
Sirius Performance Enhancements Reference Manual



Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, MA 02139

Telephone: (617) 876-6677
FAX: (617) 234-1200
E-mail: support@sirius-software.com
World Wide Web: <http://sirius-software.com>

June 17, 2005

© 2005 Sirius Software, Inc.

Proprietary Notices

Sirius Mods is a package of one or more proprietary products of Sirius Software, Inc.:

Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, Massachusetts 02139
USA

The following products are also proprietary products of Sirius Software, Inc.:

- *Sirius Performance Enhancements* (part of the *Sirius Mods*)
- *Sirius Performance Enhancements V2* (part of the *Sirius Mods*)

Model 204™ is a proprietary product of Computer Corporation of America:

Computer Corporation of America
500 Old Connecticut Path
Framingham, Massachusetts 01701
USA

Model 204™ is a registered trademark of Computer Corporation of America.

Contents

Proprietary Notices	ii
Contents	iii
Summary of Changes	v
Sirius Mods Version 6.7	v
Sirius Mods Version 6.6	v
Sirius Mods Version 6.1	v
Sirius Mods Version 5.6	vi
Chapter 1: Overview	1
Chapter 2: System Requirements	3
Chapter 3: The APSYXMP System Parameter	5
Chapter 4: The GTBLHASH and GTBLPCT System Parameters	7
Chapter 5: The MAXINCL System Parameter	9
Chapter 6: The MPSPINC System Parameter	11
Chapter 7: The PERFOPT System Parameter	13
Chapter 8: The PERFOPT2 System Parameter	23
Chapter 9: The RESLTHR System Parameter	25
Chapter 10: The SIRAPSY System Command	27
 Figures		
Figure 1: SIRAPSY command syntax	27

Summary of Changes

This section describes significant changes to the documentation. In most cases these changes correspond to changes in the installation process which occur with the release of some version of the *Sirius Mods*.

Note that this manual applies to all generally supported versions of the *Sirius Mods*, whether or not there is mention of a change for that version in this section.

Sirius Mods Version 6.7

The User Language Macro Facility documentation, formerly included in this manual, is removed. It is henceforth located only in the ***Janus SOAP Reference Manual***.

Sirius Mods Version 6.6

The X'00000010' bit setting of the PERFOPT2 parameter ([“The PERFOPT2 System Parameter” on page 23](#)) is no longer used, as the User Language Macro Facility becomes automatically available to all Sirius API products.

Sirius Mods Version 6.1

The *Sirius Performance Enhancements V3* are added to ***Sirius Performance Enhancements Reference Manual***.

Sirius Performance Enhancements V3 is a collection of performance enhancements for *Model 204* Version 4.2 that are the result of a joint development effort between Sirius and CCA. Although packaged as a Sirius product for use with *Model 204* Version 4.2, the entire set of enhancements has been integrated with *Model 204* Version 5.1.

The enhancements, each of which is separately activated by a parameter setting, include the following:

- Reduced cost of searching GTBL (GTBLHASH>1)
See [“The GTBLHASH and GTBLPCT System Parameters” on page 7](#).
- Subscripted field references (PERFOPT X'01000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).

- String arguments to dollar functions (PERFOPT X'02000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).
- IDENTIFY IMAGE statement (PERFOPT X'04000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).
- Dollar functions that return strings (PERFOPT X'08000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).
- WITH clause, assignment and comparison statements (PERFOPT X'10000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).
- Field extraction with unlocked records (PERFOPT X'20000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).
- Overhead of idle non-XDM CRAM threads (PERFOPT X'40000000' bit)
See [“The PERFOPT System Parameter” on page 13](#).

Sirius Mods Version 5.6

- The *Sirius Performance Enhancements V2* is enhanced to include the User Language Macro Facility, which requires the specification of the new X'00000010' bit setting of the PERFOPT2 parameter (see [“The PERFOPT2 System Parameter” on page 23](#)).
- Add documentation for the new *Sirius Performance Enhancements V2* parameter, MAXINCL.
- Move documentation for *Sirius Performance Enhancements* and *Sirius Performance Enhancements V2* to ***Sirius Performance Enhancements Reference Manual***.

The *Sirius Performance Enhancements* are a collection of enhancements to *Model 204* that expand functionality and improve online performance, especially on large systems. These enhancements consist of three separately packaged products (the *Sirius Performance Enhancements*, *Sirius Performance Enhancements V2*, and *Sirius Performance Enhancements V3*) that are not included with the *Sirius Mods* unless they have been specifically ordered. They are almost entirely included in later versions of *Model 204*.

