
Fast/Unload Release Notes Version 4.0

January, 2000



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>

August 5, 2010

© 2010 Sirius Software, Inc.

Proprietary Notices

The following products:

- *Fast/Reload*
- *Fast/Unload*
- *Fast/Unload User Language Interface*
- *Sirius Mods*
- *SirZap*
- *Sir2000 Field Migration Facility*

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

Contents

Proprietary Notices	ii
Contents	iii
Chapter 1: Introduction	1
Summary of significant changes	1
Performance	1
EXEC parameters	1
FUEL statements	1
#Functions	1
Special variables	2
MISSING value handling	2
Debugging enhancements	2
Statistics	2
Backwards compatibility with Fast/Unload 3.1	2
Version co-requisites	3
Chapter 2: Maintenance and Support	5
Chapter 3: New Features	7
Performance enhancements	7
New or changed program parameters	8
New FUEL statements	8
ADDC	8
DELETEC	8
FSTATS [AVGTOT MINMAX]	8
REPEAT and LEAVE	9
UNLOAD[C] field	9
New #functions	10
Special variables: output length and position	11
MISSING value handling	11
FSTATS improvements	12
FUEL outside FOR EACH RECORD loop	14
Stricter date matching	14
Other date handling enhancements	15
Debugging enhancements	15
Miscellaneous enhancements	16

Chapter 4: Compatibility/Fixes 19

 Backwards compatibility with Fast/Unload 3.1 19

 Fixes in Fast/Unload 4.0 but not in 3.1 21

 Rounding of numeric values 21

 #DATE2Nx prior to 1900, #DATE2NS seconds since midnight 24

 #Nx2DATE non-numeric date number 24

 Null REPORT lines 25

 Version co-requisites 25

CHAPTER 1 *Introduction*

This document lists the enhancements and other changes contained in the newest release of *Fast/Unload*: version 4.0. The previous generally released version of *Fast/Unload*, 3.1, was released in October/November, 1997.

Sirius Software strongly recommends that you migrate all *Fast/Unload* jobs (and modules used by the *Fast/Unload User Language Interface*) to version 4.0 of *Fast/Unload*, because we expect to be able to resolve both user errors and bugs much more quickly than with previous versions.

1.1 Summary of significant changes

Fast/Unload version 4.0 introduces the following significant changes.

1.1.1 Performance

- Large NEBUFF
- Full-cylinder size of buffers, whether database is cylinder aligned or not

1.1.2 EXEC parameters

- ABENDERR: ABEND based on return code
- NOLIST: suppress listing of FUEL program

1.1.3 FUEL statements

- ADDC: If value is MISSING, just ignore the operation
- DELETEC: If occurrence is MISSING, just ignore the operation
- LEAVE: To jump out of FOR/FROM/TO, REPEAT, or SELECT
- REPEAT: To loop until condition satisfied
- UNLOAD <field>: Allows you to re-order fields in UAI

1.1.4 #Functions

- #CONCAT_TRUNC: Concatenate strings, allowing truncation
- #DEBLANK, #STRIP: Remove leading, trailing blanks or pad characters from string or substring

- #DELWORD, #FIND, #WORD, #WORDS: Operate on string of blank-delimited words
- #LEFT, #PAD, #PADR, #RIGHT: Initial/final substring, with pad characters
- #LOWCASE, #TRANSLATE, #UPCASE: Character translation
- #ONEOF: See if string is in delimited list of strings
- #NUM2STR: Convert number to string with decimal point and specified number of integer and fraction digits.
- #REVERSE: Get reverse of string

1.1.5 Special variables

- #OUTLEN: output length and position
- #OUTPOS: output position

1.1.6 MISSING value handling

Numeric use of MISSING value is zero, simplifying FUEL coding

1.1.7 Debugging enhancements

- If a FUEL program error occurs and the listing is suppressed (e.g., with the *Fast/Unload User Language Interface*), the number of the line in error is printed
- Many errors previously reported as simply “syntax error” now print a more likely cause of error, namely, “unknown field”
- If a common ABEND error occurs and a dump is suppressed (e.g., no more space in output dataset), an explanation of the abend is printed

