap_file_descriptor (ijle_p, ijl_ordinal);
72A 10419     IFEND;
72A 10420
72A 10421 { Swap status is advanced to executing.
72A 10422
72A 10423     complete_swapin (ijl_ordinal, ijle_p, ijle_p^.swap_data.swapped_job_entry.available_modified_page_count);
73A 10424
73A 10425   PROCEND reset_sdt_xcb_tables;
  0 10426
```


NOS/VE js : monitor mode job swapper
RESTART_IDLED_TASKS

```
  0 10428
  0 10429   PROCEDURE [INLINE] restart_idled_tasks
  0 10430     (     ijl_ordinal: jmt$ijl_ordinal;
  0 10431           ijle_p: ^jmt$initiated_job_list_entry);
  0 10432
  0 10433 {
  0 10434 {       The purpose of this procedure is to restart the tasks that have been idled for swapping.
  0 10435 {  There are some timing considerations with multiple CPUs and the dispatcher.  At the time
  0 10436 {  this procedure is called the job is effectively swapped in.  The job's swap_status is set to
  0 10437 {  indicate job executing.  The job is also relinked into the null swap queue so that it can
  0 10438 {  be swapped out again if it goes into long wait before finishing the final cleanup for
  0 10439 {  swapping in.
  0 10440 {  It is not necessary to set the PTL lock to change entry status, because the transition will
  0 10441 {  not cause the swapped job count to change.  The job cannot swap out asynchronously on another
  0 10442 {  processor in long wait because the tasks have not been restarted until after the entry status
  0 10443 {  change.
  0 10444
  0 10445     jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_null);
  0 10446     advance_swap_state (ijle_p, jmc$iss_executing);
  0 10447     jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_in_memory);
  0 10448
  0 10449 {  Update counts if the job has reserved memory through the mmp$assign_pages request
  0 10450
  0 10451     IF ijle_p^.memory_reserve_request.requested_page_count > 0 THEN
  0 10452       IF (mmv$reassignable_page_frames.now - mmv$aggressive_aging_level_2) >
  0 10453             ijle_p^.memory_reserve_request.requested_page_count THEN
  0 10454         ijle_p^.memory_reserve_request.reserved_page_count :=
  0 10455               ijle_p^.memory_reserve_request.reserved_page_count +
  0 10456               ijle_p^.memory_reserve_request.requested_page_count;
  0 10457         mmv$reserved_page_count := mmv$reserved_page_count +
  0 10458               ijle_p^.memory_reserve_request.requested_page_count;
  0 10459       ELSE
  0 10460         trace (jsc$ti_reserve_memory_failed, 1);
  0 10461       IFEND;
  0 10462       ijle_p^.memory_reserve_request.requested_page_count := 0;
  0 10463     IFEND;
  0 10464
  0 10465 { If something in the job/task environment has changed, update it.
  0 10466
  0 10467     IF jmc$dsw_update_job_task_enviro IN ijle_p^.delayed_swapin_work THEN
  0 10468       tmp$update_job_task_environment (ijle_p, ijl_ordinal, tmc$fnx_swapping_job);
  0 10469     IFEND;
  0 10470
  0 10471 { While the job was swapped, if writes to local files completed with an io error, the pages
  0 10472 { were put into the swapped io error queue.  Reclaim those pages.
  0 10473
  0 10474     IF jmc$dsw_io_error_while_swapped IN ijle_p^.delayed_swapin_work THEN
  0 10475       reclaim_io_error_pages (ijl_ordinal, ijle_p);
  0 10476     IFEND;
  0 10477
  0 10478     IF syv$perf_keypoints_enabled.swapping_stack_trace THEN
  0 10479       tmp$monitor_flag_job_tasks (syc$mf_for_keypoint_traceback, ijle_p);
  0 10480     IFEND;
  0 10481
  0 10482
```

NOS/VE js : monitor mode job swapper
RESTART_IDLED_TASKS

```
o 10483 {  The XCB of this job can now be modified.
o 10484 {  This job is a candidate for being swapped out again.
o 10485
o 10486      tmp$restart_idled_tasks (ijle_p^.ajl_ordinal);
o 10487
o 10488 { While the job was swapped, if a segment that has pages in the working set changed so its
o 10489 { pages are now in the shared queue, remove the pages from the jws
o 10490
o 10491      IF jmc$dsw_job_shared_asid_changed IN ijle_p^.delayed_swapin_work THEN
o 10492        mmp$remove_swapped_shared_pages (ijle_p);
o 10493      IFEND;
o 10494
o 10495    PROCEND restart_idled_tasks;
```


NOS/VE js : monitor mode job swapper
SET_SWAPPING_EVENT

```
o 10497
o 10498    PROCEDURE [INLINE] set_swapping_event
o 10499      (     event_time: jst$swapping_event);
o 10500
o 10501 {
o 10502 {      This procedure sets up the flags so that mtm$monitor_interrupt_handler will recall
o 10503 { jsp$advance_swap immediately for swapping activity or later for polling purposes.
o 10504 {
o 10505
o 10506      VAR
o 10507        cst_p: ^ost$cpu_state_table;
o 10508
o 10509
o 10510      jsv$time_to_call_job_swapper := #FREE_RUNNING_CLOCK (0) + event_time;
o 10511
o 10512      IF event_time = jsc$se_immediate THEN
o 10513        mtp$cst_p (cst_p);
o 10514        cst_p^.dispatch_control.asynchronous_interrupts_pending := TRUE;
o 10515        osv$time_to_check_asyn := 0;
o 10516      IFEND;
o 10517
o 10518    PROCEND set_swapping_event;
```

NOS/VE js : monitor mode job swapper
DIRECTION_CHANGED_TO_IN

```
      0 10520
      0 10521    PROCEDURE direction_changed_to_in
      0 10522      (     ijl_ordinal: jmt$ijl_ordinal;
      0 10523            ijle_p: ^jmt$initiated_job_list_entry);
      0 10524
      0 10525 {
      0 10526 {     The purpose of this procedure is to swapin a job that is currently
      0 10527 {  being swapped out.
      0 10528 {
      0 10529
      0 10530      VAR
      0 10531        swap_status: jmt$ijl_swap_status;
      0 10532
      0 10533      swap_status := ijle_p^.swap_status;
      4 10534
      4 10535      IF swap_status = jmc$iss_idle_tasks_initiated THEN
      E 10536        trace (jsc$ti_sif_idle_tasks_init, 1);
      E 10537        restart_idled_tasks (ijl_ordinal, ijle_p);
    182 10538        ijle_p^.next_swap_status := jmc$iss_null;
    18A 10539
    18A 10540        ?IF debug = TRUE THEN
        10541          IF syv$allow_jr_test THEN
        10542            IF syc$tjr_mtr_rit IN syv$test_jr_system THEN
        10543              mtp$error_stop ('JOB RECOVERY TEST');
        10544            IFEND;
        10545          IFEND;
    18A 10546        ?IFEND
    18A 10547      ELSEIF (swap_status = jmc$iss_job_allocate_swap_file) OR
    1A2 10548           (swap_status = jmc$iss_wait_allocate_swap_file) OR (swap_status = jmc$iss_wait_job_io_complete) OR
    1A2 10549           (swap_status = jmc$iss_wait_allocate_sfd) THEN
    1A2 10550        trace (jsc$ti_sif_wait_state, 1);
    1A2 10551        IF swap_status = jmc$iss_wait_allocate_sfd THEN
    1B6 10552          jsv$pages_needed_for_sfd := 0;
    1B6 10553          trace (jsc$ti_zero_out_pages_for_sfd_1, 1);
    1C6 10554        IFEND;
    1C6 10555        ijle_p^.next_swap_status := jmc$iss_null;
    1C6 10556        swapin_before_io (ijl_ordinal, ijle_p);
    1DE 10557        ?IF debug = TRUE THEN
        10558          IF syv$allow_jr_test THEN
        10559            IF syc$tjr_mtr_mamtam IN syv$test_jr_system THEN
        10560              mtp$error_stop ('JOB RECOVERY TEST');
        10561            IFEND;
        10562          IFEND;
    1DE 10563        ?IFEND
    1DE 10564
    1DE 10565      ELSEIF (swap_status = jmc$iss_swapout_io_initiated) OR (swap_status = jmc$iss_wait_swapout_io_init) THEN
    1EC 10566        trace (jsc$ti_sif_swapout_io_init, 1);
    1EC 10567        ijle_p^.notify_swapper_when_io_complete := FALSE;
    1EC 10568        free_swap_file_descriptor (ijle_p, ijl_ordinal);
    2FA 10569
    2FA 10570 {  Update reassignable page frames to reflect swapout io aborted, job is being swapped in.
    2FA 10571
    2FA 10572        mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
    2FA 10573              ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
    2FA 10574        ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
```


NOS/VE js : monitor mode job swapper
DIRECTION_CHANGED_TO_IN

```
    2FA 10575        ijle_p^.next_swap_status := jmc$iss_null;
    2FA 10576        swapin_after_io (ijl_ordinal, ijle_p);
    32A 10577      ELSE
    32A 10578        mtp$error_stop ('JS - inconsistant swap status on swap direction change.');
    34A 10579      IFEND;
    34A 10580
    34A 10581    PROCEND direction_changed_to_in;
```

NOS/VE js : monitor mode job swapper
SWAPIN_BEFORE_IO

```
        0 10583
        0 10584    PROCEDURE swapin_before_io
        0 10585      (     ijl_ordinal: jmt$ijl_ordinal;
        0 10586            ijle_p: ^jmt$initiated_job_list_entry);
        0 10587
        0 10588      VAR
        0 10589        ajl_ordinal: jmt$ajl_ordinal,
        0 10590        status: syt$monitor_status;
        0 10591
        0 10592      jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$swapping_ajl, FALSE {must assign} ,
       38 10593           ajl_ordinal, status);
       38 10594      IF NOT status.normal THEN
       40 10595        trace (jsc$ti_no_ajlo_swapin_before_io, 1);
       40 10596        IF (ijle_p^.swap_status = jmc$iss_wait_job_io_complete) OR
       5E 10597             (ijle_p^.swap_status = jmc$iss_wait_allocate_sfd) THEN
       5E 10598          mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
       5E 10599                ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
       5E 10600          mmv$reassignable_page_frames.swapout_io_not_initiated :=
       82 10601                mmv$reassignable_page_frames.swapout_io_not_initiated +
       82 10602                ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
       82 10603        IFEND;
       82 10604        advance_swap_state (ijle_p, jmc$iss_swapped_no_io);
       96 10605        jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_not_init);
       B2 10606        jmp$reset_job_to_swapped_out (ijl_ordinal);
       C2 10607        RETURN;
       C4 10608      IFEND;
       C4 10609
       C4 10610      IF syv$perf_keypoints_enabled.swapping_keypoints THEN
       D0 10611        kt.s := ijle_p^.system_supplied_name (16, 4);
       DE 10612        #KEYPOINT (osk$performance, osk$m * kt.f1, ptk$swapin_job_name_1);
       F2 10613        #KEYPOINT (osk$performance, osk$m * ((kt.f2 * 256) + ajl_ordinal), ptk$swapin_job_name_2);
      10C 10614      IFEND;
      10C 10615
      10C 10616      IF (ijle_p^.swap_status <= jmc$iss_allocate_swap_file) THEN
      :.6 10617        mmv$reassignable_page_frames.swapout_io_not_initiated :=
      132 10618              mmv$reassignable_page_frames.swapout_io_not_initiated - ijle_p^.swap_data.swapped_job_page_count +
      132 10619              ijle_p^.job_fixed_contiguous_pages;
      132 10620      ELSEIF (ijle_p^.swap_status = jmc$iss_wait_job_io_complete) OR
      13E 10621           (ijle_p^.swap_status = jmc$iss_wait_allocate_sfd) THEN
      13E 10622        mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon -
      156 10623              ijle_p^.swap_data.swapped_job_page_count + ijle_p^.job_fixed_contiguous_pages;
      156 10624      IFEND;
      156 10625
      156 10626 { Swap status is advanced to executing.
      156 10627
      156 10628      complete_swapin (ijl_ordinal, ijle_p, ijle_p^.swap_data.swapped_job_entry.available_modified_page_count);
      166 10629
      166 10630    PROCEND swapin_before_io;
```


NOS/VE js : monitor mode job swapper
SWAPIN_AFTER_IO

```
        0 10632
        0 10633    PROCEDURE swapin_after_io
        0 10634      (     ijl_ordinal: jmt$ijl_ordinal;
        0 10635            ijle_p: ^jmt$initiated_job_list_entry);
        0 10636
        0 10637      VAR
        0 10638        ajl_ordinal: jmt$ajl_ordinal,
        0 10639        status: syt$monitor_status;
        0 10640
        0 10641      jmp$assign_ajl_entry (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$swapping_ajl, FALSE {must assign} ,
       38 10642           ajl_ordinal, status);
       38 10643      IF NOT status.normal THEN
       40 10644        trace (jsc$ti_no_ajlo_swapin_after_io, 1);
       40 10645        IF (ijle_p^.swap_status = jmc$iss_swapped_io_cannot_init) THEN
       58 10646          mmv$reassignable_page_frames.swapout_io_cannot_initiate :=
       58 10647                mmv$reassignable_page_frames.swapout_io_cannot_initiate +
       58 10648                ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
       58 10649          jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_cannot_init);
       8E 10650        ELSEIF (ijle_p^.swap_status = jmc$iss_swapped_io_complete) THEN
       96 10651          mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now +
       96 10652                ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
       96 10653          jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_completed);
       CC 10654        ELSEIF (ijle_p^.swap_status = jmc$iss_swapout_io_initiated) OR
       DA 10655             (ijle_p^.swap_status = jmc$iss_wait_swapout_io_init) THEN
       DA 10656          mmv$reassignable_page_frames.swapout_io_not_initiated :=
       DA 10657                mmv$reassignable_page_frames.swapout_io_not_initiated +
       DA 10658                ijle_p^.swap_data.swapped_job_page_count - ijle_p^.job_fixed_contiguous_pages;
       DA 10659          jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_not_init);
      10C 10660          advance_swap_state (ijle_p, jmc$iss_swapped_no_io);
      126 10661        ELSE
      126 10662          mtp$error_stop ('BAD SWAP STATUS-SWAPIN AFTER IO');
      146 10663        IFEND;
      146 10664        jmp$reset_job_to_swapped_out (ijl_ordinal);
      156 10665        RETURN;
      158 10666      IFEND;
      158 10667
      158 10668      IF syv$perf_keypoints_enabled.swapping_keypoints THEN
      164 10669        kt.s := ijle_p^.system_supplied_name (16, 4);
      172 10670        #KEYPOINT (osk$performance, osk$m * kt.f1, ptk$swapin_job_name_1);
      186 10671        #KEYPOINT (osk$performance, osk$m * ((kt.f2 * 256) + ajl_ordinal), ptk$swapin_job_name_2);
      1A0 10672      IFEND;
      1A0 10673
      1A0 10674 { Swap status is advanced to executing.
      1A0 10675
      1A0 10676      reset_sdt_xcb_tables (ijl_ordinal, ijle_p, FALSE, FALSE);
      1B2 10677
      1B2 10678    PROCEND swapin_after_io;
```

NOS/VE js : monitor mode job swapper
UPDATE_PROCESSOR_SELECTIONS

```
        o 10680
        o 10681 {
        o 10682 { Purpose:
        o 10683 {   This procedure is called before swapin of a job is complete in order to readjust the processors
        o 10684 {   which a job has selected and on which its tasks will execute.  Processor selections will be
        o 10685 {   adjusted IF AND ONLY IF the task has, as its processor selections, only those processors which
        o 10686 {   are not on.
        o 10687 {
        o 10688
        o 10689     PROCEDURE update_processor_selections
        o 10690       (     ijle_p: ^jmt$initiated_job_list_entry;
        o 10691             ijl_ordinal: jmt$ijl_ordinal);
        o 10692
        o 10693       VAR
        o 10694         xcb_p: ^ost$execution_control_block,
        o 10695         xcb_state: tmt$find_next_xcb_state;
        o 10696
        o 10697
        o 10698       tmp$find_next_xcb (tmc$fnx_swapping_job, ijle_p, ijl_ordinal, xcb_state, xcb_p);
       2E 10699
       2E 10700       WHILE xcb_p <> NIL DO
       3E 10701         IF (xcb_p^.processor_selections * mtv$scb.processors_logically_on) = $ost$processor_id_set [] THEN
       54 10702           xcb_p^.processor_selections := mtv$scb.processors_logically_on;
       58 10703         IFEND;
       58 10704         tmp$find_next_xcb (tmc$fnx_continue, NIL, jmv$null_ijl_ordinal, xcb_state, xcb_p);
       88 10705       WHILEND;
       9C 10706
       9C 10707     PROCEND update_processor_selections;
```


NOS/VE js : monitor mode job swapper
UPDATE_SERVER_FILES

```
        o 10709
        o 10710     PROCEDURE update_server_files
        o 10711       (     ijle_p: ^jmt$initiated_job_list_entry;
        o 10712             ijl_ordinal: jmt$ijl_ordinal);
        o 10713
        o 10714       VAR
        o 10715         fde_p: gft$file_desc_entry_p,
        o 10716         msg: string (70),
        o 10717         next_pfti: mmt$page_frame_index,
        o 10718         page_status: gft$page_status,
        o 10719         pfti: mmt$page_frame_index;
        o 10720
        o 10721       pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.bkw;
        4 10722
        4 10723 { It is not necessary to clear the valid bit before checking the modified bit in this case; the job is
        4 10724 { in the process of swapping in, so nothing else can be referencing the pages.
        4 10725
        4 10726       WHILE pfti <> 0 DO
        C 10727         next_pfti := mmv$pft_p^ [pfti].link.bkw;
        C 10728         IF mmv$pft_p^ [pfti].aste_p^.sfid.residence <> gfc$tr_system_wait_recovery THEN
       3A 10729           gfp$mtr_get_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijle_p, fde_p);
       9E 10730           IF fde_p^.media = gfc$fm_served_file THEN
       AC 10731             dfp$fetch_page_status (fde_p, 0, page_status);
       CA 10732             IF (page_status = gfc$ps_server_terminated) OR ((page_status = gfc$ps_volume_unavailable) AND
      10E 10733                   (NOT mmv$pt_p^ [mmv$pft_p^ [pfti].pti].m)) THEN
      10E 10734
      10E 10735 { If the server is terminated or server is unavailable and we are reading, delete the page.
      10E 10736
      10E 10737               mmp$delete_pt_entry (pfti, TRUE);
      122 10738               mmp$relink_page_frame (pfti, mmc$pq_free);
      136 10739             IFEND;
      136 10740           IFEND;
      136 10741         IFEND;
      136 10742         pfti := next_pfti;
      136 10743       WHILEND;
      13E 10744
      13E 10745 { Debug display
      13E 10746
      13E 10747       IF dfv$file_server_debug_enabled THEN
      14A 10748         IF (ijle_p^.terminate_access_work = $dft$mainframe_set []) AND
      15A 10749               (ijle_p^.inhibit_access_work = $dft$mainframe_set []) THEN
      15A 10750           msg := ' Job XXXXXXXXXXXXXXXXXXX swap in - server inactivation.';
      168 10751           msg (6, 19) := ijle_p^.system_supplied_name;
      172 10752           dpp$display_error (msg);
      18E 10753         IFEND;
      18E 10754         IF ijle_p^.inhibit_access_work <> $dft$mainframe_set [] THEN
      196 10755           msg := ' Job XXXXXXXXXXXXXXXXXXX swap in - server inhibit access.';
      1A4 10756           msg (6, 19) := ijle_p^.system_supplied_name;
      1AE 10757           dpp$display_error (msg);
      1CA 10758         IFEND;
      1CA 10759         IF ijle_p^.terminate_access_work <> $dft$mainframe_set [] THEN
      1D2 10760           msg := ' Job XXXXXXXXXXXXXXXXXXX swap in - server terminate access.';
      1E0 10761           msg (6, 19) := ijle_p^.system_supplied_name;
      1EA 10762           dpp$display_error (msg);
      206 10763         IFEND;
```

NOS/VE js : monitor mode job swapper
UPDATE_SERVER_FILES

```
206 10764      IFEND;
206 10765
206 10766      IF (ijle_p^.terminate_access_work = $dft$mainframe_set []) AND
216 10767            (ijle_p^.inhibit_access_work = $dft$mainframe_set []) THEN
216 10768
216 10769 { There is no need to change the access state.
216 10770
216 10771        RETURN;
218 10772      IFEND;
218 10773      dfp$set_task_segment_state (tmc$fnx_swapping_job, ijle_p, ijl_ordinal, ijle_p^.inhibit_access_work,
242 10774            ijle_p^.terminate_access_work);
242 10775
242 10776 { Dont clear inhibit - let it be cleared by either job recovery
242 10777 { or by the job when it detects that the server is not longer inactive.
242 10778
242 10779      ijle_p^.terminate_access_work := $dft$mainframe_set [];
242 10780    PROCEND update_server_files;
  0 10781
```


NOS/VE js : monitor mode job swapper
[XDCL] jsp$free_swap_resident_job

```
  0 10783
  0 10784 { PURPOSE:
  0 10785 {   This procedure advances the swapout of a swap resident (swapped_io_complete) job so
  0 10786 {   that its memory will be freed.
  0 10787 { DESIGN:
  0 10788 {   An entry status of swapin_in_progress indicates that the swap resident job has just
  0 10789 {   been readied on another processor and is in the swapping queue to swap in.  Memory
  0 10790 {   manager needs the memory that the job is holding right now, however, so the job must
  0 10791 {   be reset to swapped out so that it will swap in through the job mode scheduler path.
  0 10792 {   Because dispatcher can ready tasks and swapin jobs in monitor asynchronously, the ptl
  0 10793 {   lock must be set during the advance swap.  With the ptl lock set, dispatcher cannot
  0 10794 {   swapin a job through jmp$ready_task_in_swapped_job while the advance swap out is
  0 10795 {   going on.
  0 10796
  0 10797    PROCEDURE [XDCL] jsp$free_swap_resident_job
  4 10798      (     swap_resident_ijlo: jmt$ijl_ordinal;
  4 10799            swap_resident_ijle_p: ^jmt$initiated_job_list_entry);
  4 10800
  4 10801      jsp$relink_swap_queue (swap_resident_ijlo, swap_resident_ijle_p, jsc$isqi_swapping);
 26 10802
 26 10803      tmp$set_lock (tmv$ptl_lock);
 5E 10804
 5E 10805      IF swap_resident_ijle_p^.entry_status = jmc$ies_swapin_in_progress THEN
 6C 10806        trace (jsc$ti_free_readied_s2_job, 1);
 6C 10807        jmp$reset_job_to_swapped_out (swap_resident_ijlo);
 8E 10808      IFEND;
 8E 10809      jsp$monitor_advance_swap (swap_resident_ijlo);
 A2 10810
 A2 10811      tmp$clear_lock (tmv$ptl_lock);
 DA 10812
 DA 10813    PROCEND jsp$free_swap_resident_job;
  0 10814
```

NOS/VE js : monitor mode job swapper
JSP$IDLE_TASKS_COMPLETE

```
       0 10816
       0 10817   PROCEDURE [XDCL] jsp$idle_tasks_complete
       0 10818     (     ijl_ordinal: jmt$ijl_ordinal);
       0 10819
       0 10820 {
       0 10821 {     The purpose of this procedure is to record that all tasks are idled for a job being
       0 10822 {  swapped out.  The swapout can now be advanced.
       0 10823 {
       0 10824 {  NOTE: It is possible that this procedure is executing in more than 1 cpu simultaneously.
       0 10825 {
       0 10826
       0 10827     VAR
       0 10828       ijle_p: ^jmt$initiated_job_list_entry;
       0 10829
       0 10830
       0 10831     jmp$get_ijle_p (ijl_ordinal, ijle_p);
       4 10832
       4 10833     IF (ijle_p^.swap_status = jmc$iss_idle_tasks_initiated) THEN
      34 10834       ijle_p^.next_swap_status := jmc$iss_job_idle_tasks_complete;
      34 10835       ijle_p^.delayed_swapin_work := $jmt$delayed_swapin_work [];
      34 10836
      34 10837 { Dont clear inhibit - let it be cleared by either server job recovery
      34 10838 { or by the job when it detects that the server is not longer inactive.
      34 10839
      34 10840       ijle_p^.terminate_access_work := $dft$mainframe_set [];
      34 10841       set_swapping_event (jsc$se_immediate);
      82 10842     IFEND;
      82 10843
      82 10844   PROCEND jsp$idle_tasks_complete;
```


NOS/VE js : monitor mode job swapper
JSP$IO_COMPLETE

```
       0 10846
       0 10847   PROCEDURE [XDCL] jsp$io_complete
       4 10848     (     ijle_p: ^jmt$initiated_job_list_entry);
       4 10849
       4 10850 {
       4 10851 {     The purpose of this procedure is to record that swap io has completed and the swap can
       4 10852 {  now be advanced.
       4 10853 {
       4 10854 {  NOTE: It is possible that this procedure is executing in more than 1 cpu simultaneously.
       4 10855 {
       4 10856
       4 10857
       4 10858     ijle_p^.notify_swapper_when_io_complete := FALSE;
       4 10859
       4 10860     CASE ijle_p^.swap_status OF
      2C 10861     = jmc$iss_wait_job_io_complete =
      2C 10862       ijle_p^.next_swap_status := jmc$iss_job_io_complete;
      36 10863     = jmc$iss_swapout_io_initiated =
      36 10864       ijle_p^.next_swap_status := jmc$iss_swapout_io_complete;
      42 10865     = jmc$iss_swapin_io_initiated =
      42 10866       ijle_p^.next_swap_status := jmc$iss_swapin_io_complete;
      4E 10867     ELSE
      4E 10868       RETURN;
      50 10869     CASEND;
      50 10870
      50 10871     set_swapping_event (jsc$se_immediate);
      6E 10872
      6E 10873   PROCEND jsp$io_complete;
```

NOS/VE js : monitor mode job swapper
JSP$LONG_WAIT_AGING

```
      0 10875
      0 10876 {
      0 10877 { The purpose of this procedure is to age the working set of a job going into LONG WAIT.
      0 10878 {
      0 10879
      0 10880   PROCEDURE [XDCL] jsp$long_wait_aging
      0 10881     (     ijle_p: ^jmt$initiated_job_list_entry);
      0 10882
      0 10883     VAR
      0 10884       cptime: integer,
      0 10885       fde_p: gft$file_desc_entry_p,
      0 10886       ijl_ordinal: jmt$ijl_ordinal,
      0 10887       initial_rtc: integer,
      0 10888       jcb_p: ^jmt$job_control_block,
      0 10889       maximum_pages_to_swap: integer,
      0 10890       minimum_working_set: jmt$working_set_size,
      0 10891       modified_pages_removed: integer,
      0 10892       page_age_limit: integer,
      0 10893       pfti: mmt$page_frame_index,
      0 10894       queueid: mmt$page_frame_queue_id,
      0 10895       segment_number: ost$segment,
      0 10896       total_pages_removed: integer;
      0 10897
      0 10898     #KEYPOINT (osk$entry, 0, jsk$long_wait_aging);
      8 10899
      8 10900     jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, 0);
      8 10901     initial_rtc := ijle_p^.statistics.ready_task_count;
      8 10902
      8 10903     IF mmv$aging_algorithm >= 4 THEN
     36 10904       cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode;
     3E 10905     ELSE
     3E 10906       cptime := ijle_p^.statistics.cp_time.time_spent_in_job_mode +
     4A 10907             ijle_p^.statistics.cp_time.time_spent_in_mtr_mode;
     4A 10908     IFEND;
     4A 10909
     4A 10910     trace (jsc$ti_lwa, 1);
     4A 10911
     4A 10912     IF cptime > (jcb_p^.cptime_next_age_working_set + 2 * jcb_p^.page_aging_interval) THEN
     6C 10913       trace (jsc$ti_lwa_cp_age, 1);
     6C 10914       mmp$age_job_working_set (ijle_p, jcb_p);
     8E 10915     IFEND;
     8E 10916
     8E 10917     IF jsv$free_working_set_on_swapout THEN
     96 10918       page_age_limit := 0;
     96 10919       minimum_working_set := 0;
     9E 10920     ELSE { This is the usual case.  Freeing the working set is for test purposes. }
     9E 10921       page_age_limit := mmv$swapping_aic;
     9E 10922       minimum_working_set := jcb_p^.min_working_set_size;
     B4 10923     IFEND;
     B4 10924
     B4 10925     mmp$remove_stale_pages (ijle_p^.job_page_queue_list [mmc$pq_job_working_set], page_age_limit, jcb_p,
     F2 10926           ijle_p, mmc$pq_avail_modified, minimum_working_set, modified_pages_removed, total_pages_removed);
     F2 10927
     F2 10928     trace (jsc$ti_lwa_stale_pages_rem, total_pages_removed);
     F2 10929     trace (jsc$ti_lwa_stale_mod_pages_rem, modified_pages_removed);
```


NOS/VE js : monitor mode job swapper
JSP$LONG_WAIT_AGING

