

---

# Sirius Mods Release Notes Version 6.1

**July, 2001**

---



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](mailto:support@sirius-software.com)  
World Wide Web: <http://sirius-software.com>

August 5, 2010

© 2010 Sirius Software, Inc.

---

## *Proprietary Notices*

The following products:

- *Fast/Backup*
- *Fast/Reload*
- *Fast/Unload*
- *Fast/Unload User Language Interface*
- *Janus TCP/IP Base*
- *Janus SOAP*
- *Janus Sockets*
- *Janus Web Server*
- *Sirius Functions*
- *Sirius Mods*

are proprietary products of Sirius Software, Inc.:

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

**Model 204®** is a proprietary product of Computer Corporation of America, a wholly-owned subsidiary of Rocket Software, Inc., which owns the trademark:

**Rocket Software Corporate Office**  
**M204 Division**  
**275 Grove Street**  
**Suite 3-410**  
**Newton, Massachusetts 02466-2272**  
**USA**

**SoftSpy™** is a proprietary product of Information Technology Systems:

**Information Technology Systems**  
**95 Wells Avenue**  
**Newton, Massachusetts 02459-3216**  
**USA**

---

## *Contents*

<b>Proprietary Notices</b> . . . . .	<b>ii</b>
<b>Contents</b> . . . . .	<b>iii</b>
<b>Chapter 1: Introduction</b> . . . . .	<b>1</b>
<b>Chapter 2: New Products Introduced in Mods 6.1</b> . . . . .	<b>3</b>
Janus SOAP . . . . .	3
Sirius Performance Enhancements V3 . . . . .	3
Reduced cost of searching GTBL (GTBLHASH>1) . . . . .	3
Subscripted field references (PERFOPT X'01000000' bit) . . . . .	4
String arguments to dollar functions (PERFOPT X'02000000' bit) . . . . .	5
IDENTIFY IMAGE statement (PERFOPT X'04000000' bit) . . . . .	5
Dollar functions that return strings (PERFOPT X'08000000' bit) . . . . .	5
WITH clause, assignment and comparison statements (PERFOPT X'10000000' bit) . . . . .	5
Field extraction with unlocked records (PERFOPT X'20000000' bit) . . . . .	6
Overhead of idle non-XDM CRAM threads (PERFOPT X'40000000' bit) . . . . .	7
<b>Chapter 3: All products</b> . . . . .	<b>9</b>
Limited Janus Web Server - Free Use of Janus Web Server . . . . .	9
HTML or TEXT block . . . . .	10
<b>Chapter 4: Fast/Unload User Language Interface</b> . . . . .	<b>11</b>
FNVMASK . . . . .	11
<b>Chapter 5: Sirius Functions</b> . . . . .	<b>13</b>
\$BASE64_DECODE & \$BASE64_ENCODE . . . . .	13
\$COMMAND, \$COMMBG & \$COMMNDL available for Janus Sockets & Janus Web Server . . . . .	13
\$UNBINDW . . . . .	13
\$UNSPACE . . . . .	13
<b>Chapter 6: Janus TCP/IP Base</b> . . . . .	<b>15</b>
VSE Support . . . . .	15

<b>Chapter 7: Janus Web Server</b> . . . . .	<b>17</b>
\$WEB_END_SSLSES . . . . .	17
<b>Chapter 8: Compatibility/Bug fixes</b> . . . . .	<b>19</b>
Sirius Functions . . . . .	19
Non-zero value returned by \$SIRPROD . . . . .	19
Fixes in <b>Sirius Mods</b> 6.1 but not in 6.0 . . . . .	19
Janus Sockets . . . . .	19
Version co-requisites . . . . .	20

---

**CHAPTER 1** ***Introduction***

This document lists the enhancements and other changes contained in the newest release of *Sirius Mods*: version 6.1. The previous generally released version of *Sirius Mods*, 6.0, was released in December, 2000.