1.1.8 Statistics

- Job I/O statistics
- Table B statistics (record lengths, extension records, etc.)
- New field statistics: average and totals for occurrence counts and field lengths, counts of records missing field, complete field definition information

1.1.9 Backwards compatibility with Fast/Unload 3.1

Handling of numeric values has been changed, so that it is now consistent with User Language. This affects addition and subtraction, if it involves a number greater than or equal to 10 to the 15th power or a number with a non-zero fraction part, or conversion from a numeric string to any numeric type, if the number contains more than 15 significant digits or is a fraction with more than 16 digits after the decimal point.

1.1.10 Version co-requisites

Output from UAI with *Fast/Unload* version 4.0 requires version 5.2 or later of *Fast/Reload* to reload the file.

CHAPTER 2 *Maintenance and Support*

These enhancements to *Fast/Unload* do not affect its intrinsic functionality, but rather affect the way Sirius Software delivers maintenance and support for the product.

Expiration information

To allow you to beware that *Fast/Unload*, or one of its separate features, will expire at some time, new messages produced in FUNPRINT will list the expiration dates of all authorized products.

Authorization zap checksum

All *Fast/Unload* authorization zaps now contain a checksum which is stored with the zap. If you apply an authorization zap with incorrect content (for example, by mis-reading a fax), a FUNL error message is issued which prevents *Fast/Unload* from running. This checksum is in addition to a checksum which is displayed in a comment in the zap and which will be used with a future enhancement to *SirZap* to alert you to zap errors at the time of application.

CHAPTER 3 *New Features*

3.1 Performance enhancements

The following changes help you to achieve improved performance with *Fast/Unload*:

Large NEBUFF handling In *Fast/Unload* version 3.1, using a large number of extension buffers (e.g., NEBUFF 5000) could cause a significant CPU cost, because when a page is not found in the buffered extension pages, all buffers were searched. A new algorithm in version 4.0 uses a hashed search approach, so that only a few buffers are searched, even if the page is not found among all buffered extension pages.

Therefore, if you have enough real memory available and the extension buffer wait time is a significant portion of your overall job time, you should be able to increase NEBUFF to a large number and achieve CPU as well as wait time savings.

As noted in [“Miscellaneous enhancements” on page 16](#), several new **Job statistics** can help you analyze and tune the I/O performance of *Fast/Unload*.

Multi-track reads

In *Fast/Unload* version 3.1, in order to use SBBUFF or SEBUFF other than 1, the input *Model 204* file must be allocated to datasets aligned on cylinder boundaries. Now you may specify buffer sizes other than 1 for any dataset if it is allocated to a device which supports the DEFINE EXTENT command (most modern DASDs support DEFINE EXTENT).

This allows multi-track read for almost all files. In general, SBBUFF=15 will provide the best performance. Assuming all your files are on supporting DASD, you can place this in your standard JCL, without worrying about whether specific datasets are cylinder aligned.

3.2 New or changed program parameters

The following new parameters may be passed to *Fast/Unload*:

ABENDERR=n

ABENDERR may be used to trigger an ABEND at the end of the run, if *Fast/Unload* encounters an error resulting in a return code of *n* or greater. The default is zero, which means that *Fast/Unload* will not trigger an ABEND due to the return code; this default can be changed by a customization zap. However, the default for use with the User Language Interface is always 0.

FSTATS[=AVGTOT|MINMAX]

You may qualify the FSTATS parameter to indicate the type of FSTATS processing to produce (see “[FSTATS \[AVGTOT | MINMAX\]](#)”).

NOLIST | LIST

NOLIST may be used to suppress the program listing; LIST is used to produce it. The default is LIST (unless using the User Language Interface without the ALLMSG parameter); this default can be changed by a customization zap.

3.3 New FUEL statements

This section lists new statements which may be used in *Fast/Unload*.

3.3.1 **ADDC**

The ADDC variant of the ADD field statement allows the MISSING value to the right of the equal sign (“=”), in which case the statement is a no-op.

3.3.2 **DELETEC**

The DELETEC variant of the DELETE field statement allows you to specify a field occurrence which is missing on the current record, in which case the statement is a no-op.