```
     F2 10930
     F2 10931     IF ijle_p^.task_created_after_last_swap THEN
    116 10932       maximum_pages_to_swap := jsv$max_pages_first_swap_task;
    11E 10933     ELSE
    11E 10934       maximum_pages_to_swap := jsv$maximum_pages_to_swap;
    122 10935     IFEND;
    122 10936
    122 10937     IF (ijle_p^.job_page_queue_list [mmc$pq_job_working_set].count > maximum_pages_to_swap) THEN
    12A 10938       mmp$trim_job_working_set (ijle_p, jcb_p, TRUE); {true= trim_to_swap_size
    146 10939     IFEND;
    146 10940
    146 10941     ijle_p^.task_created_after_last_swap := FALSE;
    146 10942
    146 10943     IF ijle_p^.statistics.ready_task_count > initial_rtc THEN
    152 10944       trace (jsc$ti_lwa_ready_task, 1);
    15C 10945     IFEND;
    15C 10946
    15C 10947 { Update the MAP_PURGE_TIMESTAMP. Since long wait aging may have cleared page table
    15C 10948 { 'used' bits and NOT purge the page map, we have to insure that the map is purged before
    15C 10949 { the job is next allowed to run. Although the map could be purged at this point, it is
    15C 10950 { defered until the job is swapped in. Usually something else will have purged the map by
    15C 10951 { this time and no purge will be required.
    15C 10952
    15C 10953     ijle_p^.age_purge_timestamp := #FREE_RUNNING_CLOCK (0);
    162 10954
    162 10955 { Purge maps now in case we decided not to swap out.
    162 10956
    162 10957     mmp$conditional_purge_all_map (ijle_p^.age_purge_timestamp);
    192 10958
    192 10959 { The following code will count the pages being swapped out and determine the segment that the
    192 10960 { page belongs to. Segments greater than or equal to 40(16) are combined and output as pages
    192 10961 { of segment 40(16).
    192 10962
    192 10963     IF jsv$enable_swap_file_statistics THEN
    19A 10964       pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.bkw;
    19A 10965       WHILE pfti <> 0 DO
    1A2 10966         gfp$mtr_get_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijle_p, fde_p);
    224 10967         IF fde_p^.last_segment_number >= 40(16) THEN
    234 10968           segment_number := 40(16);
    238 10969         ELSE
    238 10970           segment_number := fde_p^.last_segment_number;
    23A 10971         IFEND;
    23A 10972         jsv$swap_file_statistics.total_pages_per_segment [segment_number] :=
    23A 10973               jsv$swap_file_statistics.total_pages_per_segment [segment_number] + 1;
    23A 10974         pfti := mmv$pft_p^ [pfti].link.bkw;
    23A 10975       WHILEND;
    26C 10976       jsv$swap_file_statistics.total_pages_per_segment [3] :=
    26C 10977             ijle_p^.job_page_queue_list [mmc$pq_job_fixed].count;
    26C 10978       jsv$swap_file_statistics.total_swaps := jsv$swap_file_statistics.total_swaps + 1;
    282 10979     IFEND;
    282 10980
    282 10981
    282 10982     IF syv$perf_keypoints_enabled.swapping_keypoints THEN
    28E 10983       pfti := ijle_p^.job_page_queue_list [mmc$pq_job_working_set].link.bkw;
    28E 10984       WHILE pfti <> 0 DO
```

NOS/VE js : monitor mode job swapper
JSP$LONG_WAIT_AGING

```
29A 10985          gfp$mtr_get_fde_p (mmv$pft_p^ [pfti].aste_p^.sfid, ijle_p, fde_p);
318 10986          #KEYPOINT (osk$performance, osk$m * fde_p^.last_segment_number, ptk$swapping_segment);
328 10987          #KEYPOINT (osk$performance, osk$m * (mmv$pft_p^ [pfti].sva.offset DIV osv$page_size),
35C 10988                ptk$swapping_page_number);
35C 10989          pfti := mmv$pft_p^ [pfti].link.bkw;
35C 10990        WHILEND;
364 10991        #KEYPOINT (osk$performance, osk$m * ijle_p^.job_page_queue_list [mmc$pq_job_fixed].count,
374 10992             ptk$swapping_job_fixed);
374 10993        #KEYPOINT (osk$performance, osk$m * modified_pages_removed, ptk$swapping_modified_pages);
380 10994        #KEYPOINT (osk$performance, osk$m * total_pages_removed, ptk$swapping_removed_pages);
38C 10995        ijl_ordinal := jmv$ajl_p^ [ijle_p^.ajl_ordinal].ijl_ordinal;
38C 10996        #KEYPOINT (osk$performance, osk$m * (ijl_ordinal.block_number * 32 + ijl_ordinal.block_index),
3BC 10997             ptk$swapping_ijl_ordinal);
3BC 10998      IFEND;
3BC 10999
3BC 11000      #KEYPOINT (osk$exit, 0, jsk$long_wait_aging);
3C0 11001
3C0 11002    PROCEND jsp$long_wait_aging;
```


NOS/VE js : monitor mode job swapper
JSP$MONITOR_ADVANCE_SWAP

```
  0 11004
  0 11005    PROCEDURE [XDCL] jsp$monitor_advance_swap
  0 11006      (    ijl_ordinal: jmt$ijl_ordinal);
  0 11007
  0 11008  {
  0 11009  {      The purpose of this procedure is to advance the swap of jobs that are
  0 11010  {   in one of the swapped but memory resident queues.
  0 11011  {
  0 11012  {      NOTE: It is the responsibility of the caller to update the swap queue
  0 11013  {   statistics.
  0 11014  {
  0 11015  {   NOTE: This procedure is entered serially if running with multiple cpu's.
  0 11016  {
  0 11017
  0 11018      VAR
  0 11019        ijle_p: ^jmt$initiated_job_list_entry,
  0 11020        poll_swapping: boolean,
  0 11021        status: syt$monitor_status;
  0 11022
  0 11023
  0 11024      jmp$get_ijle_p (ijl_ordinal, ijle_p);
  4 11025
  4 11026      jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
 46 11027
 46 11028  {   This has to call advance_swap directly because memory manager may need memory and it expects to
 46 11029  {   get it immediately.
 46 11030
 46 11031      advance_swap (ijl_ordinal, ijle_p, poll_swapping, status);
 5E 11032
 5E 11033      IF poll_swapping THEN
 66 11034        set_swapping_event (jsc$se_polling);
 80 11035      IFEND;
 80 11036
 80 11037    PROCEND jsp$monitor_advance_swap;
```

NOS/VE js : monitor mode job swapper
TRACE BUFFER FOR SCHEDULER SWAPPING REQUESTS

```
        0 11040
        0 11041   CONST
        0 11042     num_sched_swapping_calls = 80;
        0 11043
        0 11044   TYPE
        0 11045     jst$swapping_request_type = (jsc$sc_swapout_job_mode, jsc$sc_swapout_mtr_mode, jsc$sc_swapin_job_mode,
        0 11046           jsc$sc_swapin_mtr_mode, jsc$sc_swapin_mtr_direct);
        0 11047
        0 11048   VAR
        0 11049     jsv$sched_swapping_requests: [XDCL] record
        0 11050       next_index: integer,
        0 11051       sched_requests: array [0 .. num_sched_swapping_calls - 1] of record
        0 11052         request_type: ALIGNED [0 MOD 16] jst$swapping_request_type,
        0 11053         ijlo: jmt$ijl_ordinal,
        0 11054         timestamp: ost$free_running_clock,
        0 11055       recend,
        0 11056     recend;
        0 11057
        0 11058   PROCEDURE [INLINE] sched_trace
        0 11059     (     request_type: jst$swapping_request_type;
        0 11060           ijlo: jmt$ijl_ordinal);
        0 11061
        0 11062     VAR
        0 11063       i: integer;
        0 11064
        0 11065     i := jsv$sched_swapping_requests.next_index;
        0 11066     jsv$sched_swapping_requests.next_index := (i + 1) MOD num_sched_swapping_calls;
        0 11067     jsv$sched_swapping_requests.sched_requests [i].request_type := request_type;
        0 11068     jsv$sched_swapping_requests.sched_requests [i].ijlo := ijlo;
        0 11069     jsv$sched_swapping_requests.sched_requests [i].timestamp := #FREE_RUNNING_CLOCK (0);
        0 11070
        0 11071   PROCEND sched_trace;
        0 11072
```


NOS/VE js : monitor mode job swapper
JSP$MONITOR_SWAP_IN

```
        0 11075
        0 11076   PROCEDURE [XDCL] jsp$monitor_swap_in
        0 11077     (     ijl_ordinal: jmt$ijl_ordinal);
        0 11078
        0 11079
        0 11080 {
        0 11081 {   The purpose of this procedure is to swap a job in that is in long wait.
        0 11082 { The job may be in the long wait queue, swapped out or in some intermediate
        0 11083 { state.  The job is swapped in from whatever state it is in.
        0 11084 {
        0 11085 {         JSP$MONITOR_SWAP_IN (IJL_ORDINAL)
        0 11086 {
        0 11087 { IJL_ORDINAL: (input) This parameter specifies the index in the ijl table
        0 11088 {         of the entry for this job.
        0 11089 {
        0 11090
        0 11091
        0 11092     VAR
        0 11093       ajl_ordinal: jmt$ajl_ordinal,
        0 11094       ijle_p: ^jmt$initiated_job_list_entry,
        0 11095       jcb_p: ^jmt$job_control_block,
        0 11096       status: syt$monitor_status;
        0 11097
        0 11098     #KEYPOINT (osk$entry, 0, jsk$monitor_swap_in);
        8 11099
        8 11100     jmp$get_ijle_p (ijl_ordinal, ijle_p);
        8 11101
        8 11102
        8 11103     IF ijle_p^.swap_status = jmc$iss_swapped_no_io THEN
       3A 11104       sched_trace (jsc$sc_swapin_mtr_direct, ijl_ordinal);
       7A 11105       trace (jsc$ti_swapin_mtr_direct, 1);
       7A 11106
       7A 11107 {  We could just call swapin_before_io here, but for performance reasons we
       7A 11108 {  will inline the necessary code instead.
       7A 11109 {
       7A 11110 {  ***  duplicated in swapin_before_io  ***
       7A 11111
       7A 11112         jmp$assign_ajl_with_lock (ijle_p^.job_fixed_asid, ijl_ordinal, jmc$swapping_ajl, FALSE {must assign} ,
       BA 11113               ajl_ordinal, status);
       BA 11114         IF NOT status.normal THEN
       C2 11115           trace (jsc$ti_no_ajlo_mtr_swapin, 1);
       C2 11116           jmp$change_ijl_entry_status (ijle_p, jmc$ies_swapin_in_progress);
      11C 11117           jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
      13C 11118           set_swapping_event (jsc$se_immediate);
      15A 11119           RETURN;
      17A 11120         IFEND;
      17A 11121
      17A 11122         IF syv$perf_keypoints_enabled.swapping_keypoints THEN
      186 11123           kt.s := ijle_p^.system_supplied_name (16, 4);
      194 11124           #KEYPOINT (osk$performance, osk$m * kt.f1, ptk$swapin_job_name_1);
      1A8 11125           #KEYPOINT (osk$performance, osk$m * ((kt.f2 * 256) + ajl_ordinal), ptk$swapin_job_name_2);
      1C2 11126         IFEND;
      1C2 11127
      1C2 11128         mmv$reassignable_page_frames.swapout_io_not_initiated :=
      1C2 11129               mmv$reassignable_page_frames.swapout_io_not_initiated - ijle_p^.swap_data.swapped_job_page_count +
```

NOS/VE js : monitor mode job swapper
JSP$MONITOR_SWAP_IN

```
1C2 11130              ijle_p^.job_fixed_contiguous_pages;
1C2 11131
1C2 11132 {      *** duplicated in complete_swapin ***
1C2 11133
1C2 11134        jcb_p := #ADDRESS (1, mtc$job_fixed_segment + ijle_p^.ajl_ordinal, 0);
1C2 11135        jcb_p^.next_cyclic_aging_time := #FREE_RUNNING_CLOCK (0) + jcb_p^.next_cyclic_aging_time;
1F8 11136
1F8 11137 {        *** duplicated in restart_idled_tasks ***
1F8 11138
1F8 11139        jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_null);
220 11140        advance_swap_state (ijle_p, jmc$iss_executing);
232 11141
232 11142        jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_in_memory);
286 11143
286 11144 {  Update counts if the job has reserved memory through the mmp$assign_pages request
286 11145
286 11146        IF ijle_p^.memory_reserve_request.requested_page_count > 0 THEN
292 11147          IF (mmv$reassignable_page_frames.now - mmv$aggressive_aging_level_2) >
2A4 11148                ijle_p^.memory_reserve_request.requested_page_count THEN
2A4 11149            ijle_p^.memory_reserve_request.reserved_page_count :=
2A4 11150                  ijle_p^.memory_reserve_request.reserved_page_count +
2A4 11151                  ijle_p^.memory_reserve_request.requested_page_count;
2A4 11152            mmv$reserved_page_count := mmv$reserved_page_count +
2C0 11153                  ijle_p^.memory_reserve_request.requested_page_count;
2C0 11154          ELSE
2C0 11155            trace (jsc$ti_reserve_memory_failed, 1);
2CA 11156          IFEND;
2CA 11157          ijle_p^.memory_reserve_request.requested_page_count := 0;
2D2 11158        IFEND;
2D2 11159
2D2 11160        IF syv$perf_keypoints_enabled.swapping_stack_trace THEN
2DA 11161          tmp$monitor_flag_job_tasks (syc$mf_for_keypoint_traceback, ijle_p);
2F4 11162        IFEND;
2F4 11163
2F4 11164        tmp$restart_idled_tasks (ijle_p^.ajl_ordinal);
310 11165      ELSE
310 11166        sched_trace (jsc$sc_swapin_mtr_mode, ijl_ordinal);
34E 11167        trace (jsc$ti_swapin_from_mtr_mode, 1);
34E 11168        jmp$change_ijl_entry_status (ijle_p, jmc$ies_swapin_in_progress);
3A8 11169        jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
3CA 11170        set_swapping_event (jsc$se_immediate);
406 11171      IFEND;
406 11172
406 11173      #KEYPOINT (osk$exit, 0, jsk$monitor_swap_in);
40A 11174
40A 11175    PROCEND jsp$monitor_swap_in;
```


NOS/VE js : monitor mode job swapper
JSP$MONITOR_SWAP_OUT

```
  0 11178
  0 11179    PROCEDURE [XDCL] jsp$monitor_swap_out
  0 11180      (     ijl_ordinal: jmt$ijl_ordinal);
  0 11181
  0 11182
  0 11183 {
  0 11184 {    The purpose of this procedure is to prepare the specified job
  0 11185 { for swapout to mass storage. If memory is needed the swapout IO
  0 11186 { will be initiated and the memory freed.  How far the swap
  0 11187 { progresses is determined by memory thresholds.
  0 11188 {
  0 11189 {              JSP$MONITOR_SWAP_OUT (IJL_ORDINAL)
  0 11190 {
  0 11191 { IJL_ORDINAL: (input) This parameter is the 'ijl_ordinal' of the job
  0 11192 {         being swapped.
  0 11193 {
  0 11194
  0 11195      VAR
  0 11196        ijle_p: ^jmt$initiated_job_list_entry,
  0 11197        initiate_swapout_io: boolean,
  0 11198        job_page_count: mmt$page_frame_index,
  0 11199        queue_id: mmt$job_page_queue_index;
  0 11200
  0 11201      sched_trace (jsc$sc_swapout_mtr_mode, ijl_ordinal);
 48 11202
 48 11203      jmp$get_ijle_p (ijl_ordinal, ijle_p);
 48 11204      IF ijle_p^.swap_queue_link.queue_id = jsc$isqi_null THEN
 76 11205        trace (jsc$ti_swapout_from_mtr_mode, 1);
 76 11206
 76 11207        jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_swapped);
 CE 11208
 CE 11209 { ** This code is combined from code for job mode swap out requests and code in advance swap for
 CE 11210 { ** swap state jmc$iss_job_idle_tasks_complete - TJ.
 CE 11211
 CE 11212        IF syv$perf_keypoints_enabled.swapping_keypoints THEN
 DA 11213          #KEYPOINT (osk$performance, osk$m * ijle_p^.ajl_ordinal, ptk$ajl_for_swap_out);
 EA 11214        IFEND;
 EA 11215        ijle_p^.swap_io_control.spd_index := LOWERVALUE (mmt$page_frame_index);
 EA 11216        ijle_p^.delayed_swapin_work := $jmt$delayed_swapin_work [];
 EA 11217
 EA 11218 { Dont clear inhibit - let it be cleared by either server job recovery
 EA 11219 { or by the job when it detects that the server is not longer inactive.
 EA 11220
 EA 11221        ijle_p^.terminate_access_work := $dft$mainframe_set [];
 EA 11222
 EA 11223 { Swap_data.timestamp is still the time when the job completed swapin.  Swapin to swapout is residence time.
 EA 11224
 EA 11225        ijle_p^.swap_data.swapout_timestamp := #FREE_RUNNING_CLOCK (0);
100 11226
100 11227 { To prevent the situation of a task executing after monitor_swap_out has been called,
100 11228 { dispatcher idled tasks before calling scheduler/swapper to swapout the job for long
100 11229 { wait.  We advance the swap status of the job to swapped_no_io.
100 11230
100 11231        jmp$free_ajl_with_lock (ijle_p, jmc$swapping_ajl);
118 11232
```

NOS/VE js : monitor mode job swapper
JSP$MONITOR_SWAP_OUT

```
118 11233 {  Set close approximation of swapped job page count for job mode job scheduler.  The count is also
118 11234 {  used for the service class statistics.
118 11235
118 11236       calculate_swapped_pages (ijle_p);
14A 11237       jsv$swap_file_page_count.swap_count := jsv$swap_file_page_count.swap_count + 1;
14A 11238       jsv$swap_file_page_count.page_count := jsv$swap_file_page_count.page_count +
14A 11239             ijle_p^.swap_data.swapped_job_page_count;
14A 11240
14A 11241       tmp$set_lock (jmv$service_class_stats_lock);
1B6 11242       jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.residence_time :=
1B6 11243             jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
1B6 11244             residence_time + (ijle_p^.swap_data.swapout_timestamp - ijle_p^.swap_data.timestamp);
1B6 11245       jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.swapped_pages :=
1B6 11246             jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
1B6 11247             swapped_pages + ijle_p^.swap_data.swapped_job_page_count;
1B6 11248       tmp$clear_lock (jmv$service_class_stats_lock);
228 11249
228 11250
228 11251       initiate_swapout_io := ((mmv$reassignable_page_frames.now + mmv$reassignable_page_frames.soon) <=
250 11252             jmv$long_wait_swap_threshold) OR NOT jsv$enable_swap_resident_no_io;
250 11253
250 11254 { ** End duplicate code **
250 11255
250 11256       IF NOT initiate_swapout_io THEN
254 11257         advance_swap_state (ijle_p, jmc$iss_swapped_no_io);
266 11258         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapped_io_not_init);
28C 11259       ELSE
28C 11260         advance_swap_state (ijle_p, jmc$iss_flush_am_pages);
29E 11261         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
2BE 11262         set_swapping_event (jsc$se_immediate);
2FA 11263       IFEND;
2FC 11264
2FC 11265     ELSE
2FC 11266       mtp$error_stop ('JS - jsp$monitor_swap_out called for job not in null queue.');
31C 11267     IFEND;
31C 11268
31C 11269   PROCEND jsp$monitor_swap_out;
```


NOS/VE js : monitor mode job swapper
JSP$MTR_JOB_SWAPPING_REQUESTS

```
  0 11272
  0 11273   PROCEDURE [XDCL] jsp$mtr_job_swapping_requests
  0 11274     (VAR request_block: jst$rb_job_swapping_functions);
  0 11275
  0 11276 {
  0 11277 {  The purpose of this procedure is to process job swapping monitor requests from the job mode job
  0 11278 {  swapper.  The JOB SCHEDULER task is executing all the swapping requests (but not set_delayed_swapin_work).
  0 11279 {
  0 11280 {  NOTE: This procedure is entered serially if running with multiple cpu's.
  0 11281 {
  0 11282
  0 11283     VAR
  0 11284       ijle_p: ^jmt$initiated_job_list_entry,
  0 11285       ijl_ordinal: jmt$ijl_ordinal,
  0 11286       poll_swapping: boolean;
  0 11287
  0 11288     #KEYPOINT (osk$entry, 0, jsk$mtr_job_swapping_requests);
  8 11289
  8 11290     request_block.status.normal := TRUE;
  8 11291     poll_swapping := TRUE;
  8 11292     ijl_ordinal := request_block.ijl_ordinal;
  8 11293     jmp$get_ijle_p (ijl_ordinal, ijle_p);
  8 11294
  8 11295 {  Process the job swapping subfunctions.
  8 11296
  8 11297     CASE request_block.subfunction OF
 70 11298     = jsc$jss_swap_job_in =
 70 11299       sched_trace (jsc$sc_swapin_job_mode, ijl_ordinal);
 82 11300       trace (jsc$ti_swapin_from_job_mode, 1);
 82 11301
 82 11302       jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
 DC 11303
 DC 11304 { Set PTL lock because the swapped_job_count will be changed.  It can also be changed through the
 DC 11305 { task switch/monitor swap path.
 DC 11306
 DC 11307       tmp$set_lock (tmv$ptl_lock);
114 11308       jmp$change_ijl_entry_status (ijle_p, jmc$ies_swapin_in_progress);
166 11309       tmp$clear_lock (tmv$ptl_lock);
19E 11310       advance_swap (ijl_ordinal, ijle_p, poll_swapping, request_block.status);
1C0 11311       IF NOT request_block.status.normal THEN
1C8 11312         trace (jsc$ti_swapin_req_status_bad, 1);
1D2 11313       IFEND;
1D2 11314
1D2 11315       tmp$set_lock (jmv$service_class_stats_lock);
206 11316       jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.swap_wait_time :=
224 11317             jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
224 11318             swap_wait_time + (#FREE_RUNNING_CLOCK (0) - ijle_p^.job_scheduler_data.
224 11319             swapin_q_priority_timestamp);
224 11320       jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.swap_stats.
224 11321             scheduler_swapins := jmv$service_classes [ijle_p^.job_scheduler_data.service_class]^.statistics.
224 11322             swap_stats.scheduler_swapins + 1;
224 11323       tmp$clear_lock (jmv$service_class_stats_lock);
27A 11324
27A 11325     = jsc$jss_swap_job_out =
27A 11326
```

```
27A 11327 { The PTL lock must be set to check entry status, to prevent it from changing asynchronously on
27A 11328 { another processor through the dispatcher/monitor swap path.
27A 11329 { If the job's entry status is less than in_memory, the job is non_swappable.  If the entry status is
27A 11330 { greater than swapin in progress, the job is already in a swapped out state.  In either case, do nothing.
27A 11331 { The job can be swapped only if the entry status is in_memory or swapin_in_progress.
27A 11332
27A 11333       tmp$set_lock (tmv$ptl_lock);
2B2 11334
2B2 11335       IF (ijle_p^.entry_status = jmc$ies_job_in_memory) OR
2C6 11336             (ijle_p^.entry_status = jmc$ies_swapin_in_progress) THEN
2C6 11337         job_mode_swapout (ijl_ordinal, ijle_p, request_block.swapout_reason, poll_swapping,
2C6 11338             request_block.status);
6C0 11339       IFEND;
6C0 11340
6C0 11341       tmp$clear_lock (tmv$ptl_lock);
700 11342
700 11343     = jsc$jss_special_swapout =
700 11344
700 11345 { The PTL lock must be set so that the job cannot go into long wait or go ready on another processor while
700 11346 { status is being checked/changed here.
700 11347 { If the job's entry status is less than in_memory, the job is non-swappable and must be left alone.
700 11348 { If the job's entry status is greater than swapin_in_progress, the job is already in a swapped out state;
700 11349 { the entry status must be changed to operator_force_out.  If the entry status is in_memory or swapin_in_
700 11350 { progress, the job must be swapped.
700 11351
700 11352       tmp$set_lock (tmv$ptl_lock);
738 11353
738 11354       IF ijle_p^.entry_status < jmc$ies_job_in_memory THEN
746 11355         mtp$set_status_abnormal ('JM', jme$job_cant_be_swapped, request_block.status);
760 11356
760 11357       ELSEIF ijle_p^.entry_status > jmc$ies_swapin_in_progress THEN
766 11358         IF ijle_p^.entry_status = jmc$ies_job_swapped THEN
76C 11359           IF request_block.swapout_reason = jmc$sr_operator_request THEN
774 11360             jmp$change_ijl_entry_status (ijle_p, jmc$ies_operator_force_out);
7C6 11361           ELSE
7C6 11362             jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_damaged);
814 11363             ijle_p^.job_scheduler_data.swapout_reason := jmc$sr_job_damaged;
81E 11364           IFEND;
822 11365         ELSEIF ijle_p^.entry_status = jmc$ies_system_force_out THEN
828 11366           IF request_block.swapout_reason = jmc$sr_operator_request THEN
830 11367             mtp$set_status_abnormal ('JM', jme$job_dead_cannot_swap, request_block.status);
84A 11368           ELSE
84A 11369             jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_damaged);
89A 11370           IFEND;
89E 11371         ELSEIF ijle_p^.entry_status = jmc$ies_operator_force_out THEN
8A4 11372           IF request_block.swapout_reason = jmc$sr_job_damaged THEN
8AE 11373             jmp$change_ijl_entry_status (ijle_p, jmc$ies_job_damaged);
8FA 11374           IFEND;
8FE 11375
8FE 11376         ELSE
8FE 11377
8FE 11378 { The entry status must be ready_task.  It cannot be job_damaged or swapin_candidate;
8FE 11379 { job mode scheduler checks for those statuses and would not have issued the monitor request.
8FE 11380 { It is too tricky to try to remove the job from the ready task list, so return bad status.  JOB SCHEDULER
8FE 11381 { will advance the job from ready_task to swapin_candidate, and process the operator swapout from there.
```