---

**CHAPTER 2** *New Products Introduced in Mods 6.1*

## 2.1 Janus SOAP

Version 6.1 of the *Sirius Mods* introduces *Janus SOAP*, the newest member of the Janus family of products. *Janus SOAP* provides User Language programmers with facilities for processing eXtensible Markup Language (XML) documents. *Janus SOAP* adds support for tree-structured data objects to User Language, introducing rudimentary Object Oriented (OO) programming concepts to *Model 204*.

A set of \$functions provides an Application Programming Interface (API) for navigating the tree-structured objects. Functions are provided for transferring data between the tree-structured objects and simple User Language objects such as percent variables and images. For more information, please refer to the *Janus SOAP Reference Manual*

## 2.2 Sirius Performance Enhancements V3

*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. Each enhancement is separately activated by a parameter setting, as described below. Although packaged as a Sirius product for use with *Model 204* Version 4.2, the entire set of enhancements have been integrated with *Model 204* Version 5.1.

### 2.2.1 Reduced cost of searching GTBL (GTBLHASH>1)

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 right on the edge 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 don't keep any statistics on how many times this happens. The lesson would be to not run GTBL "close to the edge" anyway, since you would be likely to get an occasional \$SETG failure.

Similarly if GTBLPCT is incorrect there will be some overhead dynamically fixing things up to reflect the actual usage. This fixup is done either when a GTBLHASH section fills up or when the portion of GTBL not used for global variables would expand into the last GTBLHASH section. Again there are no stats that keep track of this and there are really no stats that help one set the correct value of GTBLPCT. However, we have not seen the "fixup" code appear in any SirTune reports so its overhead doesn't seem too dramatic.