3.3.3 **FSTATS [AVGTOT | MINMAX]**

The FSTATS directive in FUEL can be used instead of the FSTATS program parameter, to cause *Fast/Unload* to collect information about fields in the file and about records in Table B.

As described in “[FSTATS improvements](#)” on page 12, new information is collected for FSTATS in version 4.0 of *Fast/Unload*. A multi-line display is used for the following new information:

- averages and totals for each field's occurrences and lengths
- counts of records missing the field, if there are any
- all non-default field definition information, if there is any other than storage type

To restrict the FSTATS field display to only contain the storage type and the minimum and maximum of occurrences and lengths, you can use the FSTATS MINMAX directive; FSTATS AVGTOT requests the more complete field information.

If you specify the FSTATS directive in your FUEL program without a qualifying MINMAX or AVGTOT, or if you use the FSTATS program parameter without the FSTATS directive in your FUEL program, the default processing is AVGTOT. You can change this default to be MINMAX by a customization zap.

3.3.4 REPEAT and LEAVE

FUEL now has a block which allows you to loop based not on a count, but based on satisfying some condition you specify. This block is implemented by the REPEAT statement. The LEAVE REPEAT statement is used to terminate a REPEAT loop. Other forms of LEAVE can also be used to terminate other kinds of FUEL blocks.

END REPEAT

This marks the end of a REPEAT loop.

LEAVE {FOR | REPEAT | SELECT}

This jumps out of the closest enclosing FOR v FROM loop, REPEAT loop, or SELECT clause, respectively.

REPEAT This marks the start of a REPEAT loop, which is executed repeatedly until terminated, usually by a LEAVE REPEAT statement.

3.3.5 UNLOAD[C] field

The syntax of the UNLOAD statement has been modified, allowing you to specify a field, and either an occurrence or “*” (designating all not-unloaded occurrences of the field). This statement (called “UNLOAD field”) allows you to specify the order that fields are unloaded in a record (and hence the order they will be reloaded in the *Fast/Reload LAI*). In addition, you can omit the “normal UNLOAD” statement for a record, so that only the data in explicit UNLOAD field statements is unloaded.

The UNLOADC variant of the UNLOAD field statement allows you to specify a field occurrence which is missing on the current record, which unloads an initial *Model 204* record, if it is the first UNLOAD for the record, and otherwise is a no-op.

3.4 New #functions

The following new #functions may be used in *Fast/Unload*:

#CONCAT_TRUNC	Concatenate strings, allowing truncation.
#DEBLANK	Remove leading and trailing blanks from substring.
#DELWORD	Remove blank-delimited words from string.
#FIND	Word position of one word sequence within another.
#LEFT	Initial substring, followed by pad characters to specified length.
#LOWCASE	Change uppercase letters of string to lowercase.
#ONEOF	See if string is in delimited list of strings.
#NUM2STR	Convert number to string with decimal point and specified number of integer and fraction digits.
#PAD	Final substring, preceded by pad characters to specified length.
#PADR	Initial substring, followed by pad characters to specified length.
#REVERSE	Get reverse of string
#RIGHT	Final substring, preceded by pad characters to specified length.
#STRIP	Remove leading and/or trailing copies of pad character.
#TRANSLATE	Change characters of string using from/to pairings.
#UPCASE	Change lowercase letters of string to uppercase.
#WORD	Return nth blank-delimited word from string.
#WORDS	Count number of blank-delimited word in string.

3.5 Special variables: output length and position

Two new special variables have been introduced in version 4.0 of *Fast/Unload*: #OUTLEN and #OUTPOS. #OUTPOS reflects the current output position that will be used by the next PUT statement (if it does not contain the AT clause). #OUTLEN reflects the length of the current output record. Both of these are available only in a non-UAI type of unload.

3.6 MISSING value handling

Several changes have been made in version 4.0 of *Fast/Unload* to make it easier to deal with the MISSING value, and to make the handling of the MISSING value more consistent.

As in *Fast/Unload* version 3.1, the use of the MISSING value as a string produces the null, or zero-length, string.