```
8FE 11382
8FE 11383           mtp$set_status_abnormal ('JM', jme$job_in_ready_task_state, request_block.status);
914 11384         IFEND;
91A 11385
91A 11386       ELSE { entry status = jmc$ies_job_in_memory or jmc$ies_swapin_in_progress }
91A 11387
91A 11388         job_mode_swapout (ijl_ordinal, ijle_p, request_block.swapout_reason, poll_swapping,
91A 11389             request_block.status);
D06 11390         jmp$set_scheduler_event (jmc$examine_swapin_queue);
D1A 11391       IFEND;
D1A 11392
D1A 11393       tmp$clear_lock (tmv$ptl_lock);
D58 11394
D58 11395     = jsc$jss_advance_swap =
D58 11396       ijle_p^.swap_data.swapping_io_error := ioc$no_error;
D58 11397       CASE ijle_p^.swap_status OF
D72 11398       = jmc$iss_job_allocate_swap_file =
D72 11399         trace (jsc$ti_mtr_req_adv_from_aj, 1);
D72 11400         ijle_p^.next_swap_status := jmc$iss_allocate_swap_file;
D72 11401         advance_swap (ijl_ordinal, ijle_p, poll_swapping, request_block.status);
DA8 11402
DA8 11403       = jmc$iss_swapped_io_cannot_init =
DA8 11404         trace (jsc$ti_mtr_req_adv_from_sd, 1);
DA8 11405         jsp$relink_swap_queue (ijl_ordinal, ijle_p, jsc$isqi_swapping);
DCE 11406         advance_swap (ijl_ordinal, ijle_p, poll_swapping, request_block.status);
DEC 11407
DEC 11408       ELSE
DEC 11409       CASEND;
DF2 11410     = jsc$jss_initiate_swapout_io =
DF2 11411       jsp$initiate_swapout_io (request_block.pages_needed);
E0C 11412     = jsc$jss_set_delayed_swapin_work =
E0C 11413       jsp$set_delayed_swapin_work_mtr (request_block.delayed_swapin_work);
E26 11414     ELSE
E26 11415       mtp$error_stop ('JS - unimplemented subfunction code');
E48 11416     CASEND;
E48 11417
E48 11418     IF poll_swapping THEN
E50 11419       set_swapping_event (jsc$se_polling);
E68 11420     IFEND;
E68 11421
E68 11422     #KEYPOINT (osk$exit, 0, jsk$mtr_job_swapping_requests);
E6C 11423
E6C 11424   PROCEND jsp$mtr_job_swapping_requests;
  0 11425
```

NOS/VE js : monitor mode job swapper
[XDCL] jsp$recalculate_swapped_pages

```
        o 11427
        o 11428 { PURPOSE:
        o 11429 {   This procedure recalculates the swapped_job_entry.jws page count and the
        o 11430 {   number of reassignable page frames when job shared pages are removed
        o 11431 {   from the working set of a swapping job.
        o 11432 { NOTE:
        o 11433 {   Only job working set pages cound have been removed.
        o 11434
        o 11435   PROCEDURE [XDCL] jsp$recalculate_swapped_pages
        o 11436     (     ijle_p: ^jmt$initiated_job_list_entry;
        o 11437           pages_removed: mmt$page_frame_index);
        o 11438
        o 11439     VAR
        o 11440       dsw_job_shared_asid_changed: [STATIC] jmt$delayed_swapin_work := [jmc$dsw_job_shared_asid_changed];
        o 11441
        o 11442     trace (jsc$ti_recalculate_sje, pages_removed);
        4 11443
        4 11444     ijle_p^.swap_data.swapped_job_page_count := ijle_p^.swap_data.swapped_job_page_count - pages_removed;
        4 11445
        4 11446     IF (ijle_p^.swap_status >= jmc$iss_swapped_no_io) AND
       36 11447           (ijle_p^.swap_status <= jmc$iss_allocate_swap_file) THEN
       36 11448       ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] :=
       36 11449             ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] - pages_removed;
       36 11450       mmv$reassignable_page_frames.swapout_io_not_initiated :=
       36 11451             mmv$reassignable_page_frames.swapout_io_not_initiated - pages_removed;
       36 11452       trace (jsc$ti_recal_sje_s0, pages_removed);
       5C 11453     ELSEIF (ijle_p^.swap_status >= jmc$iss_wait_job_io_complete) AND
       6C 11454           (ijle_p^.swap_status <= jmc$iss_allocate_sfd) THEN
       6C 11455       ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] :=
       6C 11456             ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] - pages_removed;
       6C 11457       mmv$reassignable_page_frames.soon := mmv$reassignable_page_frames.soon - pages_removed;
       8A 11458     ELSEIF ijle_p^.swap_status = jmc$iss_swapped_io_cannot_init THEN
       94 11459       ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] :=
       94 11460             ijle_p^.swap_data.swapped_job_entry.job_page_queue_count [mmc$pq_job_working_set] - pages_removed;
       94 11461       mmv$reassignable_page_frames.swapout_io_cannot_initiate :=
       B2 11462             mmv$reassignable_page_frames.swapout_io_cannot_initiate - pages_removed;
       B2 11463     ELSEIF ijle_p^.swap_status = jmc$iss_swapped_io_complete THEN
       BA 11464       mmv$reassignable_page_frames.now := mmv$reassignable_page_frames.now - pages_removed;
       BA 11465       ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work + dsw_job_shared_asid_changed;
       BA 11466       trace (jsc$ti_recal_sje_s2, pages_removed);
       EC 11467     IFEND;
       EC 11468
       EC 11469   PROCEND jsp$recalculate_swapped_pages;
        o 11470
```


NOS/VE js : monitor mode job swapper
JSP$SET_DELAYED_SWAPIN_WORK_MTR

```
        o 11472
        o 11473   PROCEDURE [XDCL] jsp$set_delayed_swapin_work_mtr
        o 11474     (     delayed_swapin_work: jmt$delayed_swapin_work_record);
        o 11475
        o 11476     VAR
        o 11477       i: integer,
        o 11478       ijle_p: ^jmt$initiated_job_list_entry,
        o 11479       j: integer;
        o 11480
        o 11481
        o 11482   /set_ijle_work/
        4 11483     FOR i := LOWERBOUND (jmv$ijl_p.block_p^) TO jmv$ijl_p.max_block_in_use DO
       18 11484
       18 11485       IF jmv$ijl_p.block_p^ [i].index_p <> NIL THEN
       2A 11486
       2A 11487       /ijl_inner_loop/
       2A 11488         FOR j := LOWERVALUE (jmt$ijl_block_index) TO UPPERVALUE (jmt$ijl_block_index) DO
       34 11489
       34 11490           ijle_p := ^jmv$ijl_p.block_p^ [i].index_p^ [j];
       34 11491           IF ijle_p^.entry_status <> jmc$ies_entry_free THEN
       4E 11492             ijle_p^.delayed_swapin_work := ijle_p^.delayed_swapin_work +
       4E 11493                   delayed_swapin_work.delayed_swapin_work;
       4E 11494             IF jmc$dsw_update_server_files IN delayed_swapin_work.delayed_swapin_work THEN
       64 11495               ijle_p^.terminate_access_work := ijle_p^.terminate_access_work +
       64 11496                     delayed_swapin_work.terminate_access_work;
       64 11497               ijle_p^.inhibit_access_work := ijle_p^.inhibit_access_work +
       64 11498                     delayed_swapin_work.inhibit_access_work;
       64 11499
       64 11500 { The termination should always have precedence over inhibit.
       64 11501
       64 11502               ijle_p^.inhibit_access_work := ijle_p^.inhibit_access_work - ijle_p^.terminate_access_work;
       82 11503             IFEND;
       82 11504           IFEND;
       82 11505
       82 11506         FOREND /ijl_inner_loop/; { j }
       86 11507
       86 11508       IFEND;
       86 11509
       86 11510     FOREND /set_ijle_work/; { i }
       8A 11511
       8A 11512   PROCEND jsp$set_delayed_swapin_work_mtr;
```

```
        0 11514
        0 11515    PROCEDURE [XDCL] jsp$swap_polling;
        0 11516
        0 11517 {
        0 11518 {       The purpose of this procedure is to advance the swap for jobs that are
        0 11519 {  waiting for events dependent on resource availability (resources such as memory
        0 11520 {  or disk space).  The resources are needed to swap the job not to execute it.
        0 11521 {
        0 11522 {   NOTE: This procedure is entered serially if running with multiple cpu's.
        0 11523 {
        0 11524
        0 11525      VAR
        0 11526        change_swap_direction: boolean,
        0 11527        ijle_p: ^jmt$initiated_job_list_entry,
        0 11528        ijl_ordinal: jmt$ijl_ordinal,
        0 11529        last_swap_status: jmt$ijl_swap_status,
        0 11530        next_ijl_ordinal: jmt$ijl_ordinal,
        0 11531        poll_swapper_again: boolean,
        0 11532        poll_swapping: boolean,
        0 11533        status: syt$monitor_status;
        0 11534
        0 11535      #KEYPOINT (osk$entry, 0, jsk$swap_polling);
        8 11536
        8 11537 {   Set time to call swapper to maximum value so that it won't be called until necessary.
        8 11538 {   This is done now so that if an asynchronous request is received from another cpu it
        8 11539 {   will not be lost.
        8 11540
        8 11541      jsv$time_to_call_job_swapper := UPPERVALUE (ost$free_running_clock);
        8 11542
        8 11543 {   Advance swap on jobs in the swap queue.
        8 11544
        8 11545      ijl_ordinal := jsv$ijl_swap_queue_list [jsc$isqi_swapping].forward_link;
        8 11546      poll_swapper_again := FALSE;
        8 11547
        8 11548    /poll_jobs_being_swapped/
        8 11549      WHILE ijl_ordinal <> jmv$null_ijl_ordinal DO
       34 11550        jmp$get_ijle_p (ijl_ordinal, ijle_p);
       34 11551        next_ijl_ordinal := ijle_p^.swap_queue_link.forward_link;
       34 11552
       34 11553        last_swap_status := ijle_p^.swap_status;
       34 11554        change_swap_direction := ((last_swap_status <= UPPERVALUE (jmt$swapout)) AND
       C4 11555              (last_swap_status >= LOWERVALUE (jmt$swapout)) AND
       C4 11556              (ijle_p^.entry_status < jmc$ies_swapped_out)) OR ((last_swap_status <=
       C4 11557              UPPERVALUE (jmt$swapin)) AND (last_swap_status >= LOWERVALUE (jmt$swapin)) AND
       C4 11558              (ijle_p^.entry_status > jmc$ies_swapped_in));
       C4 11559
       C4 11560        CASE ijle_p^.swap_status OF
      13A 11561        = jmc$iss_executing, jmc$iss_job_idle_tasks_complete, jmc$iss_swapped_no_io, jmc$iss_flush_am_pages,
      13E 11562              jmc$iss_swapped_io_cannot_init, jmc$iss_swapped_io_complete, jmc$iss_swapout_complete =
      13E 11563
      13E 11564 {   Continue advancing the swap.
      13E 11565
      13E 11566        = jmc$iss_wait_allocate_sfd =
      13E 11567          jsv$pages_needed_for_sfd := 0;
      13E 11568          trace (jsc$ti_zero_out_pages_for_sfd_2, 1);
```