## 2.2.2 Subscripted field references (PERFOPT X'01000000' bit)

Complex repeating group structures are quite common in *Model 204* databases. Common wisdom has evolved the following code fragment as the best way to iterate over all occurrences of a repeating group, with this example having four fields in the group:

```
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.

The PERFOPT=X'01000000' optimization provides subscripted field references with with the same type of optimization as FEO. At compile time we detect a variable occurrence number reference to a field and allocate a special field position variable for that reference. At the time the field extraction is done for the first occurrence of a field we save the position in the record just as FEO saves its position. On subsequent extractions we determine if the previously saved position is still usable. The occurrence is usable if the occurrence number being requested is greater than the previous occurrence number retrieved, the record is locked (share or exclusive) or 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, and the record has not been updated since the position was saved.

A second part of this optimization reduces the cost when the subscript is a variable declared as FIXED or when the subscript is of the "OCCURRENCE IN" form. This is accomplished by eliminating extra quads and processing required to convert the fixed values to a float format, resulting in 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.

### **2.2.3 String arguments to dollar functions (PERFOPT X'02000000' bit)**

*Model 204* uses a general purpose routine to resolve the values of arguments passed to dollar functions. As the types of variables supported by *Model 204* have proliferated, the path length for this routine has grown. The PERFOPT=X'02000000' optimization improves the performance of passing arguments to dollar functions in two important cases - simple string percent variables and null strings or missing arguments. This optimizations saved about 2.5% of the overall CPU for one customer.

### **2.2.4 IDENTIFY IMAGE statement (PERFOPT X'04000000' bit)**

This optimization eliminated 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. The CPU savings was about 0.7% for one large customer.

### **2.2.5 Dollar functions that return strings (PERFOPT X'08000000' bit)**

This optimization reduces the instruction pathlength for dollar functions that return string values to string percent variables, by eliminating a generated assignment quad. One large site measured a 0.2% reduction in CPU usage and a 5% reduction in QTBL usage.

### **2.2.6 WITH clause, assignment and comparison statements (PERFOPT X'10000000' bit)**

This enhancement builds upon the observations that drove the PERFOPT=X'02000000' enhancements. Optimized versions of the assignment, comparison and WITH quads were built to reduce their overhead in common simple cases (simple percent variable to percent variable movements, string to string comparisons, fixed to fixed assignment, etc.). In addition the pathlength for cases where comparisons resulted in a JUMP (for example an IF clause that evaluates to false) was reduced. These enhancements saved about 1.3% of the CPU for one large site.

### 2.2.7 Field extraction with unlocked records (PERFOPT X'20000000' bit)

This optimization 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 N'th occurrence of a field if the record is not locked. This is required 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. The concept is that when we would ordinarily de-optimize an FEO or subscripted field reference we do a quick check to see if anyone might have changed the record since we saved our position. If not, then even though we don't have the record locked we can use our saved position.

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 parameter (for Unlocked FEO HASH). 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 obviously be associated with many records from many files. There is a system-wide doubleword counter that 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 increments the system-wide update counter. Then the file/record combination is hashed to determine its entry in the UFEOHASH 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 UFEOHASH table entry.

Before reusing a position in a record that is not locked, the update counter in the appropriate UFEOHASH 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 UFEOHASH 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, in that it might refuse to use a saved position that is safe, but the resulting simplicity keeps the cost to an acceptable level. Note also that this approach optimizes **all** FEO's and subscripted field references in a read-only environment since the hash cells will always contain an update counter of zero.

### **2.2.8 Overhead of idle non-XDM CRAM threads (PERFOPT X'40000000' bit)**

This enhancement reduces the overhead associated with idle IODEV=11 and IODEV=29 (BATCH2) threads, eliminating the performance penalty for over-allocating. The 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, hence 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.



### 3.1 Limited Janus Web Server - Free Use of Janus Web Server

A new capability, *Limited Janus Web Server*, is available starting with this release. This allows you to write limited-scale *Janus Web Server* applications without requiring you to purchase a copy of *Janus Web Server*. This allows you to implement low-volume applications using the power of the Web, even if your only *Sirius Mods* product is, for example, *Fast/Backup* or *Fast/Reload*.

The *Limited Janus Web Server* capability is provided as a no-charge update to all customers licensed for any *Sirius Mods* product (except for *Janus Web Server*). If a *Model 204* load module is authorized for any *Sirius Mods* product other than *Janus Web Server*, that load module will be automatically enabled for *Limited Janus Web Server* for as long as at least one *Sirius Mods* product remains authorized. See the Overview chapter in the ***Sirius Mods Installation Guide*** for a list of all products which are in the *Sirius Mods*.

*Limited Janus Web Server* consists of the following:

- Authorization to use the \$WEBxxx functions.
