
Fast/Unload Release Notes Version 3.0

March, 1997



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

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
Chapter 2: New Features	3
Date Processing	3
DATESTAT	3
Converting date fields	3
Date #functions	3
UAI SORT support for 2-digit years	3
%Variables and #Functions in FUEL	4
Manipulating Fields in UAI/PAI	5
Customer-written Assembler #Function Packages	5
Other Enhancements	6
FUEL after EOF	6
ZONED format	6
31-bit FUNOUT buffers	6
New field statistics	6
New parameters and special variables	6
Alert if file state questionable	7
Customize default ERROR and MISSING action/value	7
Customize FUNOUT as default	7
Chapter 3: Documentation Improvements	9
Chapter 4: Compatibility/Fixes	11
Backwards compatibility with Fast/Unload 2.01	11
Fixes in Fast/Unload 3.0 but not in 2.01	12
Version co-requisites	12

CHAPTER 1 ***Introduction***

This document lists the enhancements and other changes contained in the newest release of *Fast/Unload*: version 3.0. The previous generally released version of *Fast/Unload*, 2.01, was released in February, 1996.

CHAPTER 2 *New Features*

2.1 Date Processing

2.1.1 DATESTAT

The DATESTAT statement has been added to FUEL to scan a file, determine which fields have date values, and provide information about them. Fields are checked against up to 60 datetime formats to detect fields containing dates. Either DETAIL or SUMMARY information can be obtained; in either case DATESTAT will indicate the format of the date values and suggest the correct 100-year windows for managing date fields with *Model 204 V4R1* or components of the Sirius Sir2000 product set.

2.1.2 Converting date fields

One application of the NEW, ADD, and CHANGE statements is to manipulate date field values, making it possible to change the format or values of date fields over UAI and PAI reorgs. This includes the ability to specify rules for the handling of 2 digit (“YY”) years. These statements have been added to FUEL in version 3.0.

2.1.3 Date #functions

A set of #functions can be used in FUEL for date arithmetic, checking, and conversion. This includes the ability to specify rules for the handling of 2 digit (“YY”) years. The #functions facility is new in version 3.0.

2.1.4 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 was introduced in a zap for *Fast/Unload 3.0*. Generally, features introduced as a zap for a given version are not mentioned in the release notes of that version; we do so here due to the current importance of communicating date-related issues to our customers.

2.2 %Variables and #Functions in FUEL

New syntax elements have been added to FUEL for dramatically easier and more powerful file manipulation. As with all FUEL operations, the processing of these enhancements is very efficient. These enhancements are:

%Variables %Variables are a new form of entity, which act very much like field occurrences. Every %variable is left unchanged until your program resets it; for example, you can use a %variable to accumulate the total of a field and REPORT it at the end of the file. %Variables can be used in all contexts in which a field occurrence can be used; a %variable can also be used as a field occurrence number or as either of the limits on a FOR ... FROM ... TO statement.

There are no %variable declarations; a %variable can hold any of the basic FUEL types (string, fixed, and float).

Assignment The assignment statement, which consists of a %variable followed by an equal sign followed by an expression, is one way that a variable can get a value.

Arithmetic expression The expression after an equal sign in an assignment statement (or in an ADD or CHANGE statement) can be the result of a calculation on FUEL entities with any combination of the addition, subtraction, multiplication, and division operations.

#Functions The expression after an equal sign in an assignment statement (or in an ADD or CHANGE statement) can be the result of a FUEL function call. The FUEL function names begin with the “sharp” character (“#”); thus we use the word **#functions** to refer to them. There are 17 standard #functions in this release of *Fast/Unload*, in the following areas:

- String manipulation
- Conversion between character and hexadecimal representation
- Date and time manipulation and checking
- The *Model 204* \$SNDX algorithm

In addition to input arguments, FUEL #functions support the notion of missing arguments and output arguments. Much of the checking of the form of #function call is performed at compile time.

There is also a provision to allow customers to create their own assembler language #functions.

2.3 Manipulating Fields in UAI/PAI

The following statements have been added to allow you to modify the field values of the current record with FUEL. These modifications apply to all types of unload, but they are particularly useful in UAI or PAI unloads.