```
      13E 11569          advance_swap_state (ijle_p, jmc$iss_allocate_sfd);
      16A 11570        = jmc$iss_wait_allocate_swap_file =
      16A 11571          advance_swap_state (ijle_p, jmc$iss_allocate_swap_file);
      188 11572        = jmc$iss_wait_swapout_io_init =
      188 11573          advance_swap_state (ijle_p, jmc$iss_initiate_swapout_io);
      1A6 11574        = jmc$iss_wait_swapin_io_init =
      1A6 11575          advance_swap_state (ijle_p, jmc$iss_swapin_resource_claimed);
      1C8 11576
      1C8 11577        ELSE
      1C8 11578
      1C8 11579 {   Swap status is either jmc$iss_idle_tasks_initiated, jmc$iss_job_allocate_swap_file,
      1C8 11580 {   jmc$iss_wait_job_io_complete, jmc$iss_swapout_io_initiated, or jmc$iss_swapin_io_initiated.
      1C8 11581 {   All other states are pass thru states and will never come through here.
      1C8 11582
      1C8 11583          IF (ijle_p^.next_swap_status = jmc$iss_null) AND ((NOT change_swap_direction) OR
      1E4 11584                (ijle_p^.swap_status = jmc$iss_swapin_io_initiated)) THEN
      1E4 11585            ijl_ordinal := next_ijl_ordinal;
      1E4 11586            CYCLE /poll_jobs_being_swapped/
      1EA 11587          IFEND;
      1EA 11588        CASEND;
      1EA 11589
      1EA 11590        advance_swap (ijl_ordinal, ijle_p, poll_swapping, status);
      204 11591
      204 11592        IF poll_swapping THEN
      20C 11593          poll_swapper_again := TRUE;
      210 11594        IFEND;
      210 11595
      210 11596        ijl_ordinal := next_ijl_ordinal;
      210 11597      WHILEND /poll_jobs_being_swapped/;
      22A 11598
      22A 11599      IF (poll_swapper_again) AND (jsv$time_to_call_job_swapper = UPPERVALUE (ost$free_running_clock)) THEN
      23E 11600        set_swapping_event (jsc$se_polling);
      252 11601      IFEND;
      252 11602
      252 11603      #KEYPOINT (osk$exit, 0, jsk$swap_polling);
      256 11604
      256 11605    PROCEND jsp$swap_polling;
        0 11606
        0 11607 MODEND jsm$monitor_mode_job_swapper;
```

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

	ERROR	LINE	TEXT
WARNING	CY 821	8455	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10003	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10049	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10070	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10085	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10132	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10149	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10154	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10197	Code scheduling abandoned for this block due to register jamming.
WARNING	CY 821	10367	Code scheduling abandoned for this block due to register jamming.

LEVEL SUMMARY
**** 10 warning diagnostics

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES
active_io_count	2223	10128/M
active_io_page_count	1362	9510
advance_swap	8263	8746 11031 11310 11401 11406 11590
advance_swap_state	8766	8324 8342 8355 8367 8374 8382 8386 8400
		8404 8412 8433 8435 8455 8461 8477 8490
		8509 8515 8524 8536 8545 8572 8578 8612
		8623 8628 8645 8651 8662 8683 8696 8703
		8731 8738 8800 9117 9404 9409 9559 9572
		10446 10537 10604 10660 11140 11257 11260 11338
		11338 11389 11389 11569 11571 11573 11575
age_purge_timestamp	1378	10953/M 10957/P
ajl_ordinal	1356	5361 5398 8324/P 8532 8822 8874 9114 9117/P
		9159 9222 9269 9368 9396/P 9478 9579 9627
		9904 10056 10227 10365 10418 10486/P 10537/P 10568
		10729 10900 10966 10985 10995/S 11134 11164/P 11213
		11338 11338/P 11389 11389/P
ajl_ordinal	8998	9064/P 9073
ajl_ordinal	10589	10593/P 10613
ajl_ordinal	10638	10642/P 10671
ajl_ordinal	11093	11113/P 11125
ajlo	8263	8532/P 8532
ajlo	8861	8874/M 8877/P 8888 8903/P 8912
ajlo	9135	9148/P
ajlo	9196	9222/P 9222
ajlo	9264	9271/P 9274
ajlo	9443	9461/P 9464
ajlo	9567	9579/P 9579
ajlo	10246	10418/P 10418
ajlo	10521	10568/P 10568
allocate_swap_file	8806	8379 8846
allocation_unit_size	792	9481 9482 9628 9628 9629
amc$file_byte_limit	30	33 824
amt$file_byte_address	33	790 4541 5304 5305 5307 5316 7366 7367
		8813
amt$file_limit	824	794
asid	2062	9896/M
asid	2087	9885 9894 9897/M 10001 10139 10146/P
asid	5860	10345 10358/M 10374/M 10378/M 10382/M
asid	9001	9044/M 9046/P 9050/P 9051 9064/P 9085/P
asid	10253	10373/P 10374
asid_reassigned_timestamp	1423	8476/M 10294
assign_active	5904	9699/M
assign_pages_for_sfd	8849	8440 8449 8923 9080
ast_entry	2161	9991 10038 10044/M 10045 10055 10056/P 10061 10078
		10081 10095/M 10096/M 10097/M 10098
ast_index	8999	9046/P 9047/S
aste_p	2228	9158 9159/P 9615 9627/P 9630 9898/M 10015 10114/P
		10145 10146/P 10182 10199 10200 10728 10729/P 10966/P
		10985/P
aste_p	9002	9047/M 9048 9050/P 9054 9056/P 9059/M 9060/M 9078/P
		9085/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
aste_p	10254	10347/M	10348	10348	10349	10349			
asti	789	5670	5674/M	9905/M	10057	10228/M	10368	10368/M	10370
asti	5667	5670/M	5671/S	5671/S	5672/S	5673/M			
asti	5829	10347/S	10360/M	10375/M					
asti	10246	10368/M	10368/S	10368/S	10368/M				
asti	10255	10368/P	10370/M	10372	10373/P	10375			
asynchronous_interrupts_pending	7179	9405/M	10514/M	10841/M	10871/M	11034/M	11118/M	11170/M	11262/M
		11338/M	11389/M	11419/M	11600/M				
available_modified_page_count	1974	9177/M	9178	9547/M	10423/P	10628/P			
available_modified_page_count	9096	9110	9111/P	9119					
available_modified_page_count	9526	9539/P							
b	5391	5400	5400						
b	5435	5445	5446						
b	7267	7274	7275						
b	8806	8822	8822						
b	9126	9159	9159						
b	9314	9387	9387						
b	9432	9478	9478						
b	9594	9627	9627						
b	9735	9754	9754						
b	9843	10227	10227						
b	9861	9904	9904						
b	10246	10365	10365						
b	10797	10803	10803						
b	11179	11241	11241						
b	11273	11307	11307	11315	11315	11333	11333	11338	11338
		11352	11352	11389	11389				
backward_ijle_p	9747	9771/P	9772/M						
backward_link	2123	9770	9771/P	9779/M	9779	9781	9804/M	9806/M	
backward_link	2813	9756	9781/M	9787	9811/M	9817			
bc	7268	7271/M	7272	7276					
bc	9314	9387/M	9387	9387					
bc	9735	9754/M	9754	9754					
bc	10797	10803/M	10803	10803					
bc	11179	11241/M	11241	11241					
bc	11273	11307/M	11307	11307	11315/M	11315	11315	11333/M	11333
		11333	11338/M	11338	11338	11352/M	11352	11352	11389/M
		11389	11389						
bkw	2207	9151	9157	9610	9614	9974	10013	10047/M	10063/M
		10083/M	10096/M	10096	10112/M	10217	10721	10727	10964
		10974	10983	10989					
block_index	1332	5590/S	9684/S	9771/S	9778/S	9802/S	10831/S	10996	11024/S
		11100/S	11203/S	11293/S	11550/S				
block_number	1331	5590/S	9684/S	9771/S	9778/S	9802/S	10831/S	10996	11024/S
		11100/S	11203/S	11293/S	11550/S				
block_p	7670	5590	9684	9771	9778	9802	10831	11024	11100
		11203	11293	11483	11485	11490	11550		
boffset	9599	9628/M	9629	9634					
buffer_descriptor	9444	9473/M	9474/M	9485/M	9486/P				
buffer_descriptor_type	3043	9473/M							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
calculate_sfd_length	8927	8373	8432	8955					
calculate_swapped_pages	8959	8328	8431	8980	9182	11236			
change_asids_in_sfd	9861	9908	10003	10049	10070	10085	10154		
change_swap_direction	8287	8719/M	8724						
change_swap_direction	11526	11554/M	11583						
changing_jf_asid	9867	9903							
changing_jf_asid	9913	10139/M	10154/P	10155					
claim_pages_for_swap_in	8985	8620	9089						
clear	3805	7220/M	9394/M	9762/M	9829/M	10811/M	11248/M	11309/M	11323/M
		11338/M	11341/M	11389/M	11393/M				
cmc$central_memory_element	5100	4976	5206						
cmc$central_processor_element	5101	4965							
cmc$channel_adapter_element	5102	4969	5211						
cmc$choose_any_pp	5143	5114							
cmc$choose_pp_by_barrel	5144	5116							
cmc$choose_pp_by_channel	5143	5118							
cmc$choose_specific_pp	5144	5121							
cmc$communications_element	5105	4969	5211						
cmc$controller_element	5101	4970	5212						
cmc$data_channel_element	5102	4967	5208						
cmc$external_processor_element	5105	4970							
cmc$iou_element	5103	4976							
cmc$mainframe_element	5103	4976							
cmc$max_equipment_per_channel	3948	3951							
cmc$max_esm_size	4125	5162	5167						
cmc$max_low_speed_port_number	5170	5154							
cmc$max_side_door_port_number	5171	4926	4947	4948	5155				
cmc$max_units_per_controller	5092	5095							
cmc$pem_element	5103	4977							
cmc$pp_element	5104	4973							
cmc$storage_device_element	5104	4971							
cmt$central_memory_port_number	5222	5207							
cmt$channel_descriptor	4983	4968							
cmt$channel_identification	5073	5066	5120						
cmt$channel_ordinal	4997	4989	5074						
cmt$element_name	3945	3922	4912	4913	4923	4937	4953	4954	4966
		4984	4987	5056	5065	5075	5129	5150	5203
		5209	5210						
cmt$element_reservation	4963	4931	5236						
cmt$element_state	7051	7010	7011	7041					
cmt$element_type	5100	4964	5205						
cmt$esm_maintenance_buffer_loc	5161	4929	4943	5157					
cmt$esm_memory_size	5167	5156							
cmt$hardware_address	5063	5058							
cmt$model_number	5183	5178							
cmt$peripheral_descriptor	5053	4972							
cmt$physical_address_parts	5087	5084							
cmt$physical_address_specifier	5084	5064							
cmt$physical_equipment_number	3951	3932	5067						
cmt$physical_unit_number	5095	5068							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cmt$pp_identification	5127	5112	5122						
cmt$pp_ordinal	5133	5128							
cmt$pp_reservation	5108	4974							
cmt$pp_reservation_choices	5143	5113							
cmt$product_identification	5175	5151							
cmt$product_number	5181	5176							
cmt$serial_number	5186	5152							
cmt$upline_connection	5189	5154	5155						
complete_swapin	9093	9123	10423	10628					
condition	2899	7203/M	8381	8441	8502	8639	8695	8700	8835/M
		8839/M	8890	9036/M	9086/M	9407	9504	9968/M	10150/M
		11338	11355/M	11367/M	11383/M	11389			
condition	7198	7203							
condition	8806	8835	8839						
condition	8985	9036	9086						
condition	9843	10150							
condition	11273	11355	11367	11383					
count	2280	8328	8328	8424	8426	8431	8431	8969	8971
		9022	9182	9182	9377	10937	10977	10991	11236
		11236	11338	11389					
count	2815	9784/M	9784	9812/M	9812				
count	3028	8781/M	8781						
count	3802	7217	7218/M	7218	7278/M	7278	9387/M	9387	9394
		9394/M	9394	9754/M	9754	9762	9762/M	9762	9829
		9829/M	9829	10803/M	10803	10811	10811/M	10811	11241/M
		11241	11248	11248/M	11248	11307/M	11307	11309	11309/M
		11309	11315/M	11315	11323	11323/M	11323	11333/M	11333
		11338/M	11338	11338	11338/M	11338	11341	11341/M	11341
		11352/M	11352	11389/M	11389	11389	11389/M	11389	11393
		11393/M	11393						
count	9914	10168							
cp_time	1817	10904	10906	10907					
cptime	10884	10904/M	10906/M	10912					
cptime_next_age_working_set	1265	10912							
cst_p	6981	6983/M							
cst_p	9314	9405/P	9405/M						
cst_p	9314	9405/M							
cst_p	10498	10513/M							
cst_p	10507	10513/P	10514/M						
cst_p	10817	10841/P	10841/M						
cst_p	10817	10841/M							
cst_p	10847	10871/P	10871/M						
cst_p	10847	10871/M							
cst_p	11005	11034/P	11034/M						
cst_p	11005	11034/M							
cst_p	11076	11118/P	11118/M	11170/P	11170/M				
cst_p	11076	11118/M	11170/M						
cst_p	11179	11262/P	11262/M						
cst_p	11179	11262/M							
cst_p	11273	11338/P	11338/M	11389/P	11389/M	11419/P	11419/M		
cst_p	11273	11338/M	11389/M	11419/M					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
cst_p	11515	11600/P	11600/M						
cst_p	11515	11600/M							
current_queue	9748	9757/M	9758	9769	9774/S	9781/S	9784/S	9784/S	9787/S
		9788/S	9792/S						
current_queue_id	9915	9973/M	9974/S	10214	10215	10216/M	10216	10217/S	
current_time	8771	8777/M	8778	8797					
cybil_pointer_trick	8231	9006	9267	9450					
cyt$adaptable_array_pointer	7436	7360	7361						
delayed_swapin_work	1386	8324	8324	8324	9048	9117	9117	9117	9398/M
		9980	10278	10296	10297	10396	10402	10411	10467
		10474	10491	10537	10537	10537	10835/M	11216/M	11338/M
		11389/M	11465/M	11465	11492/M	11492			
delayed_swapin_work	1543	11493	11494						
delayed_swapin_work	2841	11413/P							
delayed_swapin_work	11474	11493	11494	11496	11498				
delta_time	8772	8778/M	8784	8786	8787	8793			
dfc$active	4380	4349							
dfc$awaiting_recovery	4381	4356							
dfc$command_record_bytes	1456	1464	4294						
dfc$deactivated	4380	4365							
dfc$division_overwrite_words	1443	1471							
dfc$esm_command_record_size	1464	1472							
dfc$esm_connection	4324	3930	4321						
dfc$esm_header_record_size	1465	1472							
dfc$esm_maintenance_buf_size	1444	1475							
dfc$esm_memory_base_shift	1450	1472	1473	1473					
dfc$header_record_bytes	1455	1465							
dfc$inactive	4380	4356							
dfc$max_data_record_bytes	1459	4959							
dfc$max_esm_divisions	1453	3959							
dfc$max_esm_memory_size	1445	1474	4924	4938					
dfc$max_number_of_mainframes	1452	1437	3958						
dfc$max_number_of_queues	4390	4006	4008	4010	4395				
dfc$max_queue_entries	4391	4138	4139	4180	4181	4396			
dfc$max_req_timeout_count_value	4275	4153	4195						
dfc$max_request_buffer_entries	3955	4869							
dfc$max_retransmit_count_value	4279	4154	4196						
dfc$maximum_lifetime	4455	4452							
dfc$maximum_queue_interfaces	5240	5243							
dfc$maximum_user_buffer_area	4402	4412	4413						
dfc$maximum_user_data_area	4406	4415	4416						
dfc$min_data_record_bytes	1460	1471	4959						
dfc$min_esm_division_size	1470	1474							
dfc$min_esm_memory_size	1446	4924	4938						
dfc$mock_connection	4325	4321							
dfc$monitor	4385	4204							
dfc$monitor_allocate	4165	4155							
dfc$monitor_io	4165	4155							
dfc$queue_assignment_strng_size	4180	4141							
dfc$recovering	4381	4370							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dfc$task_services	4385	4214							
dfc$terminated	4381	4356							
dfc$unrecovered_disk_error	4592	4620							
dfp$fetch_page_status	5269	10731							
dfp$set_task_segment_state	5276	10773							
dft$allocated_command_buffer	4289	4288							
dft$allocated_data_rma_list	4250	4249							
dft$allocated_monitor_buffer	4313	4312							
dft$channel_definition	4952	4926	4948						
dft$channel_specification	4911	3923	3924						
dft$connection_address	4055	4050	4051						
dft$connection_descriptor	4049	4036							
dft$connection_flags	4063	4056							
dft$connection_type	4324	3919	4142						
dft$cpu_queue	4129	4023							
dft$cpu_queue_entries	4134	4131							
dft$cpu_queue_entry	4185	4134							
dft$cpu_queue_header	4137	4130							
dft$cpu_queue_pva_entries	4010	3996							
dft$cpu_queue_pva_entry	4022	4011							
dft$data_descriptor	4110	4077	4078	4079					
dft$dma_adapter	3999	3986							
dft$driver_queue	4027	4019							
dft$driver_queue_entries	4068	4029							
dft$driver_queue_entry	4070	4068							
dft$driver_queue_header	4032	4028							
dft$driver_queue_header_flags	4039	4033							
dft$driver_queue_pva_entries	4008	3995							
dft$driver_queue_pva_entry	4018	4009							
dft$driver_queue_rma_entries	4006	3994							
dft$driver_queue_rma_entry	4013	4007							
dft$esm_base_addresses	3975	3968	4927	4941					
dft$esm_definition_table_entry	4922	4919	4932						
dft$esm_pp_information	3930	3925	3926						
dft$inquiry_message	4815	4806	4877						
dft$inquiry_tracer	4820	4816							
dft$interrupt	4044	4034							
dft$lifetime	4452	4448							
dft$mainframe_set	1437	1387	1388	1546	1547	5280	5281	9403	10748
		10749	10754	10759	10766	10767	10779	10840	11221
		11338	11389						
dft$maximum_data_bytes	4959	4928	4942						
dft$monitor_io_types	4165	4209							
dft$p_allocated_data_rma_list	4249	4161							
dft$p_command_buffer	4287	4198	4199						
dft$p_data_rma_list	4236	4201							
dft$p_queue_interface_table	3963	3916							
dft$p_send_data	4416	4216	4217						
dft$partner_status	4339	4146							
dft$pp_element_reservations	5236	3934							
dft$pp_status	3937	3931							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
dft$q_interface_directory_entry	3914	3912							
dft$queue_directory	3985	3970							
dft$queue_directory_index	5243	3900							
dft$queue_entry_flags	4082	4071	4193						
dft$queue_entry_index	4396	3902							
dft$queue_entry_location	3899	3889							
dft$queue_entry_type	4385	4203							
dft$queue_index	4395	3901							
dft$queue_interface_directory	3911	3909							
dft$queue_interface_table	3965	3963							
dft$request_buffer	4864	4860							
dft$request_buffer_directory	4853	3967							
dft$request_buffer_entries	4869	4866							
dft$request_buffer_entry	4874	4870							
dft$request_buffer_entry_flags	4882	4875							
dft$response_flags	4793	4785							
dft$response_parameter	4803	4787							
dft$retransmission_digit	4826	4822							
dft$rpc_progress_record	4427	4219							
dft$send_data_size	4415	4218	4224	4226	4227	4430	4431	4436	
dft$send_parameter_size	4412	4435							
dft$server_iocb_error_condition	4557	4543							
dft$server_lifetime	4448	4147							
dft$server_state	4380	4348	4383						
dft$side_door_ports	4946	4940							
dft$transaction_data	4168	4160							
dft$transaction_digit	4825	4821							
dft$transaction_state	4494	4194	4501	4817					
dfv$file_server_debug_enabled	5299	10747							
direction	8852	8876	8891	8908					
direction_changed_to_in	10521	8727	10581						
dispatch_control	7020	9405/M	10514/M	10841/M	10871/M	11034/M	11118/M	11170/M	11262/M
		11338/M	11389/M	11419/M	11600/M				
dispatching_priority	6160	9722/M							
dmc$device_manager_error_code	109	110	113	116	119	122	125	128	131
		134	137	140	143	146	149	152	155
		158	161	164	167	170	173	176	179
		182	185	188	191	194	197	200	203
		206	209	212	215	218	221	224	227
		230	233	236	239	242	245	248	251
		254	257	260	263	266	269	272	275
		278	281	284	287	290	293	296	299
		302	305	308	311	314	317	320	323
		326	329	332	335	338	341	344	347
		350	353	356	359	362	365	368	371
		374	377	380	383	386	389	392	401
		404	407	410	413	416	419	422	425
		428	431	434	437	440	443	446	449
		452	455	458	461	464	467	470	473
		476	479	482	485	488	491	494	497
		500	503	506	509	512	515	518	521

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		524	527	530	533	536	539	542	545
		548	551	554	557	560	563	566	569
		572	575	578	581	584	587	590	593
		596	599	602	605	608	611	614	617
		620	623	626	629	632	635	638	641
		644	647	650	653	656	659	662	665
		668	671	677	680	683	686	689	692
		695	698	701	704	707	710	713	716
		720	723	726	731	734	737	740	743
		746	749	752	755	758			
dmc$fas_file_allocated	764	8828							
dmc$fas_job_mode_work_required	765	8833							
dmc$fas_temp_reject	766	8837							
dmc$max_class_ordinal	7501	7498							
dmc$max_login_table_entries	7535	7530	7532						
dme$transient_error	296	8381	8839/P						
dmp$allocate_file_space	5302	8823							
dmp$recover_job_dm_tables	5322	9728							
dmp$set_fau_state	5314	9647							
dmt$active_volume_table	7317	7309							
dmt$active_volume_table_entry	7319	7317							
dmt$avt_lock	7334	7320							
dmt$class	7492	7348							
dmt$class_member	7493	7492							
dmt$disk_table_status	7381	7385							
dmt$file_allocation_status	763	5309	8812						
dmt$global_file_name	4624	4536							
dmt$internal_vsn	7516	7353							
dmt$login_table_entry_index	7530	7525							
dmt$login_table_sequence	7528	7524							
dmt$mainframe_assigned	7523	7359							
dmt$ms_active_vol_table_entry	7343	7324							
dmt$ms_avt_status	7375	7378							
dmt$ms_volume_system_status	7378	7364							
dmt$ms_volume_table_status	7385	7347							
dmt$system_class	7494	7349							
dmt$system_file_id	1486	1419	4208	4538	7354	7355	7356	7357	7358
		9435							
dpc$console_row_size	5340	5334							
dpc$top_line_message_size	5334	5329	8009						
dpp$display_error	5328	9960	10752	10757	10762				
dpt$top_line_message	8009	7984							
dsw_job_shared_asid_changed	11440	11465							
entry_status	1355	5517	5522/M	8309	8321	8324	8324/M	8336	8347
		8361	8484	8553	8564	8589	8670	8721	8723
		8725	9117	9117/M	9326	9329	9329/M	9331	9331/M
		9336	9336/M	9583	10447	10447/M	10537	10537/M	10805
		11116	11116/M	11142	11142/M	11168	11168/M	11207	11207/M
		11308	11308/M	11335	11336	11338	11338	11338/M	11338
		11338/M	11338	11338/M	11354	11357	11358	11360	11360/M

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11362	11362/M	11365	11369	11369/M	11371	11373	11373/M
		11389	11389	11389/M	11389	11389/M	11389	11389/M	11491
		11556	11558						
entry_updated	2162	9886	9893	9899/M	10021				
eoffset	9600	9629/M	9634						
eoi_byte_address	790	8830/M							
eoi_modified	813	8831/M							
event_time	9314	9405	9405						
event_time	10499	10510	10512						
event_time	10817	10841	10841						
event_time	10847	10871	10871						
event_time	11005	11034	11034						
event_time	11076	11118	11118	11170	11170				
event_time	11179	11262	11262						
event_time	11273	11338	11338	11389	11389	11419	11419		
event_time	11515	11600	11600						
existing_entry	9870	9886/M	9887	9893					
existing_pfte_p	9918	10173/M	10182	10183	10187	10188	10189	10199	10200
existing_pfti	9917	10172/M	10173/S	10185/P	10186/P				
f1	8246	9072	10612	10670	11124				
f2	8247	9073	10613	10671	11125				
fde_p	5345	5367/M	5368						
fde_p	5391	5398/M	5398						
fde_p	5393	5398/P	5400/P						
fde_p	5664	5670	5674/M						
fde_p	8806	8822/P	8822/P						
fde_p	8806	8822/M	8822						
fde_p	8811	8822/P	8823/P	8830/M	8831/M				
fde_p	9126	9159/P	9159/P						
fde_p	9126	9159/M	9159						
fde_p	9136	9159/P	9160/P						
fde_p	9432	9478/P	9478/P						
fde_p	9432	9478/M	9478						
fde_p	9445	9478/P	9481	9482	9486/P				
fde_p	9594	9627/P	9627/P						
fde_p	9594	9627/M	9627						
fde_p	9601	9627/P	9628	9628	9629	9647/P			
fde_p	9843	10056/M	10056	10227/M	10227				
fde_p	9843	10227/P	10227/P						
fde_p	9861	9904/P	9904/P						
fde_p	9861	9904/M	9904						
fde_p	9871	9904/P	9905/M						
fde_p	9919	10056/P	10057	10227/P	10228/M				
fde_p	10246	10365/P	10365/P						
fde_p	10246	10365/M	10365						
fde_p	10246	10368	10368/M						
fde_p	10256	10365/P	10367/P	10370					
fde_p	10710	10729/M	10729						
fde_p	10715	10729/P	10730	10731/P					
fde_p	10880	10966/M	10966	10985/M	10985				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
fde_p	10885	10966/P	10967	10970	10985/P	10986			
file_entry_index	1219	5357	5398	8822	9159	9478	9627	9904	10056
		10227	10365	10729	10966	10985			
file_hash	787	5368	5398	8822	9159	9478	9627	9904	10056
		10227	10365	10729	10966	10985			
file_hash	1221	5356	5368	5398	5398	8822	8822	9159	9159
		9478	9478	9627	9627	9904	9904	10056	10056
		10227	10227	10365	10365	10729	10729	10966	10966
		10985	10985						
file_status	8812	8824/P	8827						
flags	780	8831/M							
flush_am_pages_to_disk	9126	8372	9191						
forward_ijl_p	9749	9778/P	9779/M						
forward_link	2124	9772/M	9772	9774	9777	9778/P	9803/M	9810/M	11551
forward_link	2814	9774/M	9788	9792	9807/M	9818	9822	11545	
found_sva	9916	10168	10169						
free_swap_file_descriptor	9254	8532	9222	9302	9579	10418	10568		
free_swapped_jobs_mm_resources	9196	8577	8675	8702	9251	9558	10149		
fwd	2208	9540	9544	9630	9639	10048/M	10064/M	10084/M	10097/M
		10097	10113/M						
gfc$fde_size	5384	5357	5398	8822	9159	9478	9627	9904	10056
		10227	10365	10729	10966	10985			
gfc$fde_table_base	5382	5357	5383	5398	8822	9159	9478	9627	9904
		10056	10227	10365	10729	10966	10985		
gfc$fk_catalog	859	871							
gfc$fk_job_local_file	861	870							
gfc$fm_mass_storage_file	874	803							
gfc$fm_served_file	875	806	10730						
gfc$monitor_interlocks	5412	5399	8822	9159	9478	9627	9904	10227	10365
gfc$ps_server_terminated	1211	10732							
gfc$ps_volume_unavailable	1209	10732							
gfc$tr_job	1229	5360	5398	5672	8822	9159	9478	9627	9904
		10056	10182	10227	10349	10364	10365	10366	10368
		10729	10966	10985					
gfc$tr_system	1229	5359	5398	8822	9159	9478	9627	9708	9715
		9904	10038	10056	10227	10363	10365	10729	10966
		10985							
gfc$tr_system_wait_recovery	1229	9709	9717	9903	10044	10728			
gfp$mtr_get_fde_p	5343	5372	5398	8822	9159	9478	9627	9904	10056
		10227	10365	10729	10966	10985			
gfp$mtr_get_locked_fde_p	5391	5403	8822	9159	9478	9627	9904	10227	10365
gft$allocation_unit_size	830	792							
gft$attach_count	835	783	784						
gft$fde_flags	812	780							
gft$file_desc_entry_p	769	5345	9445	9601	9678	9919	10715	10885	
gft$file_descriptor_entry	777	769	782	1196					
gft$file_descriptor_index	844	1219							
gft$file_kind	855	786	867						
gft$file_media	874	802							
gft$locked_file_desc_entry_p	1196	5270	5303	5315	5393	5469	5664	6972	8811