The most significant change to the handling of the MISSING value is in a context in which a numeric value is expected; it is treated as zero.

For example, this means that %variables do not need to be initialized to zero; a statement such as

```
%COUNT = %COUNT + 1
```

can be placed in your FUEL program **without** the initializing:

```
%COUNT = 0
```

As another example, this means you can reference missing field occurrences without testing; a statement such as

```
%TOTAL = %TOTAL + SALARY
```

can be placed in your FUEL program **without** requiring a test such as

```
IF SALARY EXISTS  
  %TOTAL = %TOTAL + SALARY  
END IF
```

The handling of the MISSING value passed as an argument to a #function is determined on a case-by-case basis, but it usually follows the rules that the null string is used in string contexts, and zero is used in numeric contexts. This means, for example, that if the value of %MISS is MISSING, the following FUEL fragment:

```
%S = #SUBSTR('ABC', 1, %MISS)  
REPORT '>' WITH %S WITH '<'
```

prints the line "><".

The exceptions in the standard #functions is the CENTSPAN argument for date #functions, and the first argument of the #Nx2DATE functions; neither of these cases allow the MISSING value.

If your site uses any customer-written #functions, you should check them to see how the MISSING value behaves. Anyone responsible for these #functions should refer to the discussion of "Customer-written #function interface" in ["Backwards compatibility with Fast/Unload 3.1"](#) on page 19.

3.7 FSTATS improvements

The information provided by the FSTATS feature has been expanded, both to provide more information about fields and to provide information about Table B records. You can choose to restrict the field information so that it is essentially the same as provided in *Fast/Unload* version 3.1, using the FSTATS MINMAX directive, or you can use FSTATS AVGTOT to provide the comprehensive field information. AVGTOT is ordinarily the default for FSTATS processing; see ["FSTATS \[AVGTOT | MINMAX\]"](#) on page 8 for an explanation of specifying MINMAX or AVGTOT.

In summary, here are the changes to the information provided for each field:

INVISIBLE or NEW fields

Fields defined as INVISIBLE are now included in the list of fields, and flagged as such. Fields introduced with the NEW directive in *Fast/Unload* are now flagged as such in the list of fields. (Neither of these provide any details about their lengths or occurrences.)

Complete definition

All non-default information set by the DEFINE FIELD command is provided in the FSTATS AVGTOT display.

Records with zero occurrences

FSTATS AVGTOT displays the count of records which contain zero occurrences of the field, if there are any such records.

Totals and averages

FSTATS AVGTOT displays the total number of occurrences of the field, the average occurrences per record (among records with at least one occurrence of the field), and the total length and average length of all occurrences of the field.

In addition to the field information, there are other changes to FSTATS processing; these are done for both AVGTOT and MINMAX:

Table A consistency

Some possible inconsistencies are detected in the field definition information recorded in Table A.

Table B pages in use, Base records and Extension Records in file

These are taken from the file parameters.

Base records processed

This is the same value that is shown the FUNL0054 message; it is simply the number of *Model 204* records processed as input to *Fast/Unload*.

Base records processed without extensions

This is the number of base records processed which do not have an extension record.

Extension records processed

This is the total number of extension records of the base records processed.

Average extensions per extended record, Maximum extension chain length processed

These are, respectively, the average number of extension records and the maximum number of extensions for a single base record, both among the base records processed that have extensions.

Non-adjacent extension records processed

This is the total number of extension records processed which reside on a page other than the next page after the record (either base or extension) that contains the pointer to that extension record. It provides a measure of the number of records which have been extended since the last file reorganization. Depending on the FILEORG of the file (and to some extent the level of multiprocessing among updating transactions) it includes or excludes record which are extended in a transaction subsequent to the transaction in which they are created.

This can be one indicator of a need for a file reorganization, if your file contains extension records which are necessary due to very long records. As noted in [“Miscellaneous enhancements” on page 16](#), “Extension pg in base buffs”, which is a new **Job statistic**, can provide additional information about the placement of extension records.