The *Sirius Mods* include many products such as *Fast/Backup*, *Fast/Reload*, *Janus Web Server* and *Janus Specialty Data Store*. No *Sirius Mods* products are required to run *Sirius Performance Enhancements*, *Sirius Performance Enhancements V2*, or *Sirius Performance Enhancements V3* other than themselves.

To install *Sirius Performance Enhancements*, *Sirius Performance Enhancements V2*, or *Sirius Performance Enhancements V3*, the *Sirius Mods* must be installed. When the *Sirius Mods* are installed, all other products owned by the installing site that are part of the *Sirius Mods* will also be installed.

The version number of the *Sirius Performance Enhancements*, *Sirius Performance Enhancements V2*, and *Sirius Performance Enhancements V3* products is generally considered to be the version of the *Sirius Mods* in which they are contained:

- The *Sirius Performance Enhancements* was first available in version 4.4 of the *Sirius Mods* so the first version of *Sirius Performance Enhancements* was actually called version 4.4.
- The *Sirius Performance Enhancements V2* was first available in version 5.3 of the *Sirius Mods* so the first version of *Sirius Performance Enhancements V2* was actually called version 5.3.
- The *Sirius Performance Enhancements V3* was first available in version 6.1 of the *Sirius Mods* so the first version of *Sirius Performance Enhancements V3* was actually called version 6.1.

The ***Sirius Performance Enhancements Reference Manual*** assumes that a site is running *Sirius Mods* version 5.4 or later. Any documentation that requires a later version of the *Sirius Mods* will be clearly marked to indicate this. For example, a parameter that is only available in versions 5.6 and later of the *Sirius Mods* will have a sentence such as “This parameter is only available in version 5.6 or later of the *Sirius Mods*” in its documentation. If a feature, \$function, command, or parameter is not indicated as requiring any specific version of the *Sirius Mods*, it can be assumed that it is available as documented in all recent versions of the *Sirius Performance Enhancements*, *Sirius*

Performance Enhancements V2, and *Sirius Performance Enhancements V3*; that is, all versions since version 6.2 of the *Sirius Mods*

Most of the features available in the original *Sirius Performance Enhancements* have been incorporated into *Model 204 V4.1* and later.

There are several system parameters and commands that control the optimizations provided by the performance enhancement products. These parameters and commands are:

- “The APSYXMP System Parameter”
- “The GTBLHASH and GTBLPCT System Parameters”
- “The MAXINCL System Parameter”
- “The MPSPINC System Parameter”
- “The PERFOPT System Parameter”
- “The PERFOPT2 System Parameter”
- “The RESLTHR System Parameter”
- “The SIRAPSY System Command”

System Requirements

The *Sirius Performance Enhancements* and *Sirius Performance Enhancements V2* require the following components:

- Mainframe operating systems:
 - MVS/SP Version 1.3 or later (or MVS/XA, MVS/ESA, OS/390, or z/OS) or
 - VSE Version 2 or later or
 - CMS Release 5 or later running under
 - VM/SP Release 5 or later or
 - VM/XA or
 - VM/ESA or
 - z/VM
- *Model 204* Version 4 Release 2.0 or later.

The *Sirius Performance Enhancements V3* require *Model 204* Version 4 Release 2.0 or later, and they require the mainframe components suitable for that version.

The APSYXMP System Parameter

With *Sirius Performance Enhancements V2* it is possible to have saved APSY compilations saved to expanded storage as well as to CCATEMP. If an APSY load is required for a saved compilation, *Model 204* attempts to do the load from expanded storage rather than from CCATEMP pages that are loaded into the buffer pool. This has several benefits:

- The instruction to move data from expanded storage (MVPG) is much more efficient than the instructions for moving data from buffer pool pages (MVCL or MVC).
- It eliminates logical I/O to the CCATEMP pages in the buffer pool, reducing CPU path length and buffer pool MP lock utilization in an MP environment.
- It eliminates physical I/O for the CCATEMP pages containing saved compilations.
- It reduces the number of buffer pool pages used to hold the CCATEMP pages containing saved compilations. This makes the pages available for other uses, reducing overall disk I/O even for non-CCATEMP files.

The APSYXMP parameter indicates the maximum number of expanded storage pages that can be used to hold saved APSY compilations, and it can have a value between 0 and 4194304. A value of 0 indicates that no expanded storage will be used to hold saved compilations. The pages referred to by APSYXMP are 4096 byte System 390 pages not *Model 204*'s 6184 byte pages.

If more saved compilations exist than can be fit into APSYXMP pages (or the number of pages OS/390 makes available), compilations are discarded from expanded storage on an LRU basis and must subsequently be loaded from CCATEMP. This means that as long as APSYXMP is set high enough to hold a “reasonable” subset of saved compilations, many of its benefits should still apply since the most heavily used compilations would tend to stay in expanded storage. APSYXMP should not be set to a value greater than the expected number of available expanded storage page frames on a system, since this will produce a continuous stream of unnecessary page frame allocation errors that would be bad for performance.