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9136	9871	10256					
gft$open_count	904	785	920						
gft$page_status	1204	5272	9449	10718					
gft$queue_status	915	795							
gft$segment_lock_info	919	788							
gft$signature_lock	880	778							
gft$system_file_identifier	1218	1486	2193	5343	5391	5665	5895	6097	6188
		9676	9924						
gft$table_residence	1229	1220	5353						
gft$transfer_unit_size	841	793							
global_asids_changed	10258	10295/M	10305	10313					
global_task_id	6149	9690/P	10406/P						
hash	5350	5356/M	5363/M						
hash	5391	5398/M	5398/M						
hash	8806	8822/M	8822/M						
hash	9126	9159/M	9159/M						
hash	9432	9478/M	9478/M						
hash	9594	9627/M	9627/M						
hash	9843	10056/M	10056/M	10227/M	10227/M				
hash	9861	9904/M	9904/M						
hash	10246	10365/M	10365/M						
hash	10710	10729/M	10729/M						
hash	10880	10966/M	10966/M	10985/M	10985/M				
head	930	10126/M							
hung_task_in_job	1392	9961/M							
i	9314	9359/M	9359	9359/S	9359/S	9359/S			
i	11063	11065/M	11066	11067/S	11068/S	11069/S			
i	11076	11104/M	11104	11104/S	11104/S	11104/S	11166/M	11166	11166/S
		11166/S	11166/S						
i	11179	11201/M	11201	11201/S	11201/S	11201/S			
i	11273	11299/M	11299	11299/S	11299/S	11299/S	11338/M	11338	11338/S
		11338/S	11338/S	11389/M	11389	11389/S	11389/S	11389/S	
i	11477	11483	11485/S	11490/S					
i#build_adaptable_array_ptr	5457	8903							
i#program_error	5388	5369	5398	7215	8822	9159	9394	9478	9627
		9762	9829	9904	10056	10227	10365	10729	10811
		10966	10985	11248	11309	11323	11338	11341	11389
		11393							
i#real_memory_address	5465	8906	10332						
id	3803	7214	7272	7276/M	9387	9387/M	9394	9754	9754/M
		9762	9829	10803	10803/M	10811	11241	11241/M	11248
		11307	11307/M	11309	11315	11315/M	11323	11333	11333/M
		11338	11338/M	11338	11341	11352	11352/M	11389	11389/M
		11389	11393						
ignore_aus_obtained	8813	8824/P							
ignore_overflow	8814	8824/P							
ijl_entry	2150	8916/M	9466/M	9954					
ijl_inner_loop	11487	11487	11506						
ijl_ordinal	2186	5672	9048	9060/M	9158	9815	10077	10349	10368

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	2834	11292							
ijl_ordinal	3047	9474/M							
ijl_ordinal	3882	9454/M							
ijl_ordinal	5586	5590/S	5590/S						
ijl_ordinal	7639	10995							
ijl_ordinal	8263	8324/P	8324/P	8324/P					
ijl_ordinal	8263	8532/P							
ijl_ordinal	8264	8324/P	8338/P	8341/P	8348/P	8362/P	8366/P	8372/P	8440/P
		8449/P	8485/P	8498/P	8505/P	8527/P	8532/P	8546/P	8554/P
		8568/P	8577/P	8579/P	8596/P	8620/P	8624/P	8625/P	8636/P
		8641/P	8666/P	8675/P	8684/P	8692/P	8697/P	8698/P	8701/P
		8702/P	8704/P	8727/P					
ijl_ordinal	8851	8877/P							
ijl_ordinal	8986	9048	9060	9064/P	9078/P	9080/P			
ijl_ordinal	9093	9117/P	9117/P	9117/P					
ijl_ordinal	9094	9117/P							
ijl_ordinal	9127	9148/P	9158						
ijl_ordinal	9196	9222/P							
ijl_ordinal	9198	9222/P							
ijl_ordinal	9256	9271/P							
ijl_ordinal	9315	9334/P	9348/P	9350/P	9352/P	9359/P	9370/P		
ijl_ordinal	9433	9454	9461/P	9474					
ijl_ordinal	9553	9558/P	9560/P	9561/P					
ijl_ordinal	9567	9579/P							
ijl_ordinal	9568	9578/P	9579/P	9584/P					
ijl_ordinal	9595	9615							
ijl_ordinal	9658	9684/S	9684/S						
ijl_ordinal	9660	9686/P							
ijl_ordinal	9735	9771/S	9771/S	9778/S	9778/S	9802/S	9802/S		
ijl_ordinal	9736	9803	9807	9811					
ijl_ordinal	9844	10077	10149/P	10231/P					
ijl_ordinal	10246	10418/P							
ijl_ordinal	10247	10285/P	10324/P	10349	10367/P	10397/P	10404/P	10412/P	10418/P
		10423/P							
ijl_ordinal	10430	10445/P	10468/P	10475/P					
ijl_ordinal	10521	10537/P	10537/P	10537/P					
ijl_ordinal	10521	10568/P							
ijl_ordinal	10522	10537/P	10556/P	10568/P	10576/P				
ijl_ordinal	10585	10592/P	10605/P	10606/P	10628/P				
ijl_ordinal	10634	10641/P	10649/P	10653/P	10659/P	10664/P	10676/P		
ijl_ordinal	10691	10698/P							
ijl_ordinal	10712	10773/P							
ijl_ordinal	10817	10831/S	10831/S						
ijl_ordinal	10818	10831/P							
ijl_ordinal	10886	10995/M	10996	10996					
ijl_ordinal	11005	11024/S	11024/S						
ijl_ordinal	11006	11024/P	11026/P	11031/P					
ijl_ordinal	11076	11100/S	11100/S						
ijl_ordinal	11077	11100/P	11104/P	11112/P	11117/P	11139/P	11166/P	11169/P	
ijl_ordinal	11179	11203/S	11203/S						
ijl_ordinal	11180	11201/P	11203/P	11258/P	11261/P				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ijl_ordinal	11273	11293/S	11293/S						
ijl_ordinal	11273	11337/P	11338/P	11338/P	11338/P	11338/P	11338/P	11389/P	11389/P
		11389/P	11389/P	11389/P	11389/P				
ijl_ordinal	11285	11292/M	11293/P	11299/P	11302/P	11310/P	11337/P	11388/P	11401/P
		11405/P	11406/P						
ijl_ordinal	11515	11550/S	11550/S						
ijl_ordinal	11528	11545/M	11549	11549	11550/P	11585/M	11590/P	11596/M	
ijle_p	5344	5361							
ijle_p	5391	5398							
ijle_p	5392	5398/P							
ijle_p	5511	5517	5522/M	5526/P	5530/P				
ijle_p	5587	5590/M							
ijle_p	8263	8324/P	8324/P	8324/P	8324	8324	8324/M	8324	8324
		8324	8324/M	8324	8324/P	8324	8324/P	8324/P	8324/P
		8324	8324/P						
ijle_p	8263	8324	8324/M	8324/P	8324/P				
ijle_p	8263	8328	8328/M	8328	8328/M	8328	8328	8431	8431/M
		8431	8431/M	8431	8431				
ijle_p	8263	8532	8532/P	8532	8532/M	8532/P	8532/P	8532/P	8532/M
		8532	8532/M						
ijle_p	8265	8296	8302	8305	8309	8321	8323/M	8324/P	8327/P
		8328/P	8331	8336	8338/P	8341/P	8342/P	8347	8348/P
		8355/P	8361	8362/P	8366/P	8367/P	8372/P	8373/P	8374/P
		8378	8379/P	8382/P	8385/M	8386/P	8394	8394	8396
		8396	8398/M	8399	8400/P	8403/M	8404/P	8412/P	8423
		8424	8425	8426	8430	8430	8431/P	8432/P	8433/P
		8435/P	8440/P	8448/P	8449/P	8453	8455/P	8461/P	8472/P
		8476/M	8477/P	8483	8483	8484	8485/P	8489	8489
		8490/P	8496	8497	8498/P	8498/P	8499/P	8504/M	8505/P
		8509/P	8515/P	8520	8521/M	8522	8524/P	8527/P	8532/P
		8536/P	8542	8542	8544	8544	8545/P	8546/P	8553
		8554/P	8564	8565/M	8567	8567	8568/P	8572/P	8577/P
		8578/P	8579/P	8589	8595	8596/P	8608	8611/M	8612/P
		8619/M	8620/P	8623/P	8624/P	8628/P	8634	8635	8636/P
		8636/P	8637/P	8641/P	8645/P	8651/P	8659	8660	8662/P
		8663/M	8666/P	8670	8675/P	8683/P	8684/P	8685/P	8692/P
		8692/P	8693/P	8696/P	8697/P	8699/P	8701/P	8702/P	8703/P
		8705/P	8718	8721	8723	8725	8727/P	8730	8731/P
		8731/P	8732/M	8737	8738/P	8738/P	8739/M		
ijle_p	8767	8775	8776/M	8778	8797/M	8798/M			
ijle_p	8806	8822/P							
ijle_p	8806	8822							
ijle_p	8807	8819	8820	8821	8822/P	8822/P	8829/M		
ijle_p	8850	8871	8872	8874	8877/P	8892/P	8902	8907/M	8913
		8915/M	8915	8916/M	8916	8918/P			
ijle_p	8928	8940	8953/M						
ijle_p	8960	8969	8970/M	8971	8974/M	8977	8978		
ijle_p	8987	9009	9010	9044	9048	9051/M	9071	9078/P	9079/P
		9080/P	9083/P	9084/P					
ijle_p	9093	9117/P	9117/P	9117/P	9117	9117	9117/M	9117	9117
		9117	9117/M	9117	9117/P	9117	9117/P	9117/P	9117/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9117	9117/P						
ijle_p	9093	9117	9117/M	9117/P	9117/P				
ijle_p	9095	9111/P	9114	9117/P					
ijle_p	9126	9159/P							
ijle_p	9126	9159							
ijle_p	9126	9182	9182/M	9182	9182/M	9182	9182		
ijle_p	9128	9148/P	9159/P	9174/P	9177/M	9178	9180	9181	9182/P
		9185							
ijle_p	9196	9222	9222/P	9222	9222/M	9222/P	9222/P	9222/P	9222/M
		9222	9222/M						
ijle_p	9197	9221	9222/P	9226	9230/P	9237/P	9244/P		
ijle_p	9255	9269	9271/P	9273	9275/M	9278/P	9278/P	9282/P	9288/M
		9294	9300/M						
ijle_p	9314	9329	9329/M	9329/P	9329/P	9331	9331/M	9331/P	9331/P
		9336	9336/M	9336/P	9336/P				
ijle_p	9316	9326	9329/P	9331/P	9333	9334/P	9336/P	9347	9348/P
		9349	9350/P	9351	9352/P	9361/M	9362/M	9363	9365
		9368	9370/P	9377	9380/M	9381/M	9385/M	9388/S	9389/S
		9390	9390	9391/S	9392/S	9393	9396/P	9396/P	9398/M
		9403/M	9404/P	9409/P					
ijle_p	9432	9478/P							
ijle_p	9432	9478							
ijle_p	9434	9461/P	9463	9465/M	9466/M	9466	9468	9478/P	9496/P
		9508/M	9509/M	9510	9517/M				
ijle_p	9525	9540	9544	9547/M					
ijle_p	9554	9558/P	9559/P	9560/P	9562/P				
ijle_p	9567	9579	9579/P	9579	9579/M	9579/P	9579/P	9579/P	9579/M
		9579	9579/M						
ijle_p	9569	9572/P	9574	9574	9576	9577	9578/P	9579/P	9583
		9584/P							
ijle_p	9594	9627/P							
ijle_p	9594	9627							
ijle_p	9596	9625	9627/P						
ijle_p	9658	9684/M							
ijle_p	9659	9686/P	9728/P						
ijle_p	9735	9771/M	9778/M	9802/M					
ijle_p	9737	9757	9770	9771/P	9772	9774	9777	9778/P	9779
		9781	9804/M	9806/M	9810/M	9827/M			
ijle_p	9843	10056	10227						
ijle_p	9843	10227/P							
ijle_p	9845	9947/P	9954	9959	9961/M	9962	9963/M	9965/M	9974
		9980	9988	10001	10003/P	10049/P	10056/P	10070/P	10085/P
		10149/P	10154/P	10214	10214	10217	10227/P	10231/P	
ijle_p	9861	9904/P							
ijle_p	9861	9904							
ijle_p	9866	9904/P							
ijle_p	10246	10365/P							
ijle_p	10246	10365							
ijle_p	10246	10418	10418/P	10418	10418/M	10418/P	10418/P	10418/P	10418/M
		10418	10418/M						
ijle_p	10248	10278	10285/P	10294	10296	10297	10324/P	10358	10359/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		10365/P	10396	10397/P	10402	10403/M	10404/P	10406/P	10411
		10412/P	10417	10418/P	10423/P	10423/P			
ijle_p	10429	10447	10447/M	10447/P	10447/P				
ijle_p	10431	10445/P	10446/P	10447/P	10451	10453	10454/M	10455	10456
		10458	10462/M	10467	10468/P	10474	10475/P	10479/P	10486/P
		10491	10492/P						
ijle_p	10521	10537/P	10537/P	10537/P	10537	10537	10537/M	10537	10537
		10537	10537/M	10537	10537/P	10537	10537/P	10537/P	10537/P
		10537	10537/P						
ijle_p	10521	10537	10537/M	10537/P	10537/P				
ijle_p	10521	10568	10568/P	10568	10568/M	10568/P	10568/P	10568/P	10568/M
		10568	10568/M						
ijle_p	10523	10533	10537/P	10538/M	10555/M	10556/P	10567/M	10568/P	10573
		10573	10574/M	10575/M	10576/P				
ijle_p	10586	10592/P	10596	10597	10599	10599	10602	10602	10604/P
		10605/P	10611	10616	10618	10619	10620	10621	10623
		10623	10628/P	10628/P					
ijle_p	10635	10641/P	10645	10648	10648	10649/P	10650	10652	10652
		10653/P	10654	10655	10658	10658	10659/P	10660/P	10669
		10676/P							
ijle_p	10690	10698/P							
ijle_p	10710	10729							
ijle_p	10711	10721	10729/P	10748	10749	10751	10754	10756	10759
		10761	10766	10767	10773/P	10773/P	10774/P	10779/M	
ijle_p	10817	10831/M							
ijle_p	10828	10831/P	10833	10834/M	10835/M	10840/M			
ijle_p	10848	10858/M	10860	10862/M	10864/M	10866/M			
ijle_p	10880	10966	10985						
ijle_p	10881	10900	10901	10904	10906	10907	10914/P	10925/P	10926/P
		10931	10937	10938/P	10941/M	10943	10953/M	10957/P	10964
		10966/P	10977	10983	10985/P	10991	10995/S		
ijle_p	11005	11024/M							
ijle_p	11019	11024/P	11026/P	11031/P					
ijle_p	11076	11100/M							
ijle_p	11076	11116	11116/M	11116/P	11116/P	11142	11142/M	11142/P	11142/P
		11168	11168/M	11168/P	11168/P				
ijle_p	11094	11100/P	11103	11112/P	11116/P	11117/P	11123	11129	11130
		11134	11139/P	11140/P	11142/P	11146	11148	11149/M	11150
		11151	11153	11157/M	11161/P	11164/P	11168/P	11169/P	
ijle_p	11179	11203/M							
ijle_p	11179	11207	11207/M	11207/P	11207/P				
ijle_p	11179	11236	11236/M	11236	11236/M	11236	11236		
ijle_p	11196	11203/P	11204	11207/P	11213	11215/M	11216/M	11221/M	11225/M
		11231/P	11236/P	11239	11242/S	11243/S	11244	11244	11245/S
		11246/S	11247	11257/P	11258/P	11260/P	11261/P		
ijle_p	11273	11293/M							
ijle_p	11273	11308	11308/M	11308/P	11308/P	11338	11338/M	11338/P	11338/P
		11338	11338/M	11338/P	11338/P	11338	11338/M	11338/P	11338/P
		11360	11360/M	11360/P	11360/P	11362	11362/M	11362/P	11362/P
		11369	11369/M	11369/P	11369/P	11373	11373/M	11373/P	11373/P
		11389	11389/M	11389/P	11389/P	11389	11389/M	11389/P	11389/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		11389	11389/M	11389/P	11389/P				
ijle_p	11273	11338	11338/P	11338/P	11338	11338/P	11338/P	11338	11338/P
		11338	11338/P	11338	11338/P	11338/M	11338/M	11338	11338
		11338	11338/P	11338	11338/M	11338/M	11338/M	11338/S	11338/S
		11338	11338	11338/S	11338/S	11338	11338/P	11338/P	11338/M
		11338/M	11338/P	11338/P	11389	11389/P	11389/P	11389	11389/P
		11389/P	11389	11389/P	11389	11389/P	11389	11389/P	11389/M
		11389/M	11389	11389	11389	11389/P	11389	11389/M	11389/M
		11389/M	11389/S	11389/S	11389	11389	11389/S	11389/S	11389
		11389/P	11389/P	11389/M	11389/M	11389/P	11389/P		
ijle_p	11284	11293/P	11302/P	11308/P	11310/P	11316/S	11317/S	11318	11320/S
		11321/S	11335	11336	11337/P	11354	11357	11358	11360/P
		11362/P	11363/M	11365	11369/P	11371	11373/P	11388/P	11396/M
		11397	11400/M	11401/P	11405/P	11406/P			
ijle_p	11436	11444/M	11444	11446	11447	11448/M	11449	11453	11454
		11455/M	11456	11458	11459/M	11460	11463	11465/M	11465
ijle_p	11478	11490/M	11491	11492/M	11492	11495/M	11495	11497/M	11497
		11502/M	11502	11502					
ijle_p	11515	11550/M							
ijle_p	11527	11550/P	11551	11553	11556	11558	11560	11569/P	11571/P
		11573/P	11575/P	11583	11584	11590/P			
ijlo	5666	5672							
ijlo	9314	9359							
ijlo	10246	10368							
ijlo	11053	9359/M	11068/M	11104/M	11166/M	11201/M	11299/M	11338/M	11389/M
ijlo	11060	11068							
ijlo	11076	11104	11166						
ijlo	11179	11201							
ijlo	11273	11299	11338	11389					
in_use	2187	5671	9054	10055	10078	10092	10348	10368	
index	891	10406/P							
index_p	7683	5590	9684	9771	9778	9802	10831	11024	11100
		11203	11293	11485	11490	11550			
inhibit_access_work	1387	10749	10754	10767	10773/P	11497/M	11497	11502/M	11502
inhibit_access_work	1546	11498							
inhibit_swap_count	1361	8399							
initial_rtc	10887	10901/M	10943						
initiate_swap_io	9476	9476	9490	9493					
initiate_swapout_io	8288	8333/M	8340						
initiate_swapout_io	11197	11251/M	11256						
io_control_information	9438	9480	9486/P	9492					
io_error	2229	9645	9645						
io_function	9436	9456	9487/P	9495					
io_id	9137	9153/M	9160/P						
io_id	9446	9453/M	9454/M	9487/P					
ioc$allocate	4512	3888							
ioc$allocate_file_space	2254	8378	8385						
ioc$disk_min_ecc	2370	2371	2379						
ioc$error_on_init	2256	9645							
ioc$max_unit_number	4834	4837							
ioc$no_error	2254	8520	8659	11396					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ioc$read_ahead_on_server	4512	3890							
ioc$read_for_server	4511	3886							
ioc$read_from_client	4511	3887							
ioc$read_page	4506	3883							
ioc$st_errors	2379	2380	2381	2382	2383	2384	2385	2386	2387
		2388	2389	2390	2391	2392	2393	2394	2395
		2396	2397	2398	2399	2400	2401	2402	2403
		2404							
ioc$swap_in	4507	3881	8636/P						
ioc$swap_out	4507	3881	8498/P	9456	9495				
ioc$tape_min_ecc	2372	2373							
ioc$unit_down_on_init	2256	9645							
ioc$unrecovered_error_unit_down	2255	8504	8522	8660	9509				
ioc$write_for_server	4512	3886							
ioc$write_page	4506	3883	9160/P						
ioc$write_to_client	4512	3887							
ioe$unit_disabled	2402	8502	8639	9504					
iop$pager_io	5468	9486							
iot$interrupt	4901	4046							
iot$io_error	2254	1420	2229						
iot$io_function	4506	3880	4207	5473	6974	9436			
iot$logical_unit	4837	4788	7322						
iot$port_number	4906	4903							
iot$pp_number	4896	3987	3988						
iot$transfer_count	6326	6314							
j	8254	8256							
j	8263	8315	8322	8324	8337	8349	8428	8447	8492
		8503	8523	8526	8555	8597	8640	8661	8665
		8674	8726						
j	8806	8825	8834	8838					
j	8985	9026	9035	9049	9053	9055	9066	9082	
j	9093	9117							
j	9126	9161	9168	9186					
j	9196	9223	9232	9239	9246				
j	9314	9366							
j	9432	9489							
j	9524	9538	9539						
j	9594	9616	9626	9636	9646				
j	9843	10025	10039	10042	10051	10059	10066	10079	10093
		10123	10142	10148	10153	10159	10184	10190	10192
		10201							
j	9861	9888	9890	9895					
j	10246	10284	10300	10303	10306	10328	10338	10355	10361
		10376	10379	10383					
j	10429	10460							
j	10521	10536	10537	10550	10553	10586			
j	10584	10595							
j	10633	10644							
j	10797	10806							
j	10880	10910	10913	10928	10929	10944			

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
j	11076	11105	11115	11155	11167				
j	11179	11205							
j	11273	11300	11312	11338	11389	11399	11404		
j	11435	11442	11452	11466					
j	11479	11488	11490/S						
j	11515	11568							
jcb_p	8862	8912/M	8913/M						
jcb_p	9105	9114/M	9115/M	9115					
jcb_p	9447	9467/M	9468/M						
jcb_p	10888	10900/M	10912	10912	10914/P	10922	10925/P	10938/P	
jcb_p	11095	11134/M	11135/M	11135					
jf_asid	9920	9988/M	9989/P	10003/P	10139	10157/M			
jf_asid_changed	9921	9981/M	10004/M	10156/M	10226				
jf_aste_p	9922	9990/M	9992/M	10003/P					
jf_asti	9923	9989/P	9990/S	10003/P	10158/M	10228			
jf_asti	10259	10359/P	10360						
jf_sfid	9924	9991/M	9992	10227/P					
jmc$detached_job_wait_time_max	1290	1287							
jmc$dsw_adjust_cpu_selections	1538	10396							
jmc$dsw_io_error_while_swapped	1538	8324	9117	10474	10537				
jmc$dsw_job_asid_changed	1536	10297							
jmc$dsw_job_recovery	1535	9048	9980	10278					
jmc$dsw_job_shared_asid_changed	1536	8324	9117	10296	10491	10537	11440		
jmc$dsw_update_debug_lists	1535	10402							
jmc$dsw_update_job_task_enviro	1537	8324	9117	10467	10537				
jmc$dsw_update_server_files	1537	10411	11494						
jmc$examine_swapin_queue	2605	11390/P							
jmc$highest_det_job_wait_time	1300	1290	1301						
jmc$highest_prio_age_interval	6256	6247	6257						
jmc$highest_service_accumulator	1934	1935							
jmc$highest_service_factor_valu	6280	6273							
jmc$highest_working_set_size	2327	2318	2328	2330	2332	2334			
jmc$ies_entry_free	1709	11491							
jmc$ies_job_damaged	1717	9331/P	11338/P	11362/P	11369/P	11373/P	11389/P		
jmc$ies_job_in_memory	1712	8324/P	9117/P	10447/P	10537/P	11142/P	11335	11354	
jmc$ies_job_swapped	1714	1723	9336/P	11207/P	11338/P	11358	11389/P		
jmc$ies_operator_force_out	1715	9329/P	11338/P	11360/P	11371	11389/P			
jmc$ies_swapin_in_progress	1713	1722	5524	5525	5528	5529	8324	8324	8324
		8324	9117	9117	9117	9117	9329	9329	9329
		9329	9331	9331	9331	9331	9336	9336	9336
		9336	9340	10447	10447	10447	10447	10537	10537
		10537	10537	10805	11116/P	11116	11116	11116	11116
		11142	11142	11142	11142	11168/P	11168	11168	11168
		11168	11207	11207	11207	11207	11308/P	11308	11308
		11308	11308	11336	11338	11338	11338	11338	11338
		11338	11338	11338	11338	11338	11338	11338	11338
		11357	11360	11360	11360	11360	11362	11362	11362
		11362	11369	11369	11369	11369	11373	11373	11373
		11373	11389	11389	11389	11389	11389	11389	11389
		11389	11389	11389	11389	11389	11389		
jmc$ies_swapped_in	1722	8309	8670	8723	11558				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$ies_swapped_out	1723	8321	8336	8347	8361	8484	8553	8564	8589
		8721	8725	9583	11556				
jmc$ies_system_force_out	1716	11365							
jmc$iss_allocate_sfd	1739	8435/P	8438	8490/P	11454	11569/P			
jmc$iss_allocate_swap_file	1735	8374/P	8376	8433/P	10616	11400	11447	11571/P	
jmc$iss_executing	1728	8307	8324/P	9117/P	9365	10446/P	10537/P	11140/P	11338
		11389	11561						
jmc$iss_flush_am_pages	1732	8355/P	8367/P	8370	11260/P	11561			
jmc$iss_free_swapped_memory	1746	8536/P	8572/P	8575					
jmc$iss_idle_tasks_initiated	1729	1756	9409/P	10535	10833	11338/P	11389/P		
jmc$iss_initiate_swapout_io	1741	8477/P	8494	8524/P	11573/P				
jmc$iss_job_allocate_swap_file	1733	8386/P	8412/P	10547	11398				
jmc$iss_job_idle_tasks_complete	1730	8319	9404/P	10834	11338/P	11389/P	11561		
jmc$iss_job_io_complete	1737	8404/P	8417	10862					
jmc$iss_null	1727	8323	8732	8737	8739	10538	10555	10575	11583
jmc$iss_swapin_io_complete	1754	1757	8656	8675/P	8702/P	8730	9558/P	10149/P	10866
jmc$iss_swapin_io_initiated	1753	8651/P	10865	11584					
jmc$iss_swapin_requested	1750	1757	8612/P	8617					
jmc$iss_swapin_resource_claimed	1751	8628/P	8632	8662/P	9226	11575/P			
jmc$iss_swapout_complete	1749	1756	8578/P	8587	8623/P	8683/P	8696/P	8703/P	9351
		9559/P	11338	11389	11562				
jmc$iss_swapout_io_complete	1744	8518	8535	10864					
jmc$iss_swapout_io_initiated	1743	8515/P	10565	10654	10863				
jmc$iss_swapped_io_cannot_init	1740	1767	8479	9347	9572/P	10645	11338	11389	11403
		11458	11562						
jmc$iss_swapped_io_complete	1745	8545/P	8562	8571	9233	9349	9625	10650	11338
		11389	11463	11562					
jmc$iss_swapped_no_io	1731	1766	8342/P	8359	10604/P	10660/P	11103	11257/P	11446
		11561							
jmc$iss_wait_allocate_sfd	1738	8455/P	8461/P	10549	10551	10597	10621	11566	
jmc$iss_wait_allocate_swap_file	1734	8382/P	10548	11570					
jmc$iss_wait_job_io_complete	1736	8400/P	10548	10596	10620	10861	11453		
jmc$iss_wait_swapin_io_init	1752	8645/P	11574						
jmc$iss_wait_swapout_io_init	1742	8509/P	10565	10655	11572				
jmc$keyword_offset_maximum	1317	2319	6248						
jmc$kjl_maximum_entries	1507	1500	1501	1886					
jmc$kol_maximum_entries	1517	1502							
jmc$lock_ajl	5491	8532/P	8532/P	8877/P	8892/P	8918/P	9148/P	9174/P	9222/P
		9222/P	9271/P	9282/P	9461/P	9496/P	9579/P	9579/P	10418/P
		10418/P	10568/P	10568/P					
jmc$max_active_jobs	1498	6229	6237	6238					
jmc$max_ajl_ord	1499	1492	1498	2410					
jmc$max_dispatching_control	1673	1677							
jmc$max_dispatching_priority	1595	1555	1558	1559					
jmc$max_ijl_entries	1339	2815							
jmc$max_ijl_index_count	1340	7681							
jmc$maximum_job_classes	1864	1867							
jmc$maximum_job_count	1514	1507							
jmc$maximum_output_count	1524	1517							
jmc$maximum_service_classes	1952	1955							
jmc$min_dispatching_control	1672	1676							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmc$min_ecc	2430	2431	2438						
jmc$min_ecc_sch	2438	2439	2441	2443	2445	2447	2449	2451	2453
		2455	2457	2459	2461	2463	2465	2467	2469
		2471	2473	2478	2482	2485	2488	2491	2494
		2497	2500	2503	2506	2510	2513	2517	2520
		2523	2526	2529	2532	2535	2539	2542	2546
		2549	2552	2556	2559	2563	2567	2571	2575
		2578	2581	2584					
jmc$needed_memory_available	2607	5560/S	5562/S						
jmc$null_ajl_ordinal	2410	8532	9222	9269	9579	10418	10568		
jmc$null_service_class	1945	1946							
jmc$priority_aging_interval_max	6247	6244							
jmc$priority_p1	1609	1556	7185						
jmc$priority_p10	1618	1557	2417	2421	2424				
jmc$priority_p11	1619	2418	2420						
jmc$priority_p12	1620	2419							
jmc$priority_p13	1621	2422							
jmc$priority_p14	1622	1557	7185						
jmc$priority_p8	1616	1556							
jmc$priority_p9	1617	2425							
jmc$priority_system_job	2417	9722							
jmc$required_offset	1315	2333							
jmc$reserved_ajls	1503	1498							
jmc$restart_on_abort	1915	9962							
jmc$restart_on_recovery	1919	9963							
jmc$service_accumulator_maximum	1926	1923							
jmc$service_factor_value_max	6273	6270							
jmc$sr_job_damaged	1966	9330	11338	11363	11372	11389			
jmc$sr_operator_request	1959	9328	11338	11359	11366	11389			
jmc$swapping_ajl	5490	8327/P	8685/P	8699/P	8705/P	9084/P	9084/P	9562/P	10592/P
		10641/P	11112/P	11231/P					
jmc$system_default_offset	1316	1317	2335						
jmc$system_supplied_name_size	2108	2105							
jmc$terminate_on_recovery	1919	9965							
jmc$unlimited_offset	1313	1291	1302	1936	2329	6258			
jmc$unspecified_offset	1314	2331							
jmc$working_set_size_maximum	2318	2315							
jme$job_cant_be_swapped	2443	11355/P							
jme$job_dead_cannot_swap	2461	11367/P							
jme$job_in_ready_task_state	2578	11383/P							
jmp$activate_job_mode_swapper	5479	8387	8413	9587					
jmp$assign_ajl_entry	5480	8532	8877	9064	9148	9222	9271	9461	9579
		10418	10568	10592	10641				
jmp$assign_ajl_with_lock	5494	11112							
jmp$change_ijl_entry_status	5510	5533	8324	9117	9329	9331	9336	10447	10537
		11116	11142	11168	11207	11308	11338	11338	11338
		11360	11362	11369	11373	11389	11389	11389	
jmp$check_scheduler_memory_wait	5535	5566							
jmp$decrement_swapped_job_count	5569	5530	8324	9117	9329	9331	9336	10447	10537
		11116	11142	11168	11207	11308	11338	11338	11338
		11360	11362	11369	11373	11389	11389	11389	

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmp$free_ajl_entry	5573	8327	8532	8685	8699	8705	8892	8918	9084
		9174	9222	9282	9496	9562	9579	10418	10568
jmp$free_ajl_with_lock	5580	11231							
jmp$get_ijle_p	5586	5592	9684	9771	9778	9802	10831	11024	11100
		11203	11293	11550					
jmp$increment_swapped_job_count	5597	5526	8324	9117	9329	9331	9336	10447	10537
		11116	11142	11168	11207	11308	11338	11338	11338
		11360	11362	11369	11373	11389	11389	11389	
jmp$recognize_job_dead	5601	8704	9561						
jmp$reset_job_to_swapped_out	5606	8625	8698	10606	10664	10807			
jmp$set_entry_status_to_rt	5611	9334	11338	11389					
jmp$set_scheduler_event	5618	11390							
jmp$set_scheduler_memory_event	5556	5563							
jmt$active_job_list	7645	7623							
jmt$active_job_list_entry	7637	7645							
jmt$ajl_ordinal	1492	1356	4206	5290	5484	5499	7014	7245	7259
		7757	8861	8998	9135	9264	9443	10589	10638
		11093							
jmt$delayed_swapin_work	1539	1386	1543	9398	10835	11216	11338	11389	11440