- If *Janus TCP/IP Base* is not otherwise authorized, authorization to it with 3 threads.
- Authorization to define and use WEBSERV ports with up to 3 connections.
- Authorization to use *SirScan* to view the audit trail entries of any WEBSERV thread.
- Authorization for the HTML/TEXT statement and the SIREDDIT and WEBAUDIT parameters.
- Customer support and maintenance for *Limited Janus Web Server*.

The limitations imposed in *Limited Janus Web Server* are as follows, relative to a full *Janus Web Server* license:

- At most 3 concurrent JANUS WEBSERV threads may be active.
- *Janus Web Server* authorizes many \$functions beyond the \$WEBxxx functions; the only \$functions authorized by *Limited Janus Web Server* are the \$WEBxxx ones.

Sirius Software will encourage and assist users among its customer base to develop applications that can be shared amongst *Limited Janus Web Server* and *Janus Web Server* customers. Of course, Sirius Software hopes that the functionality provided by

*Limited Janus Web Server* convinces you to purchase full *Janus Web Server* and *SirScan* licenses.

### **3.2 HTML or TEXT block**

The HTML and TEXT User Language statements are now available to users of the *Fast/Unload User Language Interface*, *Janus Sockets*, or of *Limited Janus Web Server*. This is in addition to the previous authorization, which made these statements available to customers who own *Janus Web Server* or the *Sirius Functions*.

The following features are new or changed in the *Fast/Unload User Language Interface*:

## **4.1 FNVMASK**

The current value of the FNVMASK parameter is now automatically communicated to *Fast/Unload* when it is invoked by the *Fast/Unload User Language Interface*. As stated in [“Version co-requisites” on page 20](#), if FNVMASK is non-zero, then either *Fast/Unload* version 4.1 or later, or version 4.0 with ZAP4022 applied, must be used for the unload.



The \$functions presented in this chapter are new or have changed in version 6.1.

## 5.1 **\$BASE64\_DECODE & \$BASE64\_ENCODE**

The \$BASE64\_DECODE function converts a byte string from a base 64 encoding to the decoded byte string. The \$BASE64\_ENCODE function converts a byte string into its base 64 encoding.

These \$functions were previously introduced by ZAP6042.

## 5.2 **\$COMMAND, \$COMMBG & \$COMMNDL available for Janus Sockets & Janus Web Server**

The \$COMMAND, \$COMMBG, and \$COMMNDL functions are now available to customers who own either *Janus Sockets* or *Janus Web Server*. This is in addition to the previous authorization, which made these \$functions available to customers who own the *Sirius Functions*.

## 5.3 **\$UNBINDW**

\$UNBINDW allows semaphores bound by \$BIND to be unbound. The difference between \$UNBIND and \$UNBINDW is that \$UNBINDW allows wildcard characters in the semaphore name so that multiple semaphores can be unbound in a single call.

## 5.4 **\$UNSPACE**

\$UNSPACE normalizes a string by removing leading and trailing “space” characters and collapsing other sequences of “unquoted” spaces to single spaces. Normalization can also “undouble” quoted quote characters.



---

**CHAPTER 6**    ***Janus TCP/IP Base***

The features presented in this chapter are new or have changed in version 6.1.

## **6.1 VSE Support**

*Janus TCP/IP Base* is now supported for users of VSE/ESA 2.3 and later with Connectivity Systems Inc. TCP/IP V1.4 and later. This provides VSE support for *Janus Web Server*, *Janus Network Security*, *Janus Specialty Data Store*, *Janus Open Server*, *Janus Open Client* and *Janus Sockets*



---

**CHAPTER 7** *Janus Web Server*

The features presented in this chapter are new or have changed in version 6.1.

## **7.1 \$WEB\_END\_SSLSES**

Terminates an SSL session. This function is useful for *Janus Network Security* customers using client certificate based authentication. Terminating an SSL session with most browsers forces the end-user to provide a client certificate on subsequent connections to the server. This is important in allowing users with multiple certificates to switch certificates and in providing a true logout capability — without it, a timed out logical session would probably receive the client certificate from the timed out session and simply allow the login without any end-user interaction, making it appear as though there wasn't any session timeout at all.



This chapter lists any compatibility issues with prior versions of the *Sirius Mods* and any bugs which have been fixed in this version of the *Sirius Mods* but had not, as of the date of this release, been fixed in the immediately prior version (6.0).

The first sections (that is, all sections before “Fixes in *Sirius Mods* 6.1 but not in 6.0”) list, by product, any backwards incompatibility issues, that is, any differences in processing that result from successful execution with *Sirius Mods* version 6.0, as compared with the same inputs to *Sirius Mods* version 6.1.

## **8.1 Sirius Functions**

### **8.1.1 Non-zero value returned by \$SIRPROD**

Prior to 6.1, \$SIRPROD always returned the value 1 if the Sirius product specified by the argument value was authorized. Starting with 6.1, other non-zero values may be returned to indicate that the specified product is available.

## **8.2 Fixes in *Sirius Mods* 6.1 but not in 6.0**

This section lists other fixes to functionality existing in the *Sirius Mods* version 6.0 but which, in absence of customer problems, have not, as of the date of the release, been fixed in that version.

### **8.2.1 Janus Sockets**

#### **Missing SOCKPMAX on JANUS DISPLAY**

If the value of SOCKPMAX had been set to a non-default value, it was not displayed by the JANUS DISPLAY command.

## **8.3 Version co-requisites**

This section lists any restrictions on usage of various products (including *Sirius Mods* itself) which will be imposed by use of version 6.1 of *Sirius Mods*.

### *Fast/Unload User Language Interface* **with non-0 FNVMASK**

If FNVMASK is non-zero and the *Fast/Unload User Language Interface* is used, then either *Fast/Unload* version 4.1, or version 4.0 with ZAP4022 applied, must be used for the unload. If not, the User Language request is cancelled. Also, during *Model 204* initialization, an operator warning is issued and saved in the VIEW ERRORS table, so that corrective action may be taken in advance.