Minimum, maximum, total, and average record length and standard deviation

These are, respectively, the length of the smallest record processed, the length of the largest record processed, the average of the lengths of all records processed, and the statistical standard deviation from the average of the lengths of all records processed.

The record length used in these calculations is the Table B space used by the “internal” portion of a record. That is, it includes the space for

preallocated fields and the space used to represent non-preallocated fields, but it does not include the Table B space used for extension record pointers, pointers to slots on the page, nor spill pointers.

3.8 FUEL outside FOR EACH RECORD loop

FUEL statements (e.g., assignment statements, REPORT, etc.) may now be executed before the FOR EACH RECORD loop, and after the end of the loop. The former might be useful for initialization, and that latter might be useful for printing totals. (This feature was also implemented as part of the maintenance zaps to *Fast/Unload* versions 3.0 and 3.1, but was not previously mentioned in any release notes.)

3.9 Stricter date matching

In some #functions, the rules for matching a date to a format have been made more strict. This allows your applications to check date validity in the same manner that will be enforced by the *Sir2000 Field Migration Facility*.

The stricter date processing is as follows:

MM, DD, HH

With strict matching, these tokens will only match all numeric values; with non-strict matching, a leading blank is allowed.

ZYY, DAY

With strict matching, these tokens will not allow a leading zero; with non-strict matching, a leading zero is allowed.

MON, MONTH, WKD, WKDAY

With strict matching, these tokens will match only all uppercase strings; with non-strict matching, any case is allowed.

Mon, Month, Wkd, Wkday

With strict matching, these tokens will match only all strings with a leading uppercase letter followed by all lowercase letters; with non-strict matching, any case is allowed.

In addition, the following tokens are matched strictly with all date format usages:

DDD

This must be all numeric. In some previous versions of *Fast/Unload*, one or two leading blanks were allowed.

YY, MI, SS

These must be all numeric. In some previous versions of *Fast/Unload*, one leading blank was allowed.

The #functions which now use stricter date matching are:

#DATECHG

#DATECHK

#DATECNV

#DATEDIF

3.10 Other date handling enhancements

" date format token

The quote mark allows you to specify any character, e.g., an alphabetic letter, as a separator character (literal character).

Numeric date format separators

Any decimal digit stands for itself as a separator character (literal character).

BM, BD, BH date format tokens

These tokens correspond to MM (month number), DD (day number), and HH (hour number), except that on output, the first character will be blank if the number is less than 10.

UAI SORT support for 2-digit years

The UAI statement with the SORT clause now allows a FORMAT specification, which may be used with your sort package to indicate that a field contains a 2-digit year date. Also, the SORT OPTION statement may now be specified in combination with the UAI statement, so that you may specify the 100-year window to the sort package.

Note:

- This change is also available as a zap for *Fast/Unload* versions 3.0 and 3.1.

3.11 Debugging enhancements

Error line number for suppressed listing

If the FUEL source listing is suppressed (e.g., for the *Fast/Unload* User Language Interface without the ALLMSG parameter), any compilation error message is preceded by a message showing the line number containing the error.

Unknown field errors

In almost all cases in which every token type has been checked and the last possible legal choice is a field name, and the current input token does not match a field name, rather than producing a simple FUNL0036 Syntax error ... message, a message is now issued which contains the input token, and a description that it is an unknown field name.

UAI trailer records

The FUNOUT dataset for a UAI operation now contains various trailer records, which LAI will check to make sure a complete and correct FUNOUT dataset was supplied as the TAPEI input. As a result of this, as mentioned in [“Version co-requisites” on page 25](#), the LAI must then be done using version 5.2 or later of *Fast/Reload*.

Common ABEND explanations

In some cases, *Fast/Unload* encounters an abend due to a common error situation, such as insufficient space in the FUNOUT dataset, and suppresses a dump for these abends. *Fast/Unload* will now print a message explaining these errors to help you recover from the error.