		11440							
jmt$delayed_swapin_work_record	1542	2841	11474						
jmt$detached_job_wait_time	1287	1272							
jmt$dispatching_control	1643	6212							
jmt$dispatching_control_index	1676	1633	1643						
jmt$dispatching_controls	1646	1644							
jmt$dispatching_priority	1555	1368	1634	1635	1636	1648	6160	6162	7006
jmt$ijl_block_index	1336	1332	7683	11488	11488				
jmt$ijl_block_number	1335	1331	7671	7672					
jmt$ijl_dispatching_control	1632	1369							
jmt$ijl_entry_status	1709	1355	5512	5515	7657	7657	9323		
jmt$ijl_entry_status_statistics	7657	7650							
jmt$ijl_ordinal	1330	1262	1375	1403	2123	2124	2186	2221	2813
		2814	2834	3047	3882	5279	5291	5481	5496
		5586	5601	5606	5612	5636	5666	7033	7237
		7299	7639	7697	7762	8264	8851	8986	9094
		9127	9198	9256	9315	9433	9553	9568	9595
		9660	9736	9750	9844	10247	10430	10522	10585
		10634	10691	10712	10798	10818	10886	11006	11053
		11060	11077	11180	11285	11528	11530		
jmt$ijl_p	7669	7664							
jmt$ijl_page_fault_count	1783	1778	1779	1780					
jmt$ijl_page_stats	1777	1773							
jmt$ijl_service_class_stats	1771	1390							
jmt$ijl_statistics	1816	1389							
jmt$ijl_swap_count	1792	1788	1789						
jmt$ijl_swap_counts	1787	1409	1774						
jmt$ijl_swap_status	1727	1358	1359	1360	3023	3023	8290	8768	8773
		9199	10531	11529					
jmt$initiated_job_list_block	7680	7686							
jmt$initiated_job_list_entry	1352	1261	2150	5278	5292	5323	5344	5392	5511
		5569	5574	5581	5587	5597	5613	5713	5718

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		5726	6964	7034	7236	7254	7293	7298	7640
		7683	8265	8767	8807	8850	8928	8960	8987
		9095	9128	9197	9255	9316	9434	9525	9554
		9569	9596	9659	9679	9737	9747	9749	9751
		9845	9866	10248	10431	10523	10586	10635	10690
		10711	10799	10828	10848	10881	11019	11094	11196
		11284	11436	11478	11527				
jmt$initiated_job_list_p	7686	7670							
jmt$input_file_location	1906	1901							
jmt$jl_job_leveler_state	4521	4516							
jmt$jl_job_leveler_status	4515	4145							
jmt$job_abort_disposition	1915	1899							
jmt$job_class	1867	1414							
jmt$job_control_block	1243	5719	5725	6965	7016	8862	9105	9447	10888
		11095							
jmt$job_mode	1870	1371							
jmt$job_priority	1875	1411	1412	6221	6222	6223	6224		
jmt$job_recovery_disposition	1918	1900							
jmt$job_sched_event_selections	2612	5540							
jmt$job_scheduler_event	2610	5545							
jmt$job_scheduler_events	2590	2610	2612	5618					
jmt$job_system_id	2297	1258							
jmt$kjl_index	1886	1357	2297						
jmt$maximum_active_jobs	6229	6206							
jmt$mtr_serv_class_stat_entry	7713	7709							
jmt$priority_aging_interval	6244	6214							
jmt$queue_file_ijl_information	1898	1396							
jmt$sc_cp_stat	7741	7720	7721						
jmt$sc_pf_stat	7742	7725	7726	7727					
jmt$sc_swap_count	7744	7731	7732	7736	7738				
jmt$sc_swap_stat	7743	7733	7734	7735	7737				
jmt$scheduling_data	1402	1380							
jmt$scheduling_priority	6220	6213							
jmt$service_accumulator	1923	1404	1405	1406	6204	6205			
jmt$service_class_attributes	6192	7708							
jmt$service_class_cp_time	7719	7714							
jmt$service_class_entry	7707	7703							
jmt$service_class_index	1955	1415	6197	6207	7703				
jmt$service_class_name	6262	6199	6200						
jmt$service_class_page_faults	7724	7715							
jmt$service_class_swap_stats	7730	7716							
jmt$service_factor_value	6270	6208							
jmt$service_factors	6266	6208							
jmt$swap_data	1418	1382							
jmt$swapin	1757	8722	8723	11557	11557				
jmt$swapout	1756	8719	8720	11554	11555				
jmt$swapout_reasons	1958	1410	2837	7246	9317				
jmt$swapped_job_entry	1973	1281	1427	2151	5634	9846			
jmt$system_supplied_name	2105	1256	1353	7768					
jmt$task_time_slice	1686	1666	1667						
jmt$time_slice_values	1665	1650	6173						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jmt$user_supplied_name	2301	1257							
jmt$working_set_size	2315	1268	1269	10890					
jmv$ajl_p	7623	10995							
jmv$ijl_entry_status_statistics	7650	5519/M	5520	8324/M	8324	9117/M	9117	9329/M	9329
		9331/M	9331	9336/M	9336	10447/M	10447	10537/M	10537
		11116/M	11116	11142/M	11142	11168/M	11168	11207/M	11207
		11308/M	11308	11338/M	11338	11338/M	11338	11338/M	11338
		11360/M	11360	11362/M	11362	11369/M	11369	11373/M	11373
		11389/M	11389	11389/M	11389	11389/M	11389		
jmv$ijl_p	7664	5590	9684	9771	9778	9802	10831	11024	11100
		11203	11293	11483	11483	11485	11490	11550	
jmv$job_sched_events_selected	5540	5560							
jmv$job_scheduler_event	5545	5562							
jmv$long_wait_swap_threshold	7694	8334	11252						
jmv$memory_needed_by_scheduler	5551	5561							
jmv$null_ijl_ordinal	7697	9724/P	9770	9777	9787	9788	9792	9801	9806
		9810	9817	9818	9822	10391/P	10407/P	10704/P	11549
		11549							
jmv$service_class_stats_lock	7750	9387/P	9394/P	11241/P	11248/P	11315/P	11323/P	11338/P	11338/P
		11389/P	11389/P						
jmv$service_classes	7702	9388/M	9389	9391/M	9392	11242/M	11243	11245/M	11246
		11316/M	11317	11320/M	11321	11338/M	11338	11338/M	11338
		11389/M	11389	11389/M	11389				
jmv$system_ijl_ordinal	7762	9684/P							
job_abort_disposition	1899	9962							
job_asids_changed	10260	10297/M	10302	10313					
job_fixed_asid	1365	8532/P	8877/P	9044	9051/M	9148/P	9222/P	9271/P	9461/P
		9579/P	9988	10001	10358	10359/P	10418/P	10568/P	10592/P
		10641/P	11112/P						
job_fixed_contiguous_pages	1391	8328	8394	8396	8430	8431	8483	8489	8542
		8544	8567	8978	9181	9182	9574	9577	10573
		10599	10602	10619	10623	10648	10652	10658	11130
		11236							
job_mode	1789	9362/M	9363	11338/M	11338	11389/M	11389		
job_mode_swapout	9314	9424	11337	11388					
job_page_count	8263	8328/M	8328/M	8328	8328	8431/M	8431/M	8431	8431
job_page_count	8289	8606/M	8608/M	8608	8611				
job_page_count	8935	8940/M	8946	8948	8948				
job_page_count	8963	8966/M	8969/M	8969	8974				
job_page_count	9126	9182/M	9182/M	9182	9182				
job_page_count	9322	9375/M	9377/M	9377	9380				
job_page_count	11179	11236/M	11236/M	11236	11236				
job_page_count	11273	11338/M	11338/M	11338	11338	11389/M	11389/M	11389	11389
job_page_queue_count	1975	8328/M	8423	8425	8431/M	8609	8970/M	9182/M	11236/M
		11448/M	11449	11455/M	11456	11459/M	11460		
job_page_queue_list	1381	8328	8328	8424	8426	8431	8431	8969	8971
		9079/P	9083/P	9182	9182	9230/P	9237/P	9244/P	9377
		9540	9544	9974	10217	10721	10925/P	10937	10964
		10977	10983	10991	11236	11236	11338	11389	
job_queue_id	8263	8328	8328/S	8328/S	8328/S	8431	8431/S	8431/S	8431/S
job_queue_id	8964	8968	8969/S	8970/S	8971/S				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
job_queue_id	9126	9182	9182/S	9182/S	9182/S				
job_queue_id	11179	11236	11236/S	11236/S	11236/S				
job_recovery_disposition	1900	9963/M	9965/M						
job_scheduler_data	1380	9361/M	9362/M	9363	9388/S	9389/S	9391/S	9392/S	9396/P
		11242/S	11243/S	11245/S	11246/S	11316/S	11317/S	11318	11320/S
		11321/S	11338/M	11338/M	11338	11338/S	11338/S	11338/S	11338/S
		11338/P	11363/M	11389/M	11389/M	11389	11389/S	11389/S	11389/S
		11389/S	11389/P						
job_swap_counts	1409	9362/M	9363	11338/M	11338	11389/M	11389		
job_swapping_io	9432	8498	8636	9520					
jsc$isqi_null	2127	8324/P	9117/P	9769	9800	10445/P	10537/P	11139/P	11204
jsc$isqi_swapped_io_cannot_init	2128	9348/P	9578/P	10649/P	11338/P	11389/P			
jsc$isqi_swapped_io_completed	2128	2130	8546/P	9350/P	10653/P	11338/P	11389/P		
jsc$isqi_swapped_io_not_init	2127	2130	8341/P	10605/P	10659/P	11258/P			
jsc$isqi_swapped_out	2128	8579/P	8624/P	8684/P	8697/P	8701/P	9352/P	9560/P	11338/P
		11389/P							
jsc$isqi_swapping	2127	8296	8348/P	8366/P	8554/P	8595	8596/P	9370/P	9584/P
		9759	10801/P	11026/P	11117/P	11169/P	11261/P	11302/P	11338/P
		11389/P	11405/P	11545/S					
jsc$jss_advance_swap	2828	11395							
jsc$jss_initiate_swapout_io	2828	2838	11410						
jsc$jss_set_delayed_swapin_work	2828	2840	11412						
jsc$jss_special_swapout	2829	2836	11343						
jsc$jss_swap_job_in	2827	11298							
jsc$jss_swap_job_out	2827	2836	11325						
jsc$min_ecc	2618	2619							
jsc$min_ecc_js	2619	2622	2625	2628	2631	2634	2637	2640	2643
		2646	2649	2652	2655				
jsc$sc_swapin_job_mode	11045	11299/P							
jsc$sc_swapin_mtr_direct	11046	11104/P							
jsc$sc_swapin_mtr_mode	11046	11166/P							
jsc$sc_swapout_job_mode	11045	9359/P	11338/P	11389/P					
jsc$sc_swapout_mtr_mode	11045	11201/P							
jsc$sd_in	1430	9080/P							
jsc$sd_out	1430	8440/P	8449/P	8876	8891	8908			
jsc$se_immediate	3015	3019	9405/P	9405	10512	10841/P	10841	10871/P	10871
		11034	11118/P	11118	11170/P	11170	11262/P	11262	11338/P
		11338	11389/P	11389	11419	11600			
jsc$se_polling	3016	3019	11034/P	11419/P	11600/P				
jsc$ti_advance_from_cannot_init	8171	8492/P							
jsc$ti_allocate_swap_file	8093	8825/P							
jsc$ti_allocate_swap_file_jm	8094	8834/P							
jsc$ti_cd_idle_task_complete	8154	8337/P							
jsc$ti_cd_idle_task_complete_2	8156	8349/P							
jsc$ti_cd_to_in_at_s	8159	8597/P							
jsc$ti_cd_to_in_at_s2	8158	8555/P							
jsc$ti_change_asid	8097	9890/P							
jsc$ti_change_asid_again	8096	9888/P							
jsc$ti_change_asid_sfd	8098	9895/P							
jsc$ti_dm_transient_error	8095	8838/P							
jsc$ti_dump_shared_q_for_sfd	8167	8447/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jsc$ti_dump_shared_queue	8168	9026/P							
jsc$ti_flush_am_pc	8089	9161/P							
jsc$ti_flush_am_ready	8091	9186/P							
jsc$ti_flush_am_relink	8090	9168/P							
jsc$ti_free_memory	8085	9239/P	9246/P						
jsc$ti_free_memory_si_aborted	8084	9232/P							
jsc$ti_free_readied_s2_job	8169	10806/P							
jsc$ti_init_swapin_io_error	8160	8640/P							
jsc$ti_init_swapout_io_error	8161	8503/P							
jsc$ti_lwa	8134	10910/P							
jsc$ti_lwa_cp_age	8135	10913/P							
jsc$ti_lwa_ready_task	8138	10944/P							
jsc$ti_lwa_stale_mod_pages_rem	8137	10929/P							
jsc$ti_lwa_stale_pages_rem	8136	10928/P							
jsc$ti_max_index	8189	8197	8253						
jsc$ti_min_index	8077	8197	8253						
jsc$ti_move_am_back_to_am	8087	9538/P							
jsc$ti_move_am_back_to_am_pc	8088	9539/P							
jsc$ti_mtr_req_adv_from_aj	8173	11399/P							
jsc$ti_mtr_req_adv_from_sd	8174	11404/P							
jsc$ti_new_job_fixed_asid	8079	9049/P							
jsc$ti_no_ajl_ord_for_swap_in	8147	9066/P							
jsc$ti_no_ajlo_mtr_swapin	8184	11115/P							
jsc$ti_no_ajlo_swapin_after_io	8170	10644/P							
jsc$ti_no_ajlo_swapin_before_io	8166	10595/P							
jsc$ti_no_memory_for_swap_in	8078	9035/P							
jsc$ti_no_pages_for_sfd_on_si	8082	9082/P							
jsc$ti_page_q_counts_different	8172	8428/P							
jsc$ti_pager_io_error	8086	9489/P							
jsc$ti_pt_full_reassign_jf	8127	10159/P							
jsc$ti_recal_sje_s0	8176	11452/P							
jsc$ti_recal_sje_s2	8177	11466/P							
jsc$ti_recalculate_sje	8175	11442/P							
jsc$ti_reserve_memory_failed	8157	8324/P	9117/P	10460/P	10537/P	11155/P			
jsc$ti_reuse_job_fixed_asid	8080	9053/P							
jsc$ti_reuse_job_fixed_asid_as	8081	9055/P							
jsc$ti_riop_init	8182	9646/P							
jsc$ti_riop_m_bit_reset	8181	9636/P							
jsc$ti_riop_mem_freed	8180	9626/P							
jsc$ti_riop_relinked	8179	9616/P							
jsc$ti_rmmt_lf_assign_asid	8108	10079/P							
jsc$ti_rmmt_lf_reuse_asid	8109	10093/P							
jsc$ti_rmmt_no_change	8102	10025/P							
jsc$ti_rmmt_pf	8103	10039/P							
jsc$ti_rmmt_pf_assign_asid	8106	10059/P							
jsc$ti_rmmt_pf_rec_ptm	8104	10042/P							
jsc$ti_rmmt_pf_rec_ptu	8105	10051/P							
jsc$ti_rmmt_pf_reuse_asid	8107	10066/P							
jsc$ti_rmmt_pt_done	8110	10123/P							
jsc$ti_rmmt_pt_full	8111	10142/P							
jsc$ti_rmmt_pt_full_failed	8112	10148/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES						
jsc$ti_rmmt_pt_full_succ	8113	10153/P						
jsc$ti_rmmt_pte_exists_a	8116	10184/P						
jsc$ti_rmmt_pte_exists_am	8115	10192/P						
jsc$ti_rmmt_pte_exists_err	8117	10190/P						
jsc$ti_rmmt_pte_exists_pf	8114	10201/P						
jsc$ti_rxcb_fix_asids	8125	10338/P						
jsc$ti_rxcb_fix_jf_asid	8128	10361/P						
jsc$ti_rxcb_fix_job_asid	8129	10376/P						
jsc$ti_rxcb_fix_templ_asid	8126	10355/P						
jsc$ti_rxcb_fix_xcb_sdt	8124	10328/P						
jsc$ti_rxcb_glob_asids_changed	8123	10306/P						
jsc$ti_rxcb_job_asids_changed	8122	10303/P						
jsc$ti_rxcb_recovery	8131	10284/P						
jsc$ti_rxcb_temp_asids_changed	8121	10300/P						
jsc$ti_rxcb_zero_asid	8132	10383/P						
jsc$ti_rxcb_zero_job_asid	8130	10379/P						
jsc$ti_sfd_freed	8083	9223/P						
jsc$ti_sif_idle_tasks_init	8142	10536/P						
jsc$ti_sif_idled_tasks_comp	8155	8322/P						
jsc$ti_sif_swapout_io_init	8144	10566/P						
jsc$ti_sif_wait_state	8143	10550/P						
jsc$ti_swapin_disk_down	8163	8661/P						
jsc$ti_swapin_from_job_mode	8150	11300/P						
jsc$ti_swapin_from_mtr_mode	8151	11167/P						
jsc$ti_swapin_int_by_swapout	8146	8674/P						
jsc$ti_swapin_io_error	8140	8665/P						
jsc$ti_swapin_mtr_direct	8152	11105/P						
jsc$ti_swapin_req_status_bad	8153	11312/P						
jsc$ti_swapout_disk_down	8162	8523/P						
jsc$ti_swapout_from_job_mode	8148	9366/P	11338/P	11389/P				
jsc$ti_swapout_from_mtr_mode	8149	11205/P						
jsc$ti_swapout_int_by_swapin	8145	8726/P						
jsc$ti_swapout_io_error	8141	8526/P						
jsc$ti_swapping_queue_and_exec	8092	8315/P						
jsc$ti_zero_out_pages_for_sfd_1	8164	10553/P						
jsc$ti_zero_out_pages_for_sfd_2	8165	11568/P						
jse$bad_swap_file_data_detected	2655	8700	9968					
jse$not_enough_mem_for_swap_in	2622	9036/P	9086/P					
jse$pt_full_on_swap_in	2628	8695	10150/P					
jse$swap_file_not_allocated	2649	8835/P						
jse$unable_to_idle_all_tasks	2631	9407	11338	11389				
jsk$base	2738	2665	2669	2673	2677	2681	2685	2689	2693
		2697						
jsk$flush_am_pages_to_disk	2669	9144	9189					
jsk$free_swapped_jobs_mm_resour	2685	9217	9249					
jsk$long_wait_aging	2677	10898	11000					
jsk$monitor_swap_in	2665	11098	11173					
jsk$mtr_job_swapping_requests	2673	11288	11422					
jsk$swap_polling	2681	11535	11603					
jsp$free_swap_resident_job	10797	10813						
jsp$idle_tasks_complete	10817	10844						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jsp$initiate_swapout_io	5623	11411							
jsp$io_complete	10847	10873							
jsp$long_wait_aging	10880	11002							
jsp$monitor_advance_swap	11005	10809	11037						
jsp$monitor_swap_in	11076	11175							
jsp$monitor_swap_out	11179	11269							
jsp$mtr_job_swapping_requests	11273	11424							
jsp$recalculate_swapped_pages	11435	11469							
jsp$relink_swap_queue	9735	8324	8341	8348	8366	8546	8554	8579	8596
		8624	8684	8697	8701	9117	9348	9350	9352
		9370	9560	9578	9584	9831	10445	10537	10605
		10649	10653	10659	10801	11026	11117	11139	11169
		11258	11261	11302	11338	11338	11338	11338	11389
		11389	11389	11389	11405				
jsp$set_delayed_swapin_work_mtr	11473	11413	11512						
jsp$swap_polling	11515	11605							
jst$changed_asid_entry	2173	2164							
jst$ijl_swap_queue_id	2127	2122	2820	9738	9748				
jst$ijl_swap_queue_link	2121	1364							
jst$ijl_swap_queue_list	2820	8205	8205						
jst$ijl_swap_queue_list_entry	2812	2820							
jst$io_control_information	2135	1383	9438						
jst$job_swapping_subfunctions	2827	2835							
jst$rb_job_swapping_functions	2831	11274							
jst$swap_direction	1430	8852							
jst$swap_file_descriptor	2149	1384	8234	8936	9847				
jst$swap_file_page_count	92	8204							
jst$swap_file_statistics	3032	8218	8218						
jst$swap_state_statistics	3023	8217							
jst$swap_state_statistics_entry	3025	3024							
jst$swapped_page_descriptor	2158	2156	8904/P	8905/P	8946	8948	8948	9939	
jst$swapped_page_descriptors	2155	2152							
jst$swapping_event	3019	10499							
jst$swapping_request_type	11045	11052	11059						
jsv$enable_swap_file_statistics	8208	10963							
jsv$enable_swap_resident	8209	8534							
jsv$enable_swap_resident_no_io	8210	8334	11252						
jsv$free_working_set_on_swapout	8213	10917							
jsv$halt_on_swapin_failure	8212	9955							
jsv$ijl_serial_lock	8202	9754/P	9762/P	9829/P					
jsv$ijl_swap_queue_list	8205	9756	9774/M	9781/M	9784/M	9784	9787	9788	9792
		9807/M	9811/M	9812/M	9812	9817	9818	9822	11545
jsv$max_pages_first_swap_task	8214	10932							
jsv$maximum_pages_to_swap	8215	10934							
jsv$pages_needed_for_sfd	8216	8452/M	8452	10552/M	11567/M				
jsv$sched_swapping_requests	11049	9359	9359/M	9359/M	9359/M	9359/M	11065	11066/M	11067/M
		11068/M	11069/M	11104	11104/M	11104/M	11104/M	11104/M	11166
		11166/M	11166/M	11166/M	11166/M	11201	11201/M	11201/M	11201/M
		11201/M	11299	11299/M	11299/M	11299/M	11299/M	11338	11338/M
		11338/M	11338/M	11338/M	11389	11389/M	11389/M	11389/M	11389/M
jsv$swap_file_page_count	8204	8329/M	8329	8330/M	8330	11237/M	11237	11238/M	11238

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
jsv$swap_file_statistics	8218	10972/M	10973	10976/M	10978/M	10978			
jsv$swap_state_statistics	8217	8780/M	8781	8783/M	8784	8786	8787	8789/M	8790
		8793/M							
jsv$swap_trace	8197	8256/M	8256	8315/M	8315	8322/M	8322	8324/M	8324
		8337/M	8337	8349/M	8349	8428/M	8428	8447/M	8447
		8492/M	8492	8503/M	8503	8523/M	8523	8526/M	8526
		8555/M	8555	8597/M	8597	8640/M	8640	8661/M	8661
		8665/M	8665	8674/M	8674	8726/M	8726	8825/M	8825
		8834/M	8834	8838/M	8838	9026/M	9026	9035/M	9035
		9049/M	9049	9053/M	9053	9055/M	9055	9066/M	9066
		9082/M	9082	9117/M	9117	9161/M	9161	9168/M	9168
		9186/M	9186	9223/M	9223	9232/M	9232	9239/M	9239
		9246/M	9246	9366/M	9366	9489/M	9489	9538/M	9538
		9539/M	9539	9616/M	9616	9626/M	9626	9636/M	9636
		9646/M	9646	9888/M	9888	9890/M	9890	9895/M	9895
		10025/M	10025	10039/M	10039	10042/M	10042	10051/M	10051
		10059/M	10059	10066/M	10066	10079/M	10079	10093/M	10093
		10123/M	10123	10142/M	10142	10148/M	10148	10153/M	10153
		10159/M	10159	10184/M	10184	10190/M	10190	10192/M	10192
		10201/M	10201	10284/M	10284	10300/M	10300	10303/M	10303
		10306/M	10306	10328/M	10328	10338/M	10338	10355/M	10355
		10361/M	10361	10376/M	10376	10379/M	10379	10383/M	10383
		10460/M	10460	10536/M	10536	10537/M	10537	10550/M	10550
		10553/M	10553	10566/M	10566	10595/M	10595	10644/M	10644
		10806/M	10806	10910/M	10910	10913/M	10913	10928/M	10928
		10929/M	10929	10944/M	10944	11105/M	11105	11115/M	11115
		11155/M	11155	11167/M	11167	11205/M	11205	11300/M	11300
		11312/M	11312	11338/M	11338	11389/M	11389	11399/M	11399
		11404/M	11404	11442/M	11442	11452/M	11452	11466/M	11466
		11568/M	11568						
jsv$time_to_call_job_swapper	7776	9405/M	10510/M	10841/M	10871/M	11034/M	11118/M	11170/M	11262/M
		11338/M	11389/M	11419/M	11541/M	11599	11600/M		
keypoint_enable	6182	9695/M							
kt	8241	9071/M	9072	9073	10611/M	10612	10613	10669/M	10670
		10671	11123/M	11124	11125				
last_entry_in_queue	9750	9756/M	9801	9802/P	9804				
last_ijle_p	9751	9802/P	9803/M						
last_segment_number	798	10967	10970	10986					
last_swap_status	1360	8776/M	9625						
last_swap_status	8290	8302/M	8535/M	8571/M	8577/P	8718/M	8719	8720	8722
		8723							
last_swap_status	9199	9233							
last_swap_status	11529	11553/M	11554	11555	11556	11557			
link	2218	9157	9614	10013	10124/M	10124	10727	10974	10989
link	2279	9151	9540	9544	9610	9974	10217	10721	10964
		10983							
live_aste_p	9925	10015/M	10055	10055	10077	10078	10078	10092	10094/P
		10095	10096	10097	10098/M				
lock	5391	5400							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
lock	5432	5445							
lock	7211	7214	7217	7218/M	7218	7220/M			
lock	7264	7272	7274	7276/M	7278/M	7278			
lock	8806	8822							
lock	9126	9159							
lock	9314	9387	9387	9387/M	9387/M	9387			
lock	9314	9394	9394	9394/M	9394	9394/M			
lock	9432	9478							
lock	9594	9627							
lock	9735	9754	9754	9754/M	9754/M	9754			
lock	9735	9762	9762	9762/M	9762	9762/M	9829	9829	9829/M
		9829	9829/M						
lock	9843	10227							
lock	9861	9904							
lock	10246	10365							
lock	10797	10803	10803	10803/M	10803/M	10803			
lock	10797	10811	10811	10811/M	10811	10811/M			
lock	11179	11241	11241	11241/M	11241/M	11241			
lock	11179	11248	11248	11248/M	11248	11248/M			
lock	11273	11307	11307	11307/M	11307/M	11307	11315	11315	11315/M
		11315/M	11315	11333	11333	11333/M	11333/M	11333	11338
		11338	11338/M	11338/M	11338	11352	11352	11352/M	11352/M
		11352	11389	11389	11389/M	11389/M	11389		
lock	11273	11309	11309	11309/M	11309	11309/M	11323	11323	11323/M
		11323	11323/M	11338	11338	11338/M	11338	11338/M	11341
		11341	11341/M	11341	11341/M	11389	11389	11389/M	11389
		11389/M	11393	11393	11393/M	11393	11393/M		
locked	958	5400	5445	8822	9159	9478	9627	9904	10227
		10365							
locked	3801	7274	9387	9754	10803	11241	11307	11315	11333
		11338	11352	11389					
locked_page	2224	9541	9542						
m	2070	9158	9162	9637/M	10041	10131/M	10131	10733	
max_block_in_use	7671	11483							
max_segnum	10261	10339/M	10340	10341/M	10343				
max_segnum_to_update	10262	10314/M	10316/M	10318/M	10326	10337	10339		
maximum_pages_to_swap	10889	10932/M	10934/M	10937					
maximum_time	3027	8786	8788	8790/M	8791	8793/M			
media	802	10730							
memory_reserve_request	1374	8324	8324	8324/M	8324	8324	8324	8324/M	9117
		9117	9117/M	9117	9117	9117	9117/M	10451	10453
		10454/M	10455	10456	10458	10462/M	10537	10537	10537/M
		10537	10537	10537	10537/M	11146	11148	11149/M	11150
		11151	11153	11157/M					
min_working_set_size	1269	10922							
minimum_working_set	10890	10919/M	10922/M	10926/P					
mmc$	3337	3343	3346	3349	3352	3355	3358	3361	3365
		3368	3371	3374	3377	3380	3383	3387	3390
		3393	3396	3399	3402	3406	3409	3412	3415
		3418	3421	3424	3427	3430	3433	3436	3439

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		3442	3445	3448	3451	3454	3457	3460	3463
		3466	3470	3473	3476	3479	3482	3485	3488
		3491	3494	3497	3500	3503	3506	3509	3512
		3515	3518	3521	3524	3527	3530	3534	3538
		3541	3545	3548	3551	3554	3557	3560	3563
		3566	3569	3572	3575	3578	3581	3584	3587
		3590	3593	3596	3599	3602	3606	3610	3613
		3616	3619	3622	3625	3628	3631	3634	3637
		3640	3643	3646	3649	3652	3655	3658	3661
		3664	3668	3671	3674				
mmc$assign_active_null	5935	5936							
mmc$bd_explicit_io	3052	3044							
mmc$bd_job_swapping_io	3051	3046	9473						
mmc$bd_paging_io	3051	3044							
mmc$cell_pointer	6034	6039							
mmc$heap_pointer	6035	6043							
mmc$iocb_table_size	5260	5253							
mmc$iorc_await_io_completion	4667	4660							
mmc$iorc_write_pages	4667	4658							
mmc$irs_active	4637	4631							
mmc$irs_complete	4637	4632							
mmc$irs_none	4637	4631							
mmc$kw_asid	5960	5996							
mmc$kw_clear_space	5958	5983							
mmc$kw_current_segment_length	5957	5977							
mmc$kw_error_exit_procedure	5959	5987							
mmc$kw_gl_key	5959	5981							
mmc$kw_hardware_attributes	5961	5990							
mmc$kw_inheritance	5961	5998							
mmc$kw_max_segment_length	5958	5979							
mmc$kw_preset_value	5960	5985							
mmc$kw_ps_transfer_size	5962	6006							
mmc$kw_ring_numbers	5956	5972							
mmc$kw_segment_access_control	5960	5994							
mmc$kw_segment_number	5957	5975							
mmc$kw_shadow_segment	5962	6000							
mmc$kw_software_attributes	5959	5992							
mmc$kw_wired_segment	5962	6003							
mmc$lp_not_locked	2240	9541	9542						
mmc$lss_none	6109	9700							
mmc$max_rma_list_length	3057	3062	3063						
mmc$mpt_done	3073	10122	10195	10203	10209				
mmc$mpt_page_already_exists	3074	10167							
mmc$mpt_page_table_full	3073	10138							
mmc$pfs_failed	6952	10144							
mmc$pfs_input_asid_reassigned	6953	10152							
mmc$pq_avail	1990	2036	10183						
mmc$pq_avail_modified	1991	9151/S	9543/P	10188	10926/P				
mmc$pq_free	1989	2048	10052/P	10186/P	10194/P	10202/P	10738/P		
mmc$pq_job_base	2037	9635							
mmc$pq_job_fixed	2030	2037	2049	10977/S	10991/S				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmc$pq_job_io_error	2031	8423/S	8424/S	9617/P					
mmc$pq_job_working_set	2032	2049	2050	8425/S	8426/S	9166/P	9540/S	9544/S	10721/S
		10925/S	10937/S	10964/S	10983/S	11448/S	11449/S	11455/S	11456/S
		11459/S	11460/S						
mmc$pq_shared_first	2038	9021	10199						
mmc$pq_shared_first_site	2040	2044							
mmc$pq_shared_last	2045	10200							
mmc$pq_shared_num_sites	2041	2044							
mmc$pq_shared_other	1999	2039							
mmc$pq_shared_site_01	2001	2040							
mmc$pq_shared_site_25	2025	2045							
mmc$pq_shared_task_service	1994	2038							
mmc$pq_swapped_io_error	2028	2048	9610/S	10187	10189				
mmc$pq_wired	1992	2035							
mmc$segment_fault_processor_id	6631	6685							
mmc$sequence_pointer	6034	6041							
mmc$server_iocb_table_size	4528	4531							
mmc$ssk_none	6129	6101	9712						
mmc$ssk_segment_number	6130	6099	9714						
mme$no_free_pages	3349	8441							