APSYXMP can be used to advantage even on systems without expanded storage as long as there is a large supply of available main storage page frames. The *Sirius Performance Enhancements V2* will use main storage page frames instead of expanded storage page frames to hold saved compilations if expanded storage is not available.

Setting APSYXMP to a non-zero value forces all servers to be page-aligned in main storage, which can result in a slight increase of up to 4088 bytes per server in main storage requirements. A non-zero APSYXMP also forces the page alignment of certain

Model 204 server tables (VTBL, STBL, QTBL and NTBL), which can cause an increase in server size of, in worst case, slightly less than 16K, though more likely around 8K. This increase can be for both the main storage space required for servers and the disk space requirements for servers. It can also increase the quantity of data being moved for each server swap by up to 16K for sites not using resident QTBL and 8K for sites getting good use out of resident QTBL though the more likely effect is around 8K and 4K respectively.

Even if APSYXMP is set to a non-zero value, all pre-compiled APSY procedures are still saved to CCATEMP. This means setting APSYXMP non-zero will not decrease the number of CCATEMP writes, only the number reads.

APSYXMP can be set either in the PARM value on the EXEC card or as a user0 parameter.

There are several system statistics that are available to facilitate monitoring the efficacy of the APSY load from expanded storage feature. To avoid changing the system statistics block format, this feature reuses (or misuses) some obsolete system statistics. These statistics and their meanings are:

- OUTVMIO** The number of times a save of an APSY compilation to expanded storage failed because there were not sufficient expanded storage pages available. A non-zero value suggests over-allocation of expanded storage pages.
- PFLI** The number of (4K byte) pages read from expanded storage for APSY loads.
- PFLO** The number of requests discarded from expanded storage to make space for new ones. A non-zero value suggests that there might be a benefit in increasing APSYXMP.
- SRBT** The total APSY load count.
- SRVC** The APSY load count from expanded storage. If this is not a high percentage of SRBT, something is seriously wrong.
- TCON** The high water mark of pages used to hold APSY requests in expanded storage. If this value is close to APSYXMP, it might be a good idea to increase APSYXMP.
- TQWT** The number of times an APSY load from expanded storage failed because MVS had stolen the expanded storage page from *Model 204*. A non-zero value suggests over-allocation of expanded storage pages.

These statistics are available both as system final statistics and system partial statistics, and they can also be monitored in *SirMon*.

The GTBLHASH and GTBLPCT System Parameters

The *Model 204* global variable table consists of a single table that is scanned sequentially to locate a given variable. As the number of global variables increases, the scanning costs increase. It is not uncommon for up to 5% of the CPU consumed by an ONLINE to be spent scanning GTBL.

The GTBLHASH parameter causes that portion of GTBL used for storing global variables to be divided into GTBLHASH sections of equal size. A second system parameter, GTBLPCT, indicates the initial percentage of GTBL to be allocated to the portion of GTBL used for global variables and subject to control by GTBLHASH.

The default for GTBLHASH is 1, which deactivates the optimization. The default for GTBLPCT is 50 (for 50%).

In order to look up a name when $GTBLHASH > 1$, a hash function is used to determine the correct section, then that section is sequentially scanned. This approach reduces the average number of GTBL entries that must be scanned by a factor of GTBLHASH. Also, the overhead of updating a value is reduced, because only the entries in a bucket need to be moved to fill in holes made by deleted or updated values. Under TPNS testing, a large customer saw an overall CPU saving of 4.3% with GTBLHASH set to 23.

If you are on the verge of filling GTBL and are unlucky enough that some of the buckets get more full than the others, we might need to rearrange the buckets so all the globals fit (we do this automatically on the fly). This can be a fairly expensive operation, and currently we keep no statistics on how many times this happens. The lesson here is not to run GTBL "close to the edge" anyway, since you are likely to get an occasional \$SETG failure.

Similarly, if GTBLPCT is incorrect, there will be some overhead dynamically fixing things to reflect the actual usage. This fixup is done either when a GTBLHASH section fills or when the portion of GTBL not used for global variables would expand into the last GTBLHASH section. Again, no statistics keep track of this, and no statistics are available to help one set the correct value of GTBLPCT. However, the "fixup" code has not appeared in any SirTune reports, so its overhead seems not too dramatic.