3.12 Miscellaneous enhancements

Job statistics

In *Fast/Unload* version 3.1, the CPU time and various wait times are displayed for each phase (Compile, Unload, and Index unload) of a *Fast/Unload* job (these statistics can also be captured as SMF records under MVS). A new set of statistics have been added, showing the I/O work accomplished in each phase. The new statistics are:

- Base buffer reads: This indicates the number of EXCPs issued for reading data into the base buffers.
- Base buffer waits: This indicates the number of WAITs issued after initiating reads into the base buffers.
- Extension pg in base buffs: This indicates the number of *Model 204* file pages accessed “directly” which were currently available in pages read into base buffers. As mentioned in [“FSTATS improvements” on page 12](#), this information can be used in addition to the FSTATS information about extension records, to help you determine possible benefits obtained by reorganizing a file.
- Extension pg in exten pool: This indicates the number of *Model 204* file pages accessed “directly” which were not currently available in pages read into base buffers but which were available in the pool of extension pages.

- **Extension buffer reads:** This is the number of EXCPs to read pages accessed “directly”. It is the number of such pages that could not be found either in the base buffers or in the extension buffer pool.

CMS "job name"

The CMS user ID running *Fast/Unload* is used in place of the MVS job name (FUNPRINT headers, file ENQ entry).

Customize SORT program name

You may set the default SORT program name by applying a zap to the *Fast/Unload* load module.

Checking for null extension records

If you use the FSTATS parameter, all null extension records will be noted on the FUNPRINT dataset. These messages indicate an unnecessary condition in the *Model 204* file which can have negative impact on performance.

This section lists any compatibility issues with *Fast/Unload*, and any fixes contained in this version of *Fast/Unload* but not, as of the date of this release, in the immediately prior version (3.1).

4.1 Backwards compatibility with Fast/Unload 3.1

This section lists any differences in processing that result from execution with *Fast/Unload* version 4.0, as compared with the same inputs to *Fast/Unload* version 3.1 at current maintenance levels.

OPEN statement

In prior versions, anything on the OPEN statement after the file name was ignored. Starting with 4.0, nothing is allowed after the file name.

PAI statement

In prior versions, the PAI statement could be invoked multiple times for a single input record. Starting with 4.0, the PAI statement can only be issued once for a single input record.

Rounding of some numeric values

The value of the low order bits in the floating point representation of numeric values has been changed, so that it is now consistent with User Language. This will affect the following situations in *Fast/Unload*:

1. Addition and subtraction, if it involves a number greater than or equal to 1,000,000,000,000,000 (10 to the 15th power) or involves a number with a non-zero fraction part.
2. Conversion from a numeric string to a numeric type, if the number contains more than 15 significant digits or is a fraction with more than 16 digits after the decimal point.

See “Rounding of numeric values” on page 21 for further discussion.

#DATE2Nx

See “#DATE2Nx prior to 1900, #DATE2NS seconds since midnight” on page 24.

#Nx2DATE

See “#Nx2DATE non-numeric date number” on page 24.

Null REPORT lines

See “Null REPORT lines” on page 25.

Missing required argument

In prior versions, *Fast/Unload* did not always check whether a #function argument is supplied, if the argument was after the first. For example, the following FUEL fragment:

```
%S = #DATECHG('YYDDD', '99001')  
REPORT %S
```

formerly printed the value “99001”, even though the documentation states that the third argument to #DATECHG is a required argument. Starting in version 4.0, missing arguments will cause an error, if they are required arguments.

Missing #CONCAT argument

As a slight variation on the change which detects missing #function arguments, you are not allowed to code missing arguments to the #CONCAT function. For example, the following FUEL fragment:

```
%S = #CONCAT('A', 'B', , 'D')
```

will produce an error message, starting with version 4.0. Note that this is a run-time error; the status of an argument as required or optional is checked at compile time strictly based on the argument number, and the third argument is optional (since #CONCAT with only two arguments is allowed). Formerly, missing arguments to #CONCAT (except the first) were allowed and were treated as the zero-length string; for example, the above FUEL fragment printed the value “ABD”.

Also note that at least two arguments are required by #CONCAT; the documentation did not previously state this.

Customer-written #function interface

In prior versions, the “strict” argument value extraction services (FUNQ2SS, FUNQ2FS, and FUNQ2BS) did not allow an argument to have the MISSING value. They now do, producing either the null string (FUNQ2SS) or zero (FUNQ2FS and FUNQ2BS); termination is only caused by an omitted argument or an unconvertible value (the latter for FUNQ2FS and FUNQ2BS).