mme$page_table_full	3346	8890							
mmp$age_job_working_set	5718	10914							
mmp$asid	5628	10067	10373						
mmp$assign_asid	5735	9050	10043	10060	10080				
mmp$assign_page_to_monitor	5745	8889							
mmp$assign_specific_asid	5741	9056	10094						
mmp$asti	5752	9046	9989	10359					
mmp$claim_pages_for_swapin	5634	9078							
mmp$conditional_purge_all_map	5758	8472	9947	10957					
mmp$delete_page_from_monitor	5779	8532	9222	9278	9579	10418	10568		
mmp$delete_pt_entry	5792	10185	10737						
mmp$dump_shared_queue	5642	8448	9027						
mmp$free_asid	5786	9085	10146						
mmp$free_memory_in_job_queues	5648	9083	9230	9237	9244				
mmp$get_max_sdt_sdtx_pointer	5799	5809	9696	10275	10329				
mmp$get_verify_asti_in_fde	5663	5677	10367						
mmp$make_pt_entry	6917	10114							
mmp$nudge_periodic_call	6926	6931	8454						
mmp$process_page_table_full	6942	10140							
mmp$purge_all_map_proc	5691	5684	5765	8472	9947	10957			
mmp$relink_page_frame	6957	9166	9543	9617	10052	10186	10194	10202	10738
mmp$remove_stale_pages	5723	10925							
mmp$remove_swapped_shared_pages	5712	8324	9117	10492	10537				
mmp$replenish_free_queues	5656	9120							
mmp$trim_job_working_set	6963	10938							
mmp$write_page_to_disk	6971	9160							
mmt$active_io_count	4666	4545	4655						
mmt$active_segment_table	2198	7796							
mmt$active_segment_table_entry	2183	2161	2199	2228	5635	5737	5741	5787	6919
		6945	7812	9002	9865	9922	9925	9930	10254
mmt$ast_index	937	789	1426	2176	5628	5667	5736	5753	5829

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$attribute_keyword	5956	6944	8999	9000	9864	9923	9931	10255	10259
mmt$buffer_descriptor	3041	5971							
mmt$buffer_descriptor_type	3051	5471	9444						
mmt$eoi_state	946	3043							
mmt$global_page_queue_index	2048	791							
mmt$global_page_queue_list	2292	2292	7850	9003					
mmt$global_page_queue_list_ent	2282	7804							
mmt$hardware_attribute_set	6025	2292							
mmt$hardware_attributes	6013	5991							
mmt$io_identifier	3878	6025							
mmt$io_request_status	4637	4205	5474	6975	9137	9446			
mmt$io_status	4629	4630							
mmt$iocb_index	5253	4639	4652						
mmt$job_page_queue_index	2049	3885	3891						
		1975	2293	8292	8328	8328	8431	8431	8607
		8607	8964	8968	8968	9182	9182	9324	9376
		9376	9973	10214	10215	11199	11236	11236	11338
		11338	11389	11389					
mmt$job_page_queue_list	2293	1381	5637	5648					
mmt$link	2206	2184	2218	2219	2279				
mmt$lock_segment_status	6109	5900							
mmt$locked_page	2240	2224							
mmt$make_pt_entry_status	3073	6921	9927						
mmt$max_sdt	5839	5843							
mmt$max_sdt_p	5843	5801	9673	10266	10268				
mmt$max_sdtx	5924	5928							
mmt$max_sdtx_p	5928	5802	9674	10267	10269				
mmt$memory_reserve_request	2260	1374							
mmt$page_age	2247	2227	2251	2251					
mmt$page_frame_index	2144	2136	2138	2139	2140	2208	2208	2262	2263
		2839	5551	5623	5643	5793	6918	6957	6973
		8289	8291	8521	8565	8619	8663	8963	9139
		9140	9322	9381	9448	9508	9534	9602	9604
		9607	9917	9932	9933	10574	10717	10719	10893
		11198	11215	11338	11389	11437			
mmt$page_frame_queue_id	2050	2137	2192	2222	5727	6958	9915	10894	
mmt$page_frame_table	2233	7838							
mmt$page_frame_table_entry	2217	2159	2233	6920	9603	9606	9918		
mmt$page_queue_list_entry	2278	2283	2293	5723					
mmt$pt_full_status	6952	6946	9934						
mmt$reassignable_page_frames	7889	7886							
mmt$rma_list_entry	3065	3060	4236	4251					
mmt$rma_list_index	3062	3060							
mmt$rma_list_length	3063	3042							
mmt$sdtx_stream_data	5907	5903							
mmt$segment_access_condition	6658	6686							
mmt$segment_access_rights	6073	5899							
mmt$segment_access_state	6079	5894							
mmt$segment_descriptor	5826	5836	5840	7900					
mmt$segment_descriptor_extended	5892	5921	5925						
mmt$segment_descriptor_table_ex	5920	9661							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
mmt$segment_inheritance	5942	5896	5999						
mmt$segment_pointer_kind	6034	6038							
mmt$segment_reservation_state	6119	5897							
mmt$server_iocb_entry	4535	4213	4532						
mmt$server_state	4681	4537							
mmt$shadow_info	6094	5901							
mmt$shadow_reference_info	6283	6186							
mmt$shadow_segment_kind	6129	6098							
mmt$software_attribute_set	6027	5898	5993						
mmt$software_attributes	6021	6027							
mmt$sub_reqcodes	4667	4542	4657						
mmt$write_page_to_disk_status	3076	6977	9142						
mmt$xcb_page_wait_info	6294	6172							
mmv$aggressive_aging_level_2	7785	8324	9016	9025	9034	9117	10452	10537	11147
mmv$aging_algorithm	7792	10903							
mmv$ast_p	7796	5671	5671	5672	9047	9990	10068	10347	10368
		10368	10368						
mmv$gpql	7804	9022	9151	9610					
mmv$initial_job_fixed_ast_entry	7812	9059							
mmv$last_active_shared_queue	7850	9021							
mmv$max_template_segment_number	7823	10316							
mmv$min_avail_pages	7826	8534							
mmv$multiple_page_maps	7833	5683	5763	8472	9947	10957			
mmv$pft_p	7838	9157	9158	9158/S	9159/P	9162/S	9541/S	9541/S	9541
		9542	9613	9633	9639	10013	10124	10125/M	10126/M
		10127/M	10128/M	10173	10727	10728	10729/P	10733/S	10966/P
		10974	10985/P	10987	10989				
mmv$pt_p	7877	9158	9162	9541	9541	9637/M	10130/M	10131/M	10132/M
		10172	10733						
mmv$reassignable_page_frames	7886	5561	8324	8328/M	8328	8333	8333	8392/M	8393
		8395/M	8395	8429/M	8429	8431/M	8431	8481/M	8482
		8488/M	8488	8532/M	8532	8534	8541/M	8541	8543/M
		8543	8566/M	8566	8921/M	8921	8976/M	8977	9014
		9015	9015	9024	9024	9032	9033	9034	9117
		9179/M	9180	9182/M	9182	9222/M	9222	9293/M	9293
		9573/M	9573	9575/M	9576	9579/M	9579	10418/M	10418
		10452	10537	10568/M	10568	10572/M	10572	10598/M	10598
		10600/M	10601	10617/M	10618	10622/M	10622	10646/M	10647
		10651/M	10651	10656/M	10657	11128/M	11129	11147	11236/M
		11236	11251	11251	11450/M	11451	11457/M	11457	11461/M
		11462	11464/M	11464					
mmv$reserved_page_count	7844	8324/M	8324	9117/M	9117	10457/M	10457	10537/M	10537
		11152/M	11152						
mmv$swapping_aic	7856	10921							
mmv$time_changed_global_asid	7861	10295							
mmv$time_changed_template_asid	7869	10298							
mmv$time_map_last_purged	5776	5764	8472	9947	10957				
mmv$time_to_call_mem_mgr	6937	6928/M	8454/M						
modified_pages_removed	9138	9150/M	9167/M	9167	9176	9178			
modified_pages_removed	10891	10926/P	10929/P	10993					
monitor_lock	779	5400/P	8822/P	9159/P	9478/P	9627/P	9904/P	10227/P	10365/P

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
move_am_to_am	9524	9111	9549						
mpt_count	9926	10106/M	10143/M	10143	10144				
mpt_status	9927	10115/P	10121	10195/M	10203/M	10209			
msg	9928	9958/M	9959/M	9960/P					
msg	10716	10750/M	10751/M	10752/P	10755/M	10756/M	10757/P	10760/M	10761/M
		10762/P							
mtc$job_fixed_segment	3083	5361	5398	8532	8822	8888	8903/P	8912	9114
		9159	9222	9274	9464	9478	9579	9627	9904
		10056	10227	10365	10418	10568	10729	10900	10966
		10985	11134						
mtc$scb_max_hardware_status	8017	7976							
mtp$cst_p	6981	9405	10513	10841	10871	11034	11118	11170	11262
		11338	11389	11419	11600				
mtp$error_stop	5428	8297	8310	8532	8707	8734	8842	9222	9286
		9411	9579	9692	9760	9789	9793	9819	9823
		9956	10170	10197	10205	10418	10568	10578	10662
		11266	11338	11389	11415				
mtp$set_interlock	5432	5400	5453	8822	9159	9478	9627	9904	10227
		10365							
mtp$set_status_abnormal	7197	7204	8835	8839	9036	9086	10150	11355	11367
		11383							
mtt$idle_status_block	7954	7941							
mtt$monitor_interlock	953	779	5432						
mtt$scb_180_status	7930	7921							
mtt$scb_hardware_status	7978	7916	8001						
mtt$scb_hardware_status_count	7976	7979							
mtt$scb_hardware_status_msg	7982	7988							
mtt$scb_hardware_status_msgs	7987	7924							
mtt$scb_hardware_status_options	7968	7978	7987						
mtt$smu_communications_block	7915	7908							
mtt$step_status_block	7949	7942							
mtt$system_idle_update_request	8022	7956	7956						
mtt$system_status_block	7940	7931							
mtt$system_step_update_request	8021	7951	7951						
mtv$cst0	6987	6983	9405	10513	10841	10871	11034	11118	11170
		11262	11338	11389	11419	11600			
mtv$scb	7908	10701	10702						
mtv$system_job_monitor_xcb_p	8046	10275/P							
nat$received_message_descriptor	6310	6303	6312						
nat$received_message_list	6302	6154							
need_ajl	8263	8532/M	8532	8532					
need_ajl	9196	9222/M	9222	9222					
need_ajl	9265	9269/M	9270	9281					
need_ajl	9567	9579/M	9579	9579					
need_ajl	10246	10418/M	10418	10418					
need_ajl	10521	10568/M	10568	10568					
new_asid	9863	9896	9897						
new_asid	9929	10043/P	10049/P	10060/P	10067/P	10070/P	10080/P	10085/P	10140/P
		10154/P	10157						
new_aste_p	9865	9898	9903	9904/P					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
new_aste_p	9930	10043/P	10045/M	10046/M	10047/M	10048/M	10049/P	10060/P	10061/M
		10062/M	10063/M	10064/M	10068/M	10070/P	10080/P	10081/M	10082/M
		10083/M	10084/M	10085/P	10140/P	10154/P			
new_asti	9864	9905							
new_asti	9931	10043/P	10049/P	10057/M	10058	10060/P	10067/P	10068/S	10070/P
		10080/P	10085/P	10140/P	10154/P	10158			
new_entry_status	5512	5519/S	5520/S	5522	5525	5529			
new_entry_status	8263	8324/S	8324/S	8324	8324	8324			
new_entry_status	9093	9117/S	9117/S	9117	9117	9117			
new_entry_status	9314	9329/S	9329/S	9329	9329	9329	9331/S	9331/S	9331
		9331	9331	9336/S	9336/S	9336	9336	9336	
new_entry_status	10429	10447/S	10447/S	10447	10447	10447			
new_entry_status	10521	10537/S	10537/S	10537	10537	10537			
new_entry_status	11076	11116/S	11116/S	11116	11116	11116	11142/S	11142/S	11142
		11142	11142	11168/S	11168/S	11168	11168	11168	
new_entry_status	11179	11207/S	11207/S	11207	11207	11207			
new_entry_status	11273	11308/S	11308/S	11308	11308	11308	11338/S	11338/S	11338
		11338	11338	11338/S	11338/S	11338	11338	11338	11338/S
		11338/S	11338	11338	11338	11360/S	11360/S	11360	11360
		11360	11362/S	11362/S	11362	11362	11362	11369/S	11369/S
		11369	11369	11369	11373/S	11373/S	11373	11373	11373
		11389/S	11389/S	11389	11389	11389	11389/S	11389/S	11389
		11389	11389	11389/S	11389/S	11389	11389	11389	
new_queue	9738	9756/S	9758	9759	9800	9807/S	9811/S	9812/S	9812/S
		9817/S	9818/S	9822/S	9827				
new_swap_status	8768	8780/S	8781/S	8783/S	8784/S	8786/S	8787/S	8789/S	8790/S
		8793/S	8798						
next_cyclic_aging_time	1273	9115/M	9115	11135/M	11135				
next_ijl_ordinal	11530	11551/M	11585	11596					
next_index	11050	9359	9359/M	11065	11066/M	11104	11104/M	11166	11166/M
		11201	11201/M	11299	11299/M	11338	11338/M	11389	11389/M
next_pfti	9139	9157/M	9171						
next_pfti	9602	9614/M	9651						
next_pfti	9932	10013/M	10214	10214	10214	10214	10217/M	10220	
next_pfti	10717	10727/M	10742						
next_swap_status	1359	8323/M	8730	8731/P	8732/M	8737	8738/P	8739/M	10538/M
		10555/M	10575/M	10834/M	10862/M	10864/M	10866/M	11400/M	11583
nlc$cc_connect_confirm	6342	6333							
nlc$cc_connect_request	6341	6331							
nlc$cc_expedited_data	6347	6333							
nlc$cc_max_pdu_kind	6349	6352							
nlc$channel_connection_pdu	6365	6317							
nlc$channelnet_pdu	6365	6319							
nlt$cc_pdu_kind	6352	6330							
nlt$cc_seq#_or_connect_time	6329	6318							
nlt$cc_sequence_number	6355	6334							
nlt$device_identifier	6362	6313							
nlt$pdu_type	6365	6316							
normal	2898	7202/M	8300/M	8304	8304	8380	8389/M	8441	8446/M
		8450	8451/M	8459	8460/M	8501	8512/M	8532	8622
		8638	8647/M	8694	8817/M	8835/M	8839/M	8890	8897

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		9008/M	9036/M	9065	9081	9086/M	9222	9285	9397
		9408/M	9488	9503	9511/M	9516	9579	9691	9967/M
		10150/M	10418	10568	10594	10643	11114	11290/M	11311
		11338	11338/M	11355/M	11367/M	11383/M	11389	11389/M	
notify_swapper_when_io_complete	1367	8398/M	8403/M	9517/M	10567/M	10858/M			
now	7890	5561	8324	8333	8534	8543/M	8543	8566/M	8566
		9014	9015	9024	9032	9033	9117	10452	10537
		10651/M	10651	11147	11251	11464/M	11464		
null_sva	5761	5768							
null_sva	8263	8472							
null_sva	9843	9947							
null_sva	10880	10957							
num_sched_swapping_calls	11042	9359	11051	11066	11104	11166	11201	11299	11338
		11389							
offset	2088	9628	9634	9634	9647/P	10987			
offset	5352	5357/M	5364/M	5367					
offset	5391	5398/M	5398/M	5398					
offset	8806	8822/M	8822/M	8822					
offset	9126	9159/M	9159/M	9159					
offset	9432	9478/M	9478/M	9478					
offset	9594	9627/M	9627/M	9627					
offset	9843	10056/M	10056/M	10056	10227/M	10227/M	10227		
offset	9861	9904/M	9904/M	9904					
offset	10246	10365/M	10365/M	10365					
offset	10710	10729/M	10729/M	10729					
offset	10880	10966/M	10966/M	10966	10985/M	10985/M	10985		
old_asid	9872	9885/M	9894						
old_entry_status	5515	5517/M	5519/S	5520/S	5524	5528			
old_entry_status	8263	8324/M	8324/S	8324/S	8324	8324			
old_entry_status	9093	9117/M	9117/S	9117/S	9117	9117			
old_entry_status	9314	9329/M	9329/S	9329/S	9329	9329	9331/M	9331/S	9331/S
		9331	9331	9336/M	9336/S	9336/S	9336	9336	
old_entry_status	9323	9326/M	9340						
old_entry_status	10429	10447/M	10447/S	10447/S	10447	10447			
old_entry_status	10521	10537/M	10537/S	10537/S	10537	10537			
old_entry_status	11076	11116/M	11116/S	11116/S	11116	11116	11142/M	11142/S	11142/S
		11142	11142	11168/M	11168/S	11168/S	11168	11168	
old_entry_status	11179	11207/M	11207/S	11207/S	11207	11207			
old_entry_status	11273	11308/M	11308/S	11308/S	11308	11308	11338/M	11338/S	11338/S
		11338	11338	11338/M	11338/S	11338/S	11338	11338	11338/M
		11338/S	11338/S	11338	11338	11360/M	11360/S	11360/S	11360
		11360	11362/M	11362/S	11362/S	11362	11362	11369/M	11369/S
		11369/S	11369	11369	11373/M	11373/S	11373/S	11373	11373
		11389/M	11389/S	11389/S	11389	11389	11389/M	11389/S	11389/S
		11389	11389	11389/M	11389/S	11389/S	11389	11389	
old_entry_status	11273	11338/M	11338	11389/M	11389				
old_swap_status	8773	8775/M	8776	8780/S	8781/S	8783/S	8784/S	8786/S	8787/S
		8789/S	8790/S	8793/S					
open_validating_ring_number	5893	9705	10351						
osc$aging_interval_maximum	2339	2342							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
osc$base_exception	103	109	2370	2372	3740				
osc$call_instruction	6545	6553							
osc$data_read	6544	6553							
osc$free_running_clock_maximum	1139	1136							
osc$invalid_ring	974	1014							
osc$max_channel_number	5046	5049							
osc$max_fault_contents	6698	6692							
osc$max_idle_count	7088	7096							
osc$max_integer	4694	4699	4700						
osc$max_name_size	2305	2309	2312						
osc$max_number_of_processors	7073	7001	7229						
osc$max_page_frames	2055	1421	1422	1974	1976	2144	2185	2280	2286
		7890	7891	7892	7893	8293	8815	8867	8868
		8935	8937	9005	9096	9138	9437	9526	9862
		9873	9938						
osc$max_page_size	1127	1123							
osc$max_page_table_entries	2056	2059							
osc$max_ring	973	1014	1015						
osc$max_segment_length	997	1020	5904	5935	9699				
osc$max_status_condition_code	2864	2860	2876						
osc$max_status_condition_number	7207	7198							
osc$max_string_size	2880	2883	2886	2891					
osc$max_tasks	899	896							
osc$maximum_offset	996	997	1017	1017	1018				
osc$maximum_processor_id	6570	6566							
osc$maximum_processor_number	7065	7060							
osc$maximum_processors	7069	7065	7073						
osc$maximum_segment	995	1016							
osc$min_ecc	102	103							
osc$min_integer	4693	4697	4698						
osc$min_page_size	1126	1123							
osc$min_ring	972	1015							
osc$pr_base_constant	3694	7214	7271	9387	9394	9754	9762	9829	10803
		10811	11241	11248	11307	11309	11315	11323	11333
		11338	11338	11341	11352	11389	11389	11393	
osc$purge_all_page_seg_map	5707	5768	8472	9947	10957				
osc$segnum_job_fixed_heap	3092	10357							
osc$sva_purge_all_page_map	5703	5686							
osc$task_time_slice_maximum	1697	1700							
osc$v1_invalid_entry	5854	9698	10344						
osk$entry	2762	9144	9217	10898	11098	11288	11535		
osk$exit	2763	9189	9249	11000	11173	11422	11603		
osk$m	2801	9072	9073	9368	10612	10613	10670	10671	10986
		10987	10991	10993	10994	10996	11124	11125	11213
		11338	11389						
osk$performance	2766	9072	9073	9368	10612	10613	10670	10671	10986
		10987	10991	10993	10994	10996	11124	11125	11213
		11338	11389						
osk$system_class	2776	2760	2761	2762	2763	2764	2765	2766	
ost$aging_interval	2342	1270	1271						
ost$asid	2091	1365	2087	2163	2174	2175	2190	5480	5495

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		5629	5656	5735	5752	5786	5860	5997	6943
		9001	9863	9872	9920	9929	10253		
ost$binary_unique_name	1035	781	4624	7516	7573				
ost$byte_count	2081	5472							
ost$compare_swap_lock	1145	7335							
ost$cp_time	1804	1772	1817	6171					
ost$cp_time_value	1802	1265	1266	1407	1805	1806	6184		
ost$cpu_element_id	7057	7032							
ost$cpu_idle_statistics	7081	7035							
ost$cpu_memory_port_mask	7059	7008							
ost$cpu_running_or_stepped	7109	7106	7106						
ost$cpu_state	7104	7017							
ost$cpu_state_reason	7115	7038							
ost$cpu_state_table	7004	6981	7001	10507					
ost$cs_lock	1146	6152							
ost$cst_trace_control	7136	7036							
ost$date_time	4704	4169							
ost$debug_code	6544	6532							
ost$debug_list	6540	6444							
ost$debug_list_entry	6531	6540							
ost$debug_mask	6550	6443							
ost$exchange_package	6393	6139							
ost$execute_privilege	5873	5855	5868						
ost$execution_control_block	6138	5800	6164	7018	7239	7294	8046	9680	10272
		10694							
ost$external_code_base_pointer	7476	7399	7432						
ost$external_interrupt_request	7124	7024							
ost$family_name	2352	2347							
ost$flags	6450	6400							
ost$frame_descriptor	6508	6523							
ost$free_running_clock	1136	797	1264	1273	1376	1377	1378	1379	1413
		1423	1424	1425	1637	1649	2189	6170	7642
		7861	7869	8771	8772	11054	11541	11599	
ost$global_task_id	890	799	883	1260	1370	1399	3884	4197	6149
		6150	6582	6785	7013	7287			
ost$halfword	2096	7077							
ost$idle_type	7100	7095							
ost$key_lock	1003	5861	5982						
ost$key_lock_value	1009	1006	6467	6469					
ost$keypoint_class	6482	6413	6484						
ost$keypoint_mask	6484	6416							
ost$logical_processor_id	7060	7009							
ost$minimum_save_area	6518	6405	6493	6679					
ost$monitor_condition	6369	6376							
ost$monitor_conditions	6376	6406	6410	6498	6754	6768			
ost$monitor_fault	6675	6624							
ost$monitor_fault_contents	6692	6688							
ost$name	2312	2301	2350	2352	3915	3945	5229	6198	6262
		6717	7572	7581					
ost$non_negative_integers	4699	4152							
ost$p_register	6465	6394	6519	6746	6752				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$page_id	2061	2071							
ost$page_size	1123	1104	8053						
ost$page_table	2075	7877							
ost$page_table_entry	2066	2075	2160						
ost$page_table_index	2059	2075	2225						
ost$paging_statistics	1840	1818	6179						
ost$parcel	2098	7030	7031						
ost$physical_channel_number	5049	4991							
ost$pre_processed_for_reconfig	7132	7039							
ost$processor_element_id	7076	7057							
ost$processor_element_number	7085	7078							
ost$processor_id	6566	6142	6560						
ost$processor_id_set	6560	6141	7918	7919	8031	10701			
ost$processor_model_number	1053	1037	2358	7079					
ost$processor_serial_number	1131	1036	2359	7080					
ost$pva	1025	6438	6456	6470	6676	6769	8236		
ost$read_privilege	5876	5856	5869						
ost$real_memory_address	2079	4015	4116	4858	7029				
ost$register_number	6461	6435	6504	6512	6513	6514			
ost$ring	1014	1026	5858	5859	5893	5973	5974	6455	
ost$ring1_termination_reason	6578	6175							
ost$segment	1016	798	1027	5976	6100	6433	6534	9675	10265
		10895							
ost$segment_access_control	5866	5995							
ost$segment_descriptor	5853	5827							
ost$segment_length	1020	4540	4651	5978	5980	6002	6004	6007	
ost$segment_offset	1017	1028	2088	4539	5271	5470	5908	6535	6537
ost$signature_lock	1147	3921	7351	7352					
ost$stack_frame_save_area	6492	6526	6712						
ost$state_tables	7001	6987							
ost$status	2848	5989	6747						
ost$status_condition	2872	2899	4633	4654					
ost$status_condition_code	2876	2851	2872						
ost$string	2889	2852							
ost$string_size	2883	2890							
ost$system_flag	6853	6849							
ost$system_virtual_address	2086	2230	3045	5681	6917	6942			
ost$task_index	896	891	930	931	5294	7043	7292		
ost$task_time_slice	1700	1686							
ost$top_of_stack_pointer	6453	6445							
ost$trap_enable	6487	6402	6743						
ost$user_condition	6379	6386							
ost$user_conditions	6386	6404	6408	6496	6525	6715	6755		
ost$user_identification	2345	1259	7365						
ost$user_name	2350	2346							
ost$valid_relative_pointer	1023	804	807	6168	6169				
ost$valid_ring	1015	6445							
ost$vector_simulation_control	8029	7920							
ost$virtual_machine_identifier	6475	6396	6398	6520	7469	7478			
ost$wait	4676	4653							
ost$word	2100	4052							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
ost$write_privilege	5879	5857	5870						
ost$x_register	6462	6435	6504						
osv$cpus_logically_on	7229	7213	7270	9387	9394	9754	9762	9829	10803
		10811	11241	11248	11307	11309	11315	11323	11333
		11338	11338	11341	11352	11389	11389	11393	
osv$page_size	8053	8823/P	8830	8873	8887	8907	8947	8948	8949
		9481	9482	9486/P	9487/P	10172	10987		
osv$time_to_check_asyn	6939	6929/M	8454/M	9405/M	10515/M	10841/M	10871/M	11034/M	11118/M
		11170/M	11262/M	11338/M	11389/M	11419/M	11600/M		
page_age_limit	10892	10918/M	10921/M	10925/P					
page_aging_interval	1270	10912							
page_count	94	8330/M	8330	11238/M	11238				
page_count	3042	9485/M							
page_count	9448	9480/M	9481	9482/M	9485	9487/P			
page_status	10718	10731/P	10732	10732					
page_table_entry	2160	9896/M	10041	10130	10131	10132			
pageid	2071	9896/M							
pages_in_memory	2185	10046/M	10062/M	10082/M	10095/M	10095	10145		
pages_needed	2839	11411/P							
pages_removed	11437	11442/P	11444	11449	11451	11452/P	11456	11457	11460
		11462	11464	11466/P					
pft_entry	2159	9885	9894	9897/M	9898/M	10001	10015	10112/M	10113/M
		10114/P	10114/P	10114/P	10124/M	10125	10129	10139	10140/P
		10145	10146/P	10146/P	10168				
pft_link	2184	9630	10047/M	10048/M	10063/M	10064/M	10083/M	10084/M	10096/M
		10096	10097/M	10097					
pfte_p	9603	9613/M	9614	9615	9627/P	9628	9630	9645	9645
		9647/P							
pfti	9140	9151/M	9156	9156	9157/S	9158/S	9158/S	9159/S	9160/P
		9162/S	9166/P	9171/M					
pfti	9534	9540/M	9541	9541/S	9541	9541/S	9541/S	9542/S	9543/P
		9544/M							
pfti	9604	9610/M	9612	9612	9613/S	9617/P	9651/M		
pfti	9933	9974/M	10012	10012	10013/S	10052/P	10053/M	10105	10114/P
		10124/S	10125/S	10126/S	10127/S	10128/S	10194/P	10202/P	10220/M
pfti	10719	10721/M	10726	10726	10727/S	10728/S	10729/S	10733/S	10737/P
		10738/P	10742/M						
pfti	10893	10964/M	10965	10965	10966/S	10974/M	10974/S	10983/M	10984
		10984	10985/S	10987/S	10989/M	10989/S			
pmc$kill_task_flag	6853	6869							
pmc$mainframe_id_size	4722	4719							
pmc$max_signal_contents	6836	6830							
pmc$max_task_id	6591	6588							
pmc$processor_model_number_size	4730	4722	4727						
pmc$processor_serial_num_size	4780	4723	4777						
pmt$binary_mainframe_id	2357	1263	4143						
pmt$condition_identifier	6665	6659							
pmt$cpu_model_number	1113	1102	1109						
pmt$cpu_serial_number	1116	1103	1108						
pmt$initialization_value	1190	796	5986						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
pmt$mainframe_id	4719	4144	4955						
pmt$program_name	5229	5153							
pmt$sense_switches	2366	1274							
pmt$signal	6792	6786							
pmt$signal_contents	6830	6794							
pmt$signal_id	6797	6793							
pmt$task_id	6588	6166	6583						
pmt$vector_simulation	8036	8030							
poll_jobs_being_swapped	11548	11548	11586	11597					
poll_swapper_again	11531	11546/M	11593/M	11599					
poll_swapping	9318	9406/M							
poll_swapping	11020	11031/P	11033						
poll_swapping	11273	11338/M	11389/M						
poll_swapping	11286	11291/M	11310/P	11337/P	11388/P	11401/P	11406/P	11418	
poll_swapping	11532	11590/P	11592						
pqle	2283	9022	9151	9610					
process_io_error_on_swapin	9552	8641	8666	9564					
process_io_error_on_swapout	9567	8505	8527	9590					
processor_selections	6141	10701	10702/M						
processors_logically_on	7918	10701	10702						
pt_full_status	9934	10141/P	10144	10152					
pti	2225	9158/S	9162/S	9541/S	9541/S	9637/S	10129	10733/S	
pti	9935	10129/M	10130/S	10131/S	10132/S	10168	10172/S		
ptk$ajl_for_swap_out	3147	9368	11213	11338	11389				
ptk$performance_base	2706	3114	3117	3120	3123	3126	3129	3132	3135
		3138	3141	3144	3147	3150	3153	3156	3159
		3162	3165	3168	3171	3174	3177	3180	3183
		3186	3189	3192	3195	3198	3201	3204	3207
		3210	3213	3216	3219	3222	3225	3228	3231
		3234	3237	3240	3243	3246	3249	3252	3255
		3258	3261	3264	3267	3270	3273	3276	3279