The MAXINCL System Parameter

With MAXINCL parameter is available with the *Sirius Performance Enhancements V2* with *Sirius Mods* version 5.6 and later. MAXINCL indicates the maximum INCLUDE nesting level to be allowed by *Model 204*. The default value of MAXINCL is 5, which is the maximum INCLUDE nesting level normally allowed by *Model 204*. By setting MAXINCL to a value greater than 5, it is possible to get around the *Model 204* INCLUDE nesting limit.

MAXINCL must be given a value between 5 and 40. Each increment of MAXINCL adds 61+LAUDPROC (the value of the LAUDPROC system parameter) bytes to the fixed server requirements for every user under *Model 204 V4.2* and earlier, and 65+LAUDPROC bytes under *Model 204 V4.2.1* and later. So for example, if LAUDPROC is 60 and MAXINCL is set to 20, the extra fixed server area requirement beyond what would be required if MAXINCL were not set would be $(65+60) \times (20-5)$, or 1875 bytes under *Model 204 V4.2.1*.

MAXINCL can be set either in the PARM value on the EXEC card or as a user0 parameter.

The MPSPINC System Parameter

With MPSPINC parameter is available with the *Sirius Performance Enhancements*, but it has been obviated by the MAXSPINS parameter under *Model 204* version 4.1 and later. MPSPINC would indicate the number of times *Model 204* should retry obtaining an MP lock before issuing an operating system wait on the lock. MPSPINC is ignored unless the PERFOPT X'00000008' bit is set.

MPSPINC can be set either in the PARM value on the EXEC card or as a user0 parameter.

The PERFOPT System Parameter

The PERFOPT parameter is a collection of 32 bits. Each bit in PERFOPT controls whether a particular enhancement will be in effect for a run. Some bits in PERFOPT are associated with the *Sirius Performance Enhancements*, others with the *Sirius Performance Enhancements V2*, and others with the *Sirius Performance Enhancements V3*. PERFOPT can only be set on the EXEC card parameters or in the user0 parameters.

The default for PERFOPT is X'03FFFFFF', which means all performance enhancements will be used if appropriate. Any performance enhancements that are not appropriate to a site for whatever reason have their corresponding bits turned off in PERFOPT during initialization, so that one can determine which optimizations are actually in effect by entering `V PERFOPT` after the Online has come up.

The PERFOPT bits are:

X'00000001' If the Online is running on an MVS system and the Online is running APF authorized, the Timer PC enhancement will be used to speed up CPU accounting. This results in CPU savings over the entire Online, especially in systems with heavy scheduler traffic. This will usually be the case in any MP/204 Online.

This enhancement is obviated by the availability of the MP timer PC in *Model 204* version 4.1 and later.

X'00000002' If the Online is running on an MVS system and the Online is using Fast Cram V2.7 or later, the Fast Cram Invisible Waiters enhancement will be used to eliminate the scheduler overhead associated with the idle Cram threads. The more IODEV 11's or IODEV 29's an Online has, the more significant the overhead for these threads will be, and hence the more significant the savings from the Fast Cram Invisible Waiter enhancement. When this enhancement is in effect, there will be an extra PST in the Online called FCPST so the setting of NSUBTKS might need to be increased by one before using the Fast Cram Invisible Waiter enhancement.

X'00000004' The Timer Queue enhancement will be used. This enhancement reduces the overhead of scanning long timer queues for starting and stopping long timed waits. This tends to be an issue in any Online that has a large number of terminal threads with timeout periods. When this enhancement is in effect, there will be an extra PST in the Online called TIMERQ, so the setting of NSUBTKS might need to be increased by one before using the Timer Queue enhancement.

This enhancement works by taking all waits with time limits greater than TIMEQMIN, a user0 parameter with a default of 60 (seconds), and monitoring them periodically with the TIMERQ PST rather than placing them on a timer queue.

The PST will not wake up more frequently than indicated by TIMEQRES, a user0 parameter with a default of 5 (seconds). A high value of TIMEQMIN will tend to keep the CPU used by the TIMERQ PST low, but it also reduces some of the benefits of the timer queue length reduction. A low value of TIMEQMIN might increase the amount of CPU used by the TIMERQ PST, while increasing the benefits of the timer queue length reduction. A high value of TIMEQRES reduces the accuracy of timed waits (a 60 second wait might take 65 seconds), but it reduces the amount of CPU used by the TIMERQ PST. A low value of TIMEQRES increases the accuracy of timed waits, but it might increase the amount of CPU used by the TIMERQ PST. The defaults for TIMEQMIN and TIMEQRES should be adequate for most situations.

This enhancement is obviated by timer queue handling enhancements in *Model 204* version 4.1 and later.