If you have written assembler-language #function routines that use the strict services, we recommend that you continue to use them, allowing the same meaning of MISSING as in standard #functions and other FUEL statements.

If, however, you require that MISSING values be prevented in your #functions, you can check for, and disallow, the MISSING value upon return from the argument value extraction services. As a reminder, your #functions can terminate *Fast/Unload* using the FUNQTRM service.

Note:

As noted in “MISSING value handling” on page 11, some standard #functions did not allow the MISSING value in prior versions of *Fast/Unload*; it is now allowed in almost all cases (except when a non-null string or non-zero number is not allowed). We note this in passing, but it is **not** the type of issue we usually mention as a compatibility issue, since it represents making *Fast/Unload* more robust (a wider range of input values is now meaningfully processed).

SMF record length

To accommodate the new job statistics (as described in “Miscellaneous enhancements” on page 16), the length of SMF records has been increased, although, since the length of each record is self-defining, this should not cause any compatibility issues.

4.2 Fixes in Fast/Unload 4.0 but not in 3.1

This section lists other fixes to features existing in *Fast/Unload* version 3.1 but which, in absence of customer problems, have not, as of the date of the release, been fixed in that version.

4.2.1 Rounding of numeric values

The value of the low order bits in the floating point representation of numeric values has been changed, so that it is now consistent with User Language. This will cause the results of *Fast/Unload* version 4.0 to differ from those of 3.1 in the following situations:

1. Addition and subtraction, if it involves a number greater than or equal to 1,000,000,000,000 (10 to the 15th power) or involves a number with a non-zero fraction part.
2. Conversion from a decimal string to a numeric type, if the number contains more than 15 significant digits or is a fraction with more than 16 digits after the decimal point.

The IBM 360 floating point instructions use an exponent based on a power of two. If the value being represented contains a fractional part, or if it is beyond the range of integers that can be exactly represented, then the floating point representation of a value may be only an approximation of the value. For example, the value “one-tenth” is represented in

base 16 as the infinite hexadecimal series 1AAAAAAA... These approximations can lead to surprising results, especially when dealing with **decimal** fractions that one commonly considers as having an exact representation, e.g., the base 10 number “.1” for the value one-tenth.

To address this problem, the approach to floating point manipulation in User Language and in *Fast/Unload* is based on the following:

Decimal external inputs and outputs

For external numeric inputs and outputs of *Model 204* applications, the original source (for example, data entry fields) and final destination (for example, printed values) is expressed in decimal (base 10) notation.

Arithmetic results mirror decimal operations

When two numeric values are operated upon, in particular with addition and subtraction, the resulting value, when expressed in decimal, is as close as practical to the value that would occur if the operation were performed in decimal.

These principles are accomplished using the following algorithms:

15 digit decimal significance

An 8-byte floating point value, in the IBM 360 architecture, contains 56 bits of significance for the fraction part. 56 bits can represent the numbers from 0 to approximately $7.2E16$ (7.2 times 10 to the 16th power). Therefore, the maximum significance, in decimal, of an 8-byte FLOAT is 16 (decimal) digits. User Language uses a more conservative limit to significance, and uses 15 decimal digit significance with numeric operations.

Conversion from float to decimal

When a floating point value is converted to a string of decimal digits, that string will have a maximum of 15 significant digits. The algorithm first converts a floating point value to a value which is the product of a power of 10 and a factor which has 15 decimal integer places. That latter factor is then rounded to the nearest 15 digit integer. The power of 10 factor is then used to locate the decimal point to express the proper value.

This algorithm was already in place in *Fast/Unload* 3.1.

Conversion from decimal to numeric type

When a string of decimal digits is converted to a numeric type, a floating point representation is used in the conversion, and it is equivalent to the decimal string rounded to a maximum of 15 significant digits. If the decimal string contains 15 significant digits or less (ignoring trailing zeros, and ignoring leading zeros even after the decimal point), it is converted directly to a floating point value. Otherwise, the 16th significant digit is used to round the preceding 15 digits (e.g., $xxxx5$ becomes $xxxx+1$).