		3282	3285	3288	3291	3294	3297	3300	3303
		3306	3309	3312	3315	3318	3321	3324	3327
		3330	3333						
ptk$swapin_job_name_1	3153	9072	10612	10670	11124				
ptk$swapin_job_name_2	3156	9073	10613	10671	11125				
ptk$swapping_ijl_ordinal	3210	10997							
ptk$swapping_job_fixed	3201	10992							
ptk$swapping_modified_pages	3204	10993							
ptk$swapping_page_number	3213	10988							
ptk$swapping_removed_pages	3207	10994							
ptk$swapping_segment	3198	10986							
ptr_to_sfd	8863	8902/M	8905/P						
pva	8236	8532/M	9222/M	9274/M	9464/M	9467	9579/M	10418/M	10568/M
queue	9003	9021	9022/S						
queue_file_information	1396	9962	9963/M	9965/M					
queue_id	2122	8296	8595	9757	9827/M	11204			
queue_id	2192	10199	10200						
queue_id	2222	9635	10183	10187	10188	10189			
queue_id	8292	8607	8609/S						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
queue_id	9324	9376	9377/S						
queue_id	11273	11338	11338/S	11389	11389/S				
ready_task_count	1821	9185	9333	10901	10943	11338	11389		
reclaim_io_error_pages	9594	8324	9117	9654	10475	10537			
recover_job_dm_tables	9658	9730	10285						
recovery	9936	9980/M	10002	10040	10077				
recovery	10263	10278/M	10283						
request_block	11274	11290/M	11292	11297	11310/P	11311	11337/P	11338/P	11355/P
		11359	11366	11367/P	11372	11383/P	11388/P	11389/P	11401/P
		11406/P	11411/P	11413/P					
request_type	9314	9359							
request_type	11052	9359/M	11067/M	11104/M	11166/M	11201/M	11299/M	11338/M	11389/M
request_type	11059	11067							
request_type	11076	11104	11166						
request_type	11179	11201							
request_type	11273	11299	11338	11389					
requested_page_count	2262	8324	8324	8324	8324	8324/M	9117	9117	9117
		9117	9117/M	10451	10453	10456	10458	10462/M	10537
		10537	10537	10537	10537/M	11146	11148	11151	11153
		11157/M							
reserved_page_count	2263	8324/M	8324	9117/M	9117	10454/M	10455	10537/M	10537
		11149/M	11150						
reset_changed_asid	9937	9884/M	9982/M	10231/P					
reset_changed_asid	10250	10297							
reset_sdt_addresses	10249	10326	10331						
reset_sdt_xcb_tables	10246	10231	10425	10676					
reset_swapped_job_mm_tables	9843	8692	10233						
residence	1220	5355	5398	5672	8822	9159	9478	9627	9708
		9709/M	9715	9716/M	9903	9904	10038	10044/M	10056
		10182	10227	10349	10363	10364	10365	10366	10368
		10728	10729	10966	10985				
residence	5353	5355/M	5359	5360					
residence	5391	5398/M	5398	5398					
residence	8806	8822/M	8822	8822					
residence	9126	9159/M	9159	9159					
residence	9432	9478/M	9478	9478					
residence	9594	9627/M	9627	9627					
residence	9843	10056/M	10056	10056	10227/M	10227	10227		
residence	9861	9904/M	9904	9904					
residence	10246	10365/M	10365	10365					
residence	10710	10729/M	10729	10729					
residence	10880	10966/M	10966	10966	10985/M	10985	10985		
residence_time	7734	9389/M	9390	11242/M	11244	11338/M	11338	11389/M	11389
restart_idled_tasks	10429	8324	9117	10495	10537				
rma	2072	10172							
rma	8864	8906/P	8907						
rma	10264	10332/P	10333	10334					
rmc$external_vsn_size	7540	7546							
rmc$recorded_vsn_size	7543	7553							
rmc$unspecified_file_class	7509	7502							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
rmt$external_vsn	7546	7560							
rmt$recorded_vsn	7553	7562	7559						
rmt$volume_descriptor	7558	7565							
s	8244	9071/M	10611/M	10669/M	11123/M				
scan_available_modified_queue	9155	9155	9172						
sched_requests	11051	9359/M	9359/M	9359/M	11067/M	11068/M	11069/M	11104/M	11104/M
		11104/M	11166/M	11166/M	11166/M	11201/M	11201/M	11201/M	11299/M
		11299/M	11299/M	11338/M	11338/M	11338/M	11389/M	11389/M	11389/M
sched_trace	11058	9359	11071	11104	11166	11201	11299	11338	11389
scheduler_swapins	7736	11321/M	11322						
sdt_offset	6168	5806	9696	10276	10329				
sdt_p	5801	5806/M							
sdt_p	9658	9696/M							
sdt_p	9673	9696/P	9698						
sdt_p	10246	10276/M	10329/M						
sdt_p	10266	10329/P	10332/P	10344	10345	10347/S	10354/M	10358/M	10360/M
		10374/M	10375/M	10378/M	10382/M				
sdtx_offset	6169	5807	9696	10276	10329				
sdtx_p	5802	5807/M							
sdtx_p	9658	9696/M							
sdtx_p	9674	9696/P	9699/M	9700/M	9705	9706/M	9708	9709/M	9712
		9713	9714	9716/M					
sdtx_p	10246	10276/M	10329/M						
sdtx_p	10267	10329/P	10348	10351	10352	10363	10364	10365/P	10366
		10367/P							
sdtx_table	5921	9707							
sdtx_table	5925	9699/M	9700/M	9705	9706/M	9708	9709/M	9712	9713
		9714	9716/M	10348	10351	10352	10353	10363	10364
		10365/P	10366	10367/P					
seg	1027	8532/M	9222/M	9274/M	9464/M	9467	9579/M	10418/M	10568/M
seg	5351	5358/M	5361/M	5367					
seg	5391	5398/M	5398/M	5398					
seg	8806	8822/M	8822/M	8822					
seg	9126	9159/M	9159/M	9159					
seg	9432	9478/M	9478/M	9478					
seg	9594	9627/M	9627/M	9627					
seg	9843	10056/M	10056/M	10056	10227/M	10227/M	10227		
seg	9861	9904/M	9904/M	9904					
seg	10246	10365/M	10365/M	10365					
seg	10710	10729/M	10729/M	10729					
seg	10880	10966/M	10966/M	10966	10985/M	10985/M	10985		
segment_link	2219	9639	10112/M	10113/M					
segment_lock	5900	9700/M							
segment_number	9675	9697	9698/S	9699/S	9700/S	9705/S	9706/S	9707/S	9708/S
		9709/S	9712/S	9713/S	9714/S	9716/S			
segment_number	10265	10343	10344/S	10345/S	10347/S	10348/S	10351/S	10352/S	10353/S
		10354/S	10354/S	10357	10358/S	10360/S	10363/S	10364/S	10365/S
		10366/S	10367/S	10374/S	10375/S	10378/S	10382/S		
segment_number	10895	10968/M	10970/M	10972/S	10973/S				
segment_table_address_1	6437	10333/M							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
segment_table_address_2	6439	10334/M							
segment_table_length	6433	9697	10341						
service_class	1415	9388/S	9389/S	9391/S	9392/S	11242/S	11243/S	11245/S	11246/S
		11316/S	11317/S	11320/S	11321/S	11338/S	11338/S	11338/S	11338/S
		11389/S	11389/S	11389/S	11389/S				
set_ijle_work	11482	11482	11510						
set_polling_event	8266	8301/M	8383/M	8456/M	8462/M	8505/P	8508/M	8527/P	8646/M
set_polling_event	9570	9585/M							
set_swapping_event	10498	9405	10518	10841	10871	11034	11118	11170	11262
		11338	11389	11419	11600				
sfc$no_limit	3737	8823/P							
sfd_cell_p	8865	8888/M	8889/P	8906/P					
sfd_offset	8866	8873/M	8887/M	8887	8888	8903/P			
sfd_p	1384	8532	8532/M	8532/P	8532/M	8693/P	8902	8915/M	8916/M
		9221	9222	9222/M	9222/P	9222/M	9273	9275/M	9278/P
		9288/M	9463	9465/M	9466/M	9579	9579/M	9579/P	9579/M
		10417	10418	10418/M	10418/P	10418/M	10568	10568/M	10568/P
		10568/M							
sfd_p	8234	8532/M	8532	9222/M	9222	9273/M	9275	9463/M	9465
		9579/M	9579	10418/M	10418	10568/M	10568		
sfd_p	8936	8945	8946	8948	8948				
sfd_p	9847	9885	9886	9892	9893	9894	9896/M	9897/M	9898/M
		9899/M	9954	9975	9991	10001	10014		
sfd_page_count	8867	8871/M	8872	8889/P	8921				
sfd_page_count	8937	8946/M	8948	8948	8948	8949	8950/M	8950	8953
sfd_purge_timestamp	1379	8472/P	8532/M	9222/M	9300/M	9579/M	9947/P	10418/M	10568/M
sfid	2193	5671	9159/P	9627/P	9903	9904/P	9991	9992/M	10038
		10044/M	10055	10055	10056/P	10078	10078	10182	10348
		10349	10368	10728	10729/P	10966/P	10985/P		
sfid	5343	5355	5356	5357	5368				
sfid	5391	5398/P							
sfid	5391	5398	5398	5398	5398				
sfid	5665	5671	5672						
sfid	5895	9706/M	9707	9708	9709/M	10348	10352	10353	10363
		10364	10365/P	10366	10367/P				
sfid	8806	8822/P							
sfid	8806	8822	8822	8822	8822				
sfid	9126	9159/P							
sfid	9126	9159	9158	9159	9159				
sfid	9432	9478/P							
sfid	9432	9478	9478	9478	9478				
sfid	9435	9478/P							
sfid	9594	9627/P							
sfid	9594	9627	9627	9627	9627				
sfid	9843	10056	10056	10056	10056	10227	10227	10227	10227
sfid	9843	10227/P							
sfid	9861	9904/P							
sfid	9861	9904	9904	9904	9904				
sfid	10246	10365/P							
sfid	10246	10365	10365	10365	10365				
sfid	10246	10368	10368						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
sfid	10710	10729	10729	10729	10729				
sfid	10880	10966	10966	10966	10966	10985	10985	10985	10985
sft$counter	1850	1275	1277	1278	1280	1819	1820		
sft$file_space_limit_kind	3737	5306	5902						
shadow_info	5901	9712	9713	9714	9716/M				
shadow_segment_kind	6098	9712	9713						
shadow_sfid	6097	9714	9716/M						
soon	7891	8333	8395/M	8395	8429/M	8429	8488/M	8488	8532/M
		8532	8541/M	8541	8921/M	8921	9015	9024	9034
		9222/M	9222	9293/M	9293	9573/M	9573	9579/M	9579
		10418/M	10418	10568/M	10568	10572/M	10572	10598/M	10598
		10622/M	10622	11251	11457/M	11457			
spd_index	2136	8521/M	8565/M	8619/M	8663/M	9381/M	9480	9486/P	9492
		9508/M	10574/M	11215/M	11338/M	11389/M			
spd_index	9873	9892	9893/S	9894/S	9896/S	9897/S	9898/S	9899/S	
spd_index	9938	9975/M	9991/S	10001/S	10003/P	10014/S	10049/P	10070/P	10085/P
		10154/P	10222/M	10222					
spd_p	9939	10014/M	10015	10021	10038	10041	10044/M	10045	10055
		10056/P	10061	10078	10081	10095/M	10096/M	10097/M	10098
		10112/M	10113/M	10114/P	10114/P	10114/P	10124/M	10125	10129
		10130	10131	10132	10139	10140/P	10145	10146/P	10146/P
		10168							
specified	3879	9153/M	9453/M						
st	5840	9698	10344	10345	10347/S	10354/M	10354	10358/M	10360/M
		10374/M	10375/M	10378/M	10382/M				
starting_spd_index	9862	9885/S	9886/S	9892					
statistics	1389	9185	9333	10901	10904	10906	10907	10943	11338
		11389							
statistics	7709	9388/M	9389	9391/M	9392	11242/M	11243	11245/M	11246
		11316/M	11317	11320/M	11321	11338/M	11338	11338/M	11338
		11389/M	11389	11389/M	11389				
status	2833	11290/M	11310/P	11311	11338/P	11355/P	11367/P	11383/P	11389/P
		11401/P	11406/P						
status	7199	7202/M	7203/M						
status	8263	8532/P	8532/P	8532					
status	8267	8300/M	8304	8304	8379/P	8380	8381	8389/M	8440/P
		8441	8441	8446/M	8449/P	8450	8451/M	8459	8460/M
		8499/P	8501	8502	8512/M	8620/P	8622	8637/P	8638
		8639	8647/M	8693/P	8694	8695	8700		
status	8806	8835/M	8835/M	8839/M	8839/M				
status	8808	8817/M	8835/P	8839/P					
status	8853	8878/P	8889/P	8890	8890	8897			
status	8985	9036/M	9036/M	9086/M	9086/M				
status	8988	9008/M	9036/P	9064/P	9065	9080/P	9081	9086/P	
status	9141	9149/P							
status	9196	9222/P	9222/P	9222					
status	9266	9272/P	9279/P	9285					
status	9319	9396/P	9397	9407	9408/M				
status	9439	9462/P	9487/P	9488	9503	9504	9511/M	9516	
status	9567	9579/P	9579/P	9579					
status	9605	9647/P							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
status	9677	9690/P	9691						
status	9843	10150/M	10150/M						
status	9848	9967/M	9968/M	10150/P					
status	10246	10418/P	10418/P	10418					
status	10521	10568/P	10568/P	10568					
status	10590	10593/P	10594						
status	10639	10642/P	10643						
status	11021	11031/P							
status	11096	11113/P	11114						
status	11273	11338/P	11338	11338	11338/M	11389/P	11389	11389	11389/M
status	11273	11355/M	11355/M	11367/M	11367/M	11383/M	11383/M		
status	11533	11590/P							
ste	5827	9698	10344	10345	10358/M	10374/M	10378/M	10382/M	
stt$set_name	7572	7363							
subfunction	2835	11297							
sum_shared	9004	9020/M	9022/M	9022	9024				
sva	2230	9628	9634	9634	9647/P	9885	9894	9897/M	10001
		10114/P	10139	10140/P	10146/P	10168	10987		
sva	5681	5686							
swap_count	93	8329/M	8329	11237/M	11237				
swap_data	1382	8328/M	8328/M	8328	8331	8378	8385/M	8394	8396
		8423	8425	8430	8431/M	8431/M	8431	8448/P	8453
		8476/M	8483	8489	8496	8497	8498/P	8504/M	8520
		8522	8532/P	8532	8542	8544	8567	8608	8611/M
		8634	8635	8636/P	8659	8660	8692/P	8778	8797/M
		8819	8820	8821	8822/P	8829/M	8871	8872	8913
		8915	8940	8953/M	8970/M	8974/M	8977	9009	9010
		9078/P	9177/M	9178	9180	9182/M	9182/M	9182	9222/P
		9222	9278/P	9294	9380/M	9385/M	9390	9390	9393
		9468	9509/M	9547/M	9574	9576	9579/P	9579	10294
		10418/P	10418	10423/P	10568/P	10568	10573	10599	10602
		10618	10623	10628/P	10648	10652	10658	11129	11225/M
		11236/M	11236/M	11236	11239	11244	11244	11247	11338/M
		11338/M	11338	11338	11338	11389/M	11389/M	11389	11389
		11389	11396/M	11444/M	11444	11448/M	11449	11455/M	11456
		11459/M	11460						
swap_file_descriptor_page_count	1977	8448/P	8453	8497	8532/P	8532	8635	8820	8871
		8953/M	9010	9222/P	9222	9279/P	9294	9579/P	9579
		10418/P	10418	10568/P	10568				
swap_file_descriptor_pfti	2140	8907/M	10214	10214					
swap_file_length_in_pages	1422	8821	8829/M						
swap_file_sfid	1419	8498/P	8636/P	8822/P					
swap_io_control	1383	8499/P	8521/M	8565/M	8619/M	8637/P	8663/M	8907/M	9381/M
		9508/M	10214	10214	10574/M	11215/M	11338/M	11389/M	
swap_queue_link	1364	8296	8595	9757	9770	9771/P	9772/M	9772	9774
		9777	9778/P	9779/M	9779	9781	9803/M	9804/M	9806/M
		9810/M	9827/M	11204	11551				
swap_reason	9317	9328	9330	9361					
swap_reason	11273	11338	11338	11338	11389	11389	11389		
swap_resident_ijle_p	10799	10801/P	10805						
swap_resident_ijlo	10798	10801/P	10807/P	10809/P					

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
swap_stats	7716	9388/M	9390	9391/M	9392	11242/M	11243	11245/M	11246
		11316/M	11317	11320/M	11322	11338/M	11338	11338/M	11338
		11389/M	11389	11389/M	11389				
swap_status	1358	8302	8305	8718	8775	8798/M	9226	9347	9349
		9351	9365	10533	10596	10597	10616	10620	10621
		10645	10650	10654	10655	10833	10860	11103	11338
		11338	11338	11338	11389	11389	11389	11389	11397
		11446	11447	11453	11454	11458	11463	11553	11560
		11584							
swap_status	10531	10533/M	10535	10547	10548	10548	10549	10551	10565
		10565							
swap_wait_time	7735	11316/M	11318						
swapin_after_io	10633	8485	8568	10576	10678				
swapin_before_io	10584	8338	8362	10556	10630				
swapin_q_priority_timestamp	1413	11319							
swapout_io_cannot_initiate	7893	8481/M	8482	9575/M	9576	10646/M	10647	11461/M	11462
swapout_io_not_initiated	7892	8328/M	8328	8392/M	8393	8431/M	8431	8976/M	8977
		9179/M	9180	9182/M	9182	10600/M	10601	10617/M	10618
		10656/M	10657	11128/M	11129	11236/M	11236	11450/M	11451
swapout_reason	1410	9361/M	9396/P	11338/M	11338/P	11363/M	11389/M	11389/P	
swapout_reason	2837	11337/P	11359	11366	11372	11388/P			
swapout_timestamp	1425	9385/M	9390	11225/M	11244	11338/M	11338	11389/M	11389
swapped_job_entry	1281	8913/M	9468/M						
swapped_job_entry	1427	8328/M	8423	8425	8431/M	8448/P	8453	8497	8532/P
		8532	8608	8635	8692/P	8820	8871	8913	8915
		8953/M	8970/M	9010	9078/P	9177/M	9178	9182/M	9222/P
		9222	9278/P	9294	9468	9547/M	9579/P	9579	10418/P
		10418	10423/P	10568/P	10568	10628/P	11236/M	11448/M	11449
		11455/M	11456	11459/M	11460				
swapped_job_entry	2151	8915/M							
swapped_job_page_count	1421	8328/M	8328	8331	8394	8396	8430	8431/M	8431
		8483	8489	8496	8542	8544	8567	8611/M	8634
		8819	8872	8940	8974/M	8977	9009	9180	9182/M
		9182	9380/M	9393	9574	9576	10573	10599	10602
		10618	10623	10648	10652	10658	11129	11236/M	11236
		11239	11247	11338/M	11338	11389/M	11389	11444/M	11444
swapped_page_descriptors	2152	9885	9886	9892	9893	9894	9896/M	9897/M	9898/M
		9899/M	9975	9991	10001	10014			
swapped_pages	7733	9391/M	9393	11245/M	11247	11338/M	11338	11389/M	11389
swapping_io_error	1420	8378	8385/M	8504/M	8520	8522	8659	8660	9509/M
		11396/M							
swapping_keypoints	3868	9070	9367	10610	10668	10982	11122	11212	11338
		11389							
swapping_stack_trace	3870	8324	9117	10478	10537	11160			
syc$mf_cause_job_recovery	6611	9690/P							
syc$mf_for_keypoint_traceback	6614	8324/P	9117/P	10479/P	10537/P	11161/P			
syc$ucr_condition	6703	6714							
syc$user_defined_condition	6704	6716							
system_breakpoint_selected	1385	10403/M							
system_ijle_p	9679	9684/P							
system_job_monitor_sdt_p	10268	10275/P	10354						

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
system_job_monitor_sdtx_p	9661	9706							
system_job_monitor_sdtx_p	10269	10276/P	10285/P	10352					
system_supplied_name	1353	9071	9954	9954	9959	10611	10669	10751	10756
		10761	11123						
syt$180_idle_code	7144	7025	7932	7934					
syt$monitor_flag	6610	6595	7253	7288					
syt$monitor_flags	6595	6140							
syt$monitor_request_code	2909	2832	4648						
syt$monitor_status	2897	2833	4649	5317	5475	5485	5500	5748	5781
		7199	7247	7289	8267	8808	8853	8988	9141
		9266	9319	9439	9605	9677	9848	10590	10639
		11021	11096	11533					
syt$perf_keypoints_enabled	3865	3861							
syt$test_jr_set	36	8223	8223	8223					
syv$perf_keypoints_enabled	3861	8324	9070	9117	9367	10478	10537	10610	10668
		10982	11122	11160	11212	11338	11389		
tail	931	10127/M							
task_created_after_last_swap	1398	10931	10941/M						
task_queue	2226	10126/M	10127/M						
temp_asti	9000	9050/P							
template_asids_changed	10270	10298/M	10299	10315					
terminate_access_work	1388	9403/M	10748	10759	10766	10774/P	10779/M	10840/M	11221/M
		11338/M	11389/M	11495/M	11495	11502			
terminate_access_work	1547	11496							
time	5758	5764							
time	8263	8472							
time	9843	9947							
time	10880	10957							
time_spent_in_job_mode	1805	10904	10906						
time_spent_in_mtr_mode	1806	10907							
timestamp	1424	8778	8797/M	9390	11244	11338	11389		
timestamp	10271	10294/M	10295	10298					
timestamp	11054	9359/M	11069/M	11104/M	11166/M	11201/M	11299/M	11338/M	11389/M
tmc$	3740	3746	3749	3752	3755	3758	3761	3764	3767
		3770	3773	3776	3779	3782	3785	3788	3791
tmc$broken_task_fault_id	6631	6681							
tmc$btc_invalid_a0	6729	6750							
tmc$btc_invalid_p	6729	6750							
tmc$btc_mcr_traps_disabled	6730	6751							
tmc$btc_mf_traps_disabled	6729	6749							
tmc$btc_mntr_fault_buffer_full	6728	6749							
tmc$btc_system_error	6731	6745							
tmc$btc_ucr_traps_disabled	6730	6751							
tmc$dummy_fault	6632	6687							
tmc$flag_available_31	6866	6870							
tmc$fnx_continue	5286	9724/P	10391/P	10407/P	10704/P				
tmc$fnx_swapping_job	5286	8324/P	9117/P	9686/P	10324/P	10404/P	10468/P	10537/P	10698/P
		10773/P							
tmc$maximum_monitor_faults	6636	6627							
tmc$maximum_signals	6846	6843							

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
tmc$maximum_system_task_id	6879	6882							
tmc$mcr_fault	6631	6683							
tmc$signal_available_63	6828	6839							
tmc$stid_null_task	6885	6882							
tmc$ts_io_wait_not_queued	3851	3833							
tmc$ts_page_wait	3853	3834							
tmc$ts_ready	3838	3829							
tmc$ts_timed_wait_not_queued	3845	3832							
tmc$ts_timeout_reqexp_longvlong	3843	3831							
tmc$ts_timeout_reqexp_shortshrt	3842	3830							
tmp$clear_lock	7211	7224	9394	9762	9829	10811	11248	11309	11323
		11338	11341	11389	11393				
tmp$find_next_xcb	7235	9686	9724	10324	10391	10404	10407	10698	10704
tmp$idle_tasks_in_job	7244	9396	11338	11389					
tmp$monitor_flag_job_tasks	7252	8324	9117	10479	10537	11161			
tmp$restart_idled_tasks	7259	8324	9117	10486	10537	11164			
tmp$set_lock	7264	7282	9387	9754	10803	11241	11307	11315	11333
		11338	11352	11389					
tmp$set_monitor_flag	7287	9690							
tmp$set_up_debug_registers	7292	10406							
tmp$update_job_task_environment	7298	8324	9117	10468	10537				
tmt$broken_task_condition	6728	6744							
tmt$broken_task_monitor_fault	6742	6682							
tmt$dispatch_control	7174	7020							
tmt$dual_state_priority_entry	7189	7007	7186						
tmt$find_next_xcb_state	5288	7238	9681	10273	10695				
tmt$fnx_search_type	5286	5277	5289	7235	7300				
tmt$mcr_faults	6767	6684							
tmt$monitor_fault_buffer	6621	6177							
tmt$monitor_fault_buffers	6627	6622	6623	6624					
tmt$monitor_fault_identifiers	6630	6680	6756						
tmt$ptl_lock	3798	7211	7264	7750	8058	8202	8202		
tmt$signal	6784	6779							
tmt$signal_buffer	6776	6178							
tmt$signal_buffers	6843	6777	6778	6779					
tmt$system_flags	6849	6153							
tmt$system_task_id	6882	6144							
tmt$task_queue_link	929	922	2226						
tmt$task_status	3838	7177							
tmv$ptl_lock	8058	10803/P	10811/P	11307/P	11309/P	11333/P	11341/P	11352/P	11393/P
tmv$swapin_in_progress	8063	8650/M	8650	8658/M	8658				
total_pages_per_segment	3033	10972/M	10973	10976/M					
total_pages_removed	10896	10926/P	10928/P	10994					
total_swapped_page_count	8293	8496/M	8499/P	8634/M	8637/P				
total_swapped_page_count	8815	8819/M	8821	8823/P	8829	8830			
total_swapped_page_count	8868	8872/M	8904/P						
total_swapped_page_count	9005	9009/M	9014	9015	9025	9027/P	9032	9033	
total_swapped_page_count	9437	9480	9492						
total_swaps	3034	10978/M	10978						
total_time	3026	8784/M	8784						
trace	8252	8257	8315	8322	8324	8337	8349	8428	8447

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
		8492	8503	8523	8526	8555	8597	8640	8661
		8665	8674	8726	8825	8834	8838	9026	9035
		9049	9053	9055	9066	9082	9117	9161	9168
		9186	9223	9232	9239	9246	9366	9489	9538
		9539	9616	9626	9636	9646	9888	9890	9895
		10025	10039	10042	10051	10059	10066	10079	10093
		10123	10142	10148	10153	10159	10184	10190	10192
		10201	10284	10300	10303	10306	10328	10338	10355
		10361	10376	10379	10383	10460	10536	10537	10550
		10553	10566	10595	10644	10806	10910	10913	10928
		10929	10944	11105	11115	11155	11167	11205	11300
		11312	11338	11389	11399	11404	11442	11452	11466
		11568							
trace_index	8253	8256/S	8256/S						
trace_index	8263	8315/S	8315/S	8322/S	8322/S	8324/S	8324/S	8337/S	8337/S
		8349/S	8349/S	8428/S	8428/S	8447/S	8447/S	8492/S	8492/S
		8503/S	8503/S	8523/S	8523/S	8526/S	8526/S	8555/S	8555/S
		8597/S	8597/S	8640/S	8640/S	8661/S	8661/S	8665/S	8665/S
		8674/S	8674/S	8726/S	8726/S				
trace_index	8806	8825/S	8825/S	8834/S	8834/S	8838/S	8838/S		
trace_index	8985	9026/S	9026/S	9035/S	9035/S	9049/S	9049/S	9053/S	9053/S
		9055/S	9055/S	9066/S	9066/S	9082/S	9082/S		
trace_index	9093	9117/S	9117/S						
trace_index	9126	9161/S	9161/S	9168/S	9168/S	9186/S	9186/S		
trace_index	9196	9223/S	9223/S	9232/S	9232/S	9239/S	9239/S	9246/S	9246/S
trace_index	9314	9366/S	9366/S						
trace_index	9432	9489/S	9489/S						
trace_index	9524	9538/S	9538/S	9539/S	9539/S				
trace_index	9594	9616/S	9616/S	9626/S	9626/S	9636/S	9636/S	9646/S	9646/S
trace_index	9843	10025/S	10025/S	10039/S	10039/S	10042/S	10042/S	10051/S	10051/S
		10059/S	10059/S	10066/S	10066/S	10079/S	10079/S	10093/S	10093/S
		10123/S	10123/S	10142/S	10142/S	10148/S	10148/S	10153/S	10153/S
		10159/S	10159/S	10184/S	10184/S	10190/S	10190/S	10192/S	10192/S
		10201/S	10201/S						
trace_index	9861	9888/S	9888/S	9890/S	9890/S	9895/S	9895/S		
trace_index	10246	10284/S	10284/S	10300/S	10300/S	10303/S	10303/S	10306/S	10306/S
		10328/S	10328/S	10338/S	10338/S	10355/S	10355/S	10361/S	10361/S
		10376/S	10376/S	10379/S	10379/S	10383/S	10383/S		
trace_index	10429	10460/S	10460/S						
trace_index	10521	10536/S	10536/S	10537/S	10537/S	10550/S	10550/S	10553/S	10553/S
		10566/S	10566/S						
trace_index	10584	10595/S	10595/S						
trace_index	10633	10644/S	10644/S						
trace_index	10797	10806/S	10806/S						
trace_index	10880	10910/S	10910/S	10913/S	10913/S	10928/S	10928/S	10929/S	10929/S
		10944/S	10944/S						
trace_index	11076	11105/S	11105/S	11115/S	11115/S	11155/S	11155/S	11167/S	11167/S
trace_index	11179	11205/S	11205/S						
trace_index	11273	11300/S	11300/S	11312/S	11312/S	11338/S	11338/S	11389/S	11389/S
		11399/S	11399/S	11404/S	11404/S				
trace_index	11435	11442/S	11442/S	11452/S	11452/S	11466/S	11466/S		

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter

NOS/VE js : monitor mode job swapper
JSP$SWAP_POLLING

IDENTIFIER	DEFINED ON LINE	REFERENCES							
trace_index	11515	11568/S	11568/S						
try	8869	8884/M	8890	8896/M	8896				
tu_pfte_p	9606	9633/M	9634	9634	9635	9637/S			
tu_pfti	9607	9630/M	9632	9632	9633/S	9639/M	9639/S		
u	2069	10130/M	10130						
update_processor_selections	10689	10397	10707						
update_segnum_sfd_p	8263	8532/M	8532/M	8532					
update_segnum_sfd_p	9196	9222/M	9222/M	9222					
update_segnum_sfd_p	9267	9273/M	9274/M	9275					
update_segnum_sfd_p	9450	9463/M	9464/M	9465	9467				
update_segnum_sfd_p	9567	9579/M	9579/M	9579					
update_segnum_sfd_p	10246	10418/M	10418/M	10418					
update_segnum_sfd_p	10521	10568/M	10568/M	10568					
update_server_files	10710	10412	10780						
v	2067	9541	9541	10132/M	10132				
v1	5854	9698	10344						
write_status	9142	9160/P							
xcb_p	5800	5806	5806	5807	5807				
xcb_p	9658	9696	9696	9696	9696				
xcb_p	9680	9686/P	9688	9688	9690/P	9695/M	9696/P	9697	9722/M
		9724/P							
xcb_p	10246	10276	10276	10276	10276	10329	10329	10329	10329
xcb_p	10272	10324/P	10327	10327	10329/P	10333/M	10334/M	10341	10391/P
		10404/P	10405	10405	10406/P	10406/P	10407/P		
xcb_p	10694	10698/P	10700	10700	10701	10702/M	10704/P		
xcb_state	9681	9686/P	9724/P						
xcb_state	10273	10324/P	10391/P	10404/P	10407/P				
xcb_state	10695	10698/P	10704/P						
xp	6139	9697	10333/M	10334/M	10341				

*** REFERENCE ABBREVIATIONS : M=modify, A=attribute, S=subscript, I=I/O ref, R=read, W=write, P=parameter