X'00000008' If the Online is using the MP/204 feature for *Model 204* V2.2 or V3.1, the MP Spin Lock enhancement will be used. When an MP lock conflict is encountered in *Model 204*, this enhancement will “spin,” attempting to obtain the lock rather than immediately placing the task into a wait state. Since the overhead of entering a wait state is great and MP locks tend to be held for fairly short periods, “spinning” on a lock can often result in great CPU savings.

The number of times to try to get a lock before giving up is indicated by MPSPINC, a system manager resettable parameter with a default of 200. Increasing MPSPINC might reduce the number of MP lock waits but increase the amount of CPU spent spinning on locks. This would generally be an acceptable tradeoff to make in situations where there is almost always an idle processor in the complex. Reducing MPSPINC might increase the number of MP lock waits but reduce the amount of CPU spent spinning on locks. This might be reasonable where the processors in a complex tend to be overcommitted.

This enhancement is obviated by the MAXSPINS parameter in *Model 204* version 4.1 and later.

X'00000010' Speeds up the scanning of VTBL to locate CCATEMP pages associated with a request that must be released.

This enhancement is obviated by the performance improvements in *Model 204* version 3.2 and later.

X'0000020' Eliminates IOS branch entry feature's use of the local lock for updating I/O counters to reduce local lock conflicts, and makes all disk and server I/O ECBs "internal" to eliminate the cost of placing them on the wait ECB list when the *Model 204* maintask goes into a operating system wait.

This enhancement is obviated by the identical performance improvements in in *Model 204* version 4.1 and later.

X'0000040' Makes the most common Horizon waits "invisible" so that threads in Horizon waits do not have to be scanned by the *Model 204* scheduler every time it is entered.

This enhancement is obviated by the identical performance improvements in in *Model 204* version 4.1 and later.

X'0000080' Makes PAUSE, *SLEEP, and dead thread waits (for example IODEV 3's that have reached the end of the input stream) "invisible," so that threads in these waits do not have to be scanned by the *Model 204* scheduler every time it is entered.

This enhancement is obviated by the identical performance improvements in in *Model 204* version 4.1 and later.

X'00000100' Eliminates the NTBL scan performed for every FIND, PLACE, and REMOVE statement to determine if a found set is a global found set.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00000200' Optimizes the release of bitmap pages in found sets at the start of FIND statements and in RELEASE statements in the common case where relatively few segments in a foundset actually contain records.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00000400' Significantly reduces the length of the NTBL scan performed for screen item name variables for COMMON SCREENS.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00000800' Eliminates the long chain of userid locks, which thus eliminates the cost of scanning this chain at logon time when LOGONENQ is set to a non-zero value.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00001000' Tries to keep table B and transaction backout log pages open as long as possible to reduce logical disk I/O's (DKPR's) to eliminate associated path length and reduce the number of MP locks associated with logical disk I/O's.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00002000' Uses OS/390's SUSPEND/RESUME services instead of its WAIT/POST services for deactivating/activating *Model 204* tasks for MP lock waits and posts or for subtasks going idle and being woken. IBM recommends SUSPEND/RESUME as more efficient services.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00004000' Runs a parallel-mode thread on the maintask immediately after obtaining a critical file resource lock rather than sending it back to a subtask. This eliminates the time a critical file resource is held while the holding thread waits to be run on a subtask, reducing the overall time the critical file resource is held, and possibly reducing the number and length of critical file resource conflicts.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00008000' Makes pending update locks single-record locks rather than bitmap locks, in the case where relatively few records in a file have pending updates. There is a pending update lock associated with every record that has an uncommitted update. While it is much more efficient to represent these locks as bitmaps when many records in a file have pending updates, the opposite is the case when only one or a few records in a file have pending updates. In online systems the latter case is probably much more common.

By making pending update locks single record locks in the common case, buffer pool use for pending lock bitmap pages is reduced and the logical disk I/O's required to test for record locking conflicts are also eliminated. Because pending update locks are exclusive locks, these logical disk I/O's can be quite frequent for busy files. In the case where many records in a file have pending updates, these updates are converted to bitmaps when this optimization is used.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00010000' Eliminates the sequential scan of CCARF during roll-forward recovery to find the journal entry associated with the last checkpoint, presumably