- ADD** This statement allows you to add a new field occurrence. The keyword ADD is followed by a field name, an equals sign, and an expression (with the same form of expression as supported in the assignment statement).
- CHANGE** This statement allows you to modify a field occurrence. The keyword CHANGE is followed by a field name and optional occurrence number, an equals sign, and an expression (with the same form of expression as supported in the assignment statement).
- DELETE** This statement allows you to remove a field occurrence. The keyword DELETE is followed by a field name and optional occurrence number.
- NEW** This FUEL statement must occur before the UAI and FOR EACH RECORD statements. It introduces a new field name into the file; you can use ADD to create occurrences for this field. This can be useful for items like creating derived invisible field values

2.4 Customer-written Assembler #Function Packages

You can extend the set of #functions available at your site by writing one or more collections of them and making them available to FUEL. The items enabling you to do this are:

- The FUNCTIONS statement has been added to FUEL to identify the location of functions you have written.
- An efficient, easy-to-use interface has been designed for matching a function name to the code implementing that function.
- An efficient, easy-to-use interface has been designed for performing common operations needed by #functions, such as obtaining the value of arguments and setting the result value and output arguments.
- The interfaces have been designed with compatibility in mind; with new releases of *Fast/Unload* you will not even need to reassemble your #functions.

2.5 Other Enhancements

A number of other enhancements to *Fast/Unload* have been made and are described in this section.

2.5.1 FUEL after EOF

Statements can be coded after the END FOR statement which closes the FOR EACH RECORD loop. These statements are executed after the end of the input file (after the end of each input file, if the \$FUNLOAD record set is from a *Model 204* group).

2.5.2 ZONED format

ZONED can be specified as a format type in the PUT and UAI SORT statements. This designates the IBM zoned decimal format.

2.5.3 31-bit FUNOUT buffers

If the version of MVS supports the 31-bit BSAM interface, *Fast/Unload* will place the FUNOUT buffers above the 16 megabyte line.

2.5.4 New field statistics

In addition to the maximums already produced by the FSTATS parameter, the following values will now be reported for each field:

1. Minimum number of occurrences
2. Minimum length

This could inform you, for example, if fields are missing which should be present on all records.

2.5.5 New parameters and special variables

A new parameter and associated special variable have been introduced (UPARM and #UPARM, respectively), two group-related special variables have been introduced (#GRPSIZ and #GRPMEM), and a new value for the HARDERR parameter has been introduced (IGNORE0).

- UPARM allows passing a string which can be accessed in the FUEL program as a new special variable (#UPARM).

- The value of #GRPSIZ is the number of files being unloaded; the value of #GRPMEM is the number of the current file being unloaded (1 for the first file, etc.).
- HARDERR=IGNORE0 is like HARDERR=IGNORE, but does not change the job step completion code.

2.5.6 Alert if file state questionable

Fast/Unload now checks the file status before unloading it. A new statement, CHECK, allows you to change what to check, including both file status values and the presence of procedure or invisible field definitions. You can also customize the default checks for your site.

2.5.7 Customize default ERROR and MISSING action/value

You can apply a zap so that your site's default PUT MISSING value (for non-STRING output format) is 0, and a zap so that your site's default PUT ERROR clause is CANCEL.

2.5.8 Customize FUNOUT as default

You can apply a zap so that your site's default value for the FUNOUT parameter is ON; this customized default can be over-ridden by specifying the SORTOUT parameter.

CHAPTER 3 *Documentation Improvements*

Previously undocumented program parameters and details about FUEL are documented. More examples have been added to the documentation. Some error messages have been improved: for example, relating runtime errors to line numbers in the FUEL code, indicating attempts to use unavailable features, and giving more information when an unload terminates prematurely.

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 (2.01).

4.1 Backwards compatibility with Fast/Unload 2.01

This section lists any differences in processing that result from execution with *Fast/Unload* version 3.0, as compared with the same inputs to *Fast/Unload* version 2.01 at current maintenance levels.

Group member error terminates unload

If an error occurs to terminate the unload of a group member, the program will end. Previous versions went on to the next member in the group.

Questionable FISTAT prevents unload

If a file is “broken” (logically or physically) or is in deferred update mode for a UAI OINDEX or UAI INVISIBLE type of unload, the unload job will be terminated. If you do not need to correct the underlying problem in the file, you can override this checking by using the CHECK xxx ALLOW statement.

Syntax check lines past record loop

In previous versions of *Fast/Unload*, any lines in the FUNIN DD after the end of the FOR EACH RECORD loop are ignored. Therefore, if you had any such input lines in old *Fast/Unload* programs, you may either receive compilation errors for them, or, if they are legal FUEL syntax, they will be executed.

ZONED or LENGTH as field names

1. If either of the words ZONED or LENGTH is part of a field name used on the UAI SORT statement, it must be quoted.
2. If the word ZONED is part of a field name used on the PUT statement, it must be quoted.

4.2 Fixes in Fast/Unload 3.0 but not in 2.01

This section lists other fixes to features existing in *Fast/Unload* version 2.01 but which, in absence of customer problems, have not, as of the date of the release, been fixed in that version.

There are no problems without fixes.

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 3.0 of *Fast/Unload*.

There are no such restrictions.