This algorithm is available with *Fast/Unload 3.1*, by applying a maintenance zap.

Canonical representation

In very many situations, there are multiple floating point values that represent the same decimal value, when using the above algorithm for conversion to decimal. To achieve consistency, *Model 204* adjusts floating point values to a canonical representation, rounded to 15 decimal digits. That is, if the floating point value **f1** converts to the decimal value **dec** (with float to decimal conversion as above) then the actual value used, **f2**, is the decimal value **dec** converted to floating point (with decimal to float conversion as above). The situation in which **f1** and **f2** are different frequently occurs with floating point addition and subtraction.

This approach is not available in *Fast/Unload 3.1*.

Multiplication and division

These operations are performed by simply invoking the IBM 360 8-byte floating point instructions, and using the result. This consists of adding (or, for division, subtracting) the exponents and multiplying the fractions. The 360 floating point instructions truncate any unused bits in the product, without rounding.

This algorithm was already in place in *Fast/Unload 3.1*.

Addition and subtraction

These operations are performed by invoking the IBM 360 8-byte floating point instructions, and rounding the result to 15 decimal digits.

The 360 floating point addition and subtraction instructions consist of converting the number with the smaller exponent to an equivalent value with exponent matching the larger, and performing the addition or subtraction. The 360 floating point instructions truncate any unused bits in the result, without rounding.

After this operation, the result is rounded to 15 decimal digits, as described above under “Canonical representation”. The 15 decimal digits will include as many leading zeroes as necessary to pad the result value to the number of significant digits as the addend with the larger absolute value. This achieves the objective of “Arithmetic results mirror decimal operations”, described above.

This approach is not available in *Fast/Unload 3.1*.

Following is one example of the “before” and “after” behavior of *Fast/Unload*. As mentioned above, some of the changes to the algorithms (in particular, all of the changes **except** the changes for addition and subtraction) have been issued as maintenance zaps to *Fast/Unload* versions, including 3.1. In this example, since it

depends on the change in addition and subtraction, “before” refers to *Fast/Unload* version 3.1, either with or without the zap maintenance. “After” refers to the behavior of *Fast/Unload* version 4.0.

```
%T2 = 1
FOR J FROM 1 TO 7 /* Get 1E-7
  %T2 = %T2 / 10
END FOR
PUT %T2
OUTPUT
FOR J FROM 1 TO 5 /* Get .01
  %T2 = %T2 * 10
END FOR
PUT %T2
OUTPUT
%T = 0
FOR J FROM 1 TO 10 /* Get .1
  %T = %T + %T2
END FOR
PUT %T
OUTPUT
```

Before results (version 3.1, with or without maintenance):

```
0.00000001
0.009999999999999999
0.09999999999999999
```

After results (version 4.0):

```
0.00000001
0.009999999999999999
0.1
```

4.2.2 #DATE2Nx prior to 1900, #DATE2NS seconds since midnight

Using the #DATE2Nx functions in previous versions of *Fast/Unload*, dates prior to 1900 returned values greater than zero; they should be less than zero, and this is now fixed. Also in those versions, using #DATE2NS, the number of seconds in the part of a day were returned multiplied by a factor of 100; this is now fixed.

4.2.3 #Nx2DATE non-numeric date number

Using the #Nx2DATE functions in previous versions of *Fast/Unload*, non-numeric values of the first argument were taken as 0 (January 1, 1900). They should have been handled as invalid arguments, and this is now fixed.

4.2.4 Null REPORT lines

In previous versions of *Fast/Unload*, a REPORT statement which produces a null string is not produced on the output FUNPRINT file. This is now fixed; null output lines are produced on FUNPRINT.

4.3 Version co-requisites

This section lists any restrictions on usage of various products (including *Fast/Unload* itself) which will be imposed by use of version 4.0 of *Fast/Unload*.

Fast/Reorg When the UAI statement of *Fast/Unload* version 4.0 is used to unload a file for reorganization, version 5.2 or later of *Fast/Reload* must be used to reload the file.