the checkpoint to which files have been rolled back. This optimization does a binary search on CCARF to locate the last checkpoint marker record in the journal. This optimization will work with simple journals, multi-volume journals, concatenated journals, ring journals, and any combination of these, though benefits will probably be minor with ring journals.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00020000' Overlaps roll-back I/O during recovery to speed up the roll-back process. On systems with many updated files, this can speed up the roll-back process by a factor of 5 or more.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00040000' Does a sort of the internal APSY procedure dictionary for locked procedures (those with pre-compiled and non-pre-compiled prefixes) so that subsequent lookups can be done with a binary search rather than a sequential scan. Especially useful in systems with heavily used subsystems with large numbers (>1000) of pre-compiled and non-pre-compiled procedures.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00080000' Does not create an enqueueing chain entry for locked APSY procedures (those with pre-compiled and non-pre-compiled prefixes). This drastically reduces the scan of these chains for most purposes. Threads requiring an exclusive procedure enqueue (to update a procedure) simply scan the APSY internal procedure dictionaries for enqueueing conflicts.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00100000' Releases FILE resource locks held by non-pre-compiled APSY procedures as soon as possible. FILE resource locks tend to be held indefinitely in non-AUTOCOMMIT subsystems which can result in a large number of extremely long FILE resource lock chains. These chains are scanned moderately frequently, so this update eliminates almost all of the overhead of these scans.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'00200000' Eliminates the test in MP lock spin code that checks if the task that holds a lock is dispatchable. This test is too clever by a half in many cases, because the task holding the lock might be about to be dispatched by OS/390. So a task might still be better off “spinning” on the lock rather than waiting, even if all the dispatchability bits are not on yet in the task that holds the MP lock.

This enhancement is only available with *Sirius Performance Enhancements V2*.

X'01000000' Provides subscripted field references with the same type of optimization as in a *Model 204* FEO statement. The FEO behavior is shown in the example below of iterating over all occurrences of a repeating group of four fields:

```
0: FEO FOO
   %FOO = VALUE IN 0
   %BAR = BAR(OCCURRENCE IN 0)
   %SNA = SNA(OCCURRENCE IN 0)
   %FU  = FU(OCCURRENCE IN 0)
   . . . .
```

The FEO is optimized so that, except in certain special cases (unlocked record, updated record), FEO picks up where it left off in the record when searching for the next occurrence. Thus FEO does not have to repeatedly scan from the start of the record for each subsequent occurrence of FOO.

However, extracting each occurrence of the other three fields (BAR, SNA, and FU) requires a scan from the beginning of the record, skipping over the occurrences already retrieved. This exponential behavior can cause field extraction to become a significant amount of the total CPU consumed by a *Model 204* ONLINE (reaching 17% for one large customer, for example).

The PERFOPT=X'01000000' optimization detects at compile time a variable occurrence number reference to a field, and it allocates a special field position variable for that reference. When the field extraction is done for the first occurrence of a field, the optimization saves the position in the record, just as FEO saves its position. On subsequent extractions, the optimization determines if the previously saved position is still usable. The occurrence is usable if these are true:

- The occurrence number being requested is greater than the previous occurrence number retrieved.
- The record is locked (share or exclusive), or it is a sort record, or the PERFOPT=X'20000000' optimization is active.

- The fieldcode has not changed (only an issue for fieldname variables).
- The same iteration of the record's FOR loop is being processed.
- The record has not been updated since the position was saved.

A second part of this optimization reduces the cost in either of these cases:

- The subscript is a variable declared as FIXED.
- The subscript is of the "OCCURRENCE IN" form.

This saving is accomplished by eliminating extra quads and processing required to convert the fixed values to a float format. The result is a CPU reduction as well as a reduction in QTBL usage. One large customer saw a total CPU saving of 3.0% from this optimization.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'02000000' Improves the performance of passing arguments to *Model 204* dollar functions in two important cases:

- simple string percent variables
- null strings or missing arguments

This optimization saved about 2.5% of the overall CPU for one customer.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'04000000' Eliminates a scan of NTBL that was performed whenever an IDENTIFY IMAGE statement was executed. For requests that use large amounts of NTBL, the savings can be significant. For example, the CPU savings was about 0.7% for one large customer.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'08000000' Reduces the instruction pathlength for dollar functions that return string values to string percent variables. This saving is accomplished by eliminating a generated assignment quad. One large site measured a 0.2% reduction in CPU usage and a 5% reduction in QTBL usage.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'1000000' Reduces overhead in WITH clause, assignment, and comparison statements. Optimized versions of the assignment, comparison, and WITH quads reduce overhead in common simple cases (simple percent variable to percent variable movements, string to string comparisons, fixed to fixed assignment, etc.). Also reduces the pathlength for cases where comparisons resulted in a JUMP (for example, an IF clause that evaluates to false). These enhancements saved about 1.3% of the CPU for one large site.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'2000000' Allows *Model 204* to save a position within a record, even when the record is not locked. Without this optimization, the *Model 204* FEO statement and the PERFOPT=X'01000000' subscripted field extraction optimization must always scan from the beginning of a record for the Nth occurrence of a field, if the record is not locked. This is required when the record is not locked because a saved position could be rendered invalid as a result of updates to the record. The frequent use (and perhaps overuse) of FDWOL coupled with extraction of repeating groups causes performance degradation.

This optimization provides a highly efficient means for determining whether the contents of a specific record **may** have been changed while a position was held within the record: When an FEO or subscripted field reference would ordinarily be de-optimized, a quick check is done to see if anyone might have changed the record since we saved our position. If not, the saved position can be used, even though the record is not locked.

The key to the performance of this optimization is that it uses a simple hash table with a mechanism that can return false negatives. That is, we might incorrectly determine that it is not safe to use a saved position, when in fact the affected record has not changed. Our experience has shown that the tradeoff of efficiency and accuracy works very well: one large customer saw a 5.0% reduction in *Model 204* with this enhancement.

When this enhancement is activated, a hash table is constructed during *Model 204* initialization. The number of entries in the table is determined by the UFEOHASH (Unlocked FEO HASH) parameter. The default value for UFEOHASH is 1021, which should be large enough for most sites. Each record in each file will hash to a specific table entry, while each table entry may be associated with many records from many files. A system-wide, doubleword counter is initialized to zero.

Before *Model 204* alters a record in a way that would invalidate saved positions, that thread grabs the "direct" critical file resource in exclusive

mode, and it increments the system-wide update counter. Then the file/record combination is hashed to determine its entry in the UFEHASH table, and the incremented update counter is placed into the entry. This is a very efficient operation, taking less than 15 instructions.

Whenever FEO or the PERFOPT=X'01000000' optimization saves a position for a record, it also saves the current value of the update counter from that record's UFEHASH table entry. Before reusing a position in a record that is not locked, the update counter in the appropriate UFEHASH table entry is compared to the update counter value saved with the position. If the two match, it is safe to depend upon the saved position. Otherwise, either the record being processed, or some other record that hashes to the same UFEHASH table entry, has been changed since the position was saved, and the record must be scanned from the beginning. This code errs on the conservative side (it might refuse to use a saved position that is safe), but the resulting simplicity keeps the cost to an acceptable level.

Note: This approach optimizes **all** FEOs and subscripted field references in a read-only environment, since the hash cells will always contain an update counter of zero.

This enhancement is only available with *Sirius Performance Enhancements V3*.

X'40000000' Reduces the overhead associated with idle IODEV=11 and IODEV=29 (BATCH2) threads, eliminating the performance penalty for over-allocating. This enhancement has no effect for users of the XDM CRAM or Fast/CRAM, since these versions already utilize a different technique for accomplishing the same result.

The enhancement makes use of an OS/390 system service for POST Exits, so the *Model 204* load module must be APF-authorized to activate the savings.

A large *Model 204* ONLINE measured savings of about 0.6% using this enhancement.

This enhancement is only available with *Sirius Performance Enhancements V3*.

The PERFOPT2 System Parameter

The PERFOPT2 parameter is a collection of 32 bits. Each bit in PERFOPT2 controls whether a particular enhancement will be in effect for a run. All bits in PERFOPT2 are associated with the *Sirius Performance Enhancements V2*. PERFOPT2 can only be set on the EXEC card parameters or in the user0 parameters.

The default for PERFOPT2 is X'00000000': none of the performance enhancements associated with these bits is used unless explicitly requested. Any requested performance enhancements associated with PERFOPT2 that are not appropriate to a site for whatever reason has its corresponding bit turned off in PERFOPT2 during initialization, so that one can determine which optimizations are actually in effect by entering "V PERFOPT2" after the Online is up.

Enhancements are controlled by PERFOPT2 rather than PERFOPT where it is important for system administrators to be aware of issues associated with the enhancements before turning them on. By making the default for PERFOPT2 X'00000000', the associated enhancements are not used unless explicitly, and so (hopefully) consciously and conscientiously, set.

The PERFOPT2 bits are:

X'00000001' Turns on server swapping to expanded storage or, if expanded storage is not available, to main storage. Only the amount of storage required to hold all swapped out users is needed on the system, unlike swapping to disk where disk space needs to be allocated for the maximum size server for each user. If there is insufficient expanded and/or main storage to hold swapped out users, paging will result which can have a disastrous effect on performance.

Server swapping to expanded storage forces page alignment of servers, so it can increase main storage requirements for an Online by up to 4088 bytes per server.

Using this feature eliminates the need to have CCASERVR and CCASRVxx datasets, and, of course, it dramatically increases the speed of server swaps.

There are several system parameters that are available to facilitate monitoring the efficacy of the server swapping to expanded storage feature. To avoid changing the system statistics block format, this feature reuses (or misuses) some CMS system statistics. These statistics and their meanings are:

INVMFS The number of times a server read from expanded storage failed, because either an expanded storage page or server page was stolen or moved by MVS.

A large value for INVMFS indicates over-allocation of real and/or expanded storage and suggests paging associated with expanded storage server swapping. Paging is synchronous on a task level and so will have a severe negative effect on *Model 204* performance.

INVMIO The high-water mark of expanded storage pages used for swapped out servers. If this number should get at all close to 500,000 (2 Gigabytes), it is getting close to the limit of what the feature could handle so is a time to worry.

OUTVMFS The number of times a server write to expanded storage failed because either an expanded storage page or server page was stolen or moved by MVS. A large value for OUTVMFS indicates over-allocation of real and/or expanded storage.

These statistics are available both as system final stats and system partial stats and can also be monitored in *SirMon*.

X'00000002' Reduces QTBL and VTBL usage associated with \$functions, especially those that return numeric values and whose result is going to a floating point variable or that have unspecified numeric parameters. While this optimization is guaranteed to reduce QTBL and VTBL usage and CPU overhead, caution must be taken if code developed in a region running with this optimization might be moved to a region without, since the table size requirements will undoubtedly increase.

X'00000004' Reduces VTBL usage associated with intermediate expression results in a FOR loop. While this optimization is guaranteed to reduce VTBL usage, caution must be taken if code developed in a region running with this optimization might be moved to a region without since the table size requirements will undoubtedly increase. This optimization has no CPU benefit: it simply reduces the required size of VTBL.

X'00000008' Enables the use of IN GROUP MEMBER member FOR RECORD NUMBER. Caution must be taken if code developed in a region running with this feature might be moved to a region without since any code using IN GROUP MEMBER member FRN will simply not work.

X'00000010' Formerly, enabled the use of the *Sirius Performance Enhancements V2* User Language Macro Facility. As of *Sirius Mods* version 6.6, that facility is automatically available to Sirius API customers, and this bit setting is now irrelevant.

The RESLTHR System Parameter

The resident QTBL feature is designed to reduce the cost of a server swap: by having a shared copy of QTBL and NTBL for requests that get server swapped frequently, these tables need not be swapped. The quantity of data being server swapped is reduced, thus increasing the speed of server swaps. The *Model 204* parameter RESTHRSH controls this feature. It indicates the number of times a pre-compiled APSY procedure needs to server-swap before its QTBL and NTBL are made resident (shared).

This feature has the side benefit that after a procedure is made “resident,” QTBL and NTBL do not need to be APSY loaded, which reduces the overhead of doing an APSY load for resident procedures. Unfortunately, there is no way of getting this side benefit for procedures that are frequently APSY loaded but never server swapped, other than cumbersome tricks to make these procedures server swap the first few times they are run to make them resident.

The RESLTHR parameter provides a mechanism for causing procedures that are frequently APSY loaded to become resident, which saves on the overhead of APSY loading the QTBLs and NTBLs of these procedures. The value of RESLTHR (any number that can be expressed as a 32 bit integer) indicates the number of times a request needs to be APSY loaded before its QTBL and NTBL are made resident (shared).

This parameter is only available with the *Sirius Performance Enhancements V2*, and it can only be set on the EXEC card parameters or in the user0 parameters.

A greater level of control for QTBL and NTBL residency than either RESTHRSH or RESLTHR is provided by the SIRAPSY command (“[SIRAPSY](#)” on page 27).

CHAPTER 10 *The SIRAPSY System Command*

This SIRAPSY command allows the manual setting of the resident/non-resident attribute of a pre-compiled subsystem procedure. This is useful in overriding *Model 204*'s default behavior if the decisions it makes based on the *Model 204* RESTHRSH parameter and RESLTHR (“[The RESLTHR System Parameter](#)” on page 25) are not appropriate.

The SIRAPSY command can only be run against an active subsystem, so a logical place for SIRAPSY commands might be in a subsystem start/init procedure.

```
SIRAPSY DIS | DISPLAY | RES | NORES subsystem proc
```

SIRAPSY command syntax

where

subsystem is the name of the subsystem to which the command applies.

proc is the name of the procedure to which the command applies.

For example, the following command sets SIRMON procedure MOPR-MENU to become resident the next time it is loaded:

```
SIRAPSY RES SIRMON MOPR-MENU
```

The following command sets SIRMON procedure MOPR-SYSOVR so that it will never be set to resident. SIRAPSY NORES does not drop the residency attribute of a procedure once it is made resident.

```
SIRAPSY NORES SIRMON MOPR-SYSOVR
```

This command is only available with the *Sirius Performance Enhancements V2*, and it can only be issued by a system manager or system administrator.

Figures

Figures

