

## What's New in Data-Miner Version 7.1

Data-Miner version 7.1 is currently for z/VSE only. The z/OS version is in development. The z/OS version will be identical to the z/VSE version where permitted by the operating system.

In all the examples below, the following field definitions are used:

KEY	1	8	C	[8 byte character key starting at byte 1 of the record]
NAME	20	16	C	[16 byte character name starting at byte 20]
AMOUNT	50	7	P 2	[7 byte packed number, 2 decimal places, starts at byte 50]

### Commenting

Comments can be entered in a Data-Miner script in many ways, some of which are new in 7.1:

- \* as the first character in a command line (not necessarily in the first position of the line); for example,

```
* UPDATE TOTAL AMOUNT
```

- /\* following the last part of a command that you want Data-Miner to treat as an executable command; for example,

```
ADD 1 TO COUNT /* INCREASE RECORD COUNTER
```

- /\* can appear in the middle of a command and causes the remainder of the line it is on to be treated as a comment; for example,

```
NEWVAL = OLDVAL + SALE /* ADD SALE TO VALUE  
+ TAX /* AND ADD TAX, TOO
```

- \ in the first position of a command causes it to be ignored completely—the command does not print on the script listing. This is a useful way of temporarily removing a command from a script. To ignore a multi-line command, you must put a \ as the first character of each line.
- ? in the first position of a line causes it not to be printed in the script listing, but it is still executed. To suppress printing of a multi-line command, you must put a ? in the first position of each line.

### Support for Larger Numbers

Data-Miner now supports 31-digit packed numbers and binary numbers up to 2,000,000,000. Data-Miner's built-in total fields (T0 to T9) are still limited to 15 digits for backward compatibility with earlier versions.

## ***SORT Command***

Data-Miner can sort a file into any order and using any field types recognised by the customer's sort program. The sort can use multiple keys in any combination of ascending and descending. All fields and files used by the Data-Miner SORT command are defined the same way as any other field and can be used in any other Data-Miner commands executed at the same time as the sort. Data-Miner's syntax is considerably simpler than the syntax used by SORT programs.

### **Examples**

1. Sort the ACCOUNT file in ascending NAME order.

```
SORT ACCOUNT BY NAME
```

2. Sort the ACCOUNT file in ascending NAME and descending AMOUNT order.

```
SORT ACCOUNT BY NAME -AMOUNT
```

3. Sort the ACCOUNT file in ascending NAME order, omitting all the records with a negative AMOUNT.

```
SORT ACCOUNT BY NAME  
SKIP AMOUNT < 0
```

## ***Timing Commands***

Data-Miner allows you to say when you want a command or set of commands to be executed using the timing commands. Timing commands are meant to be used mainly in conjunction with the SORT command but can also be used with any Data-Miner script. They save you setting up switches, creating "first time" code, and so on:-

### **BEFORE INPUT**

Commands in this timing are executed before the main file is read. The most common use is to execute "first time" commands. This may be useful, for example, for reading a control record from a file before processing the main file.

### **DURING INPUT**

Commands in this timing are executed for every record read from the main file. This can be used to accumulate totals, to make decisions on whether to keep or skip records, and so on. DURING INPUT and DURING OUTPUT commands are executed at the same time in a non-SORT script. In a SORT script, DURING INPUT commands are run on the records from the unsorted input file. If you are familiar with SORT exits, this is similar to E15 processing or the COBOL INPUT-PROCEDURE clause.

### **DURING OUTPUT**

Commands in this timing are executed for every record read from the main file. This is the default timing and is used to specify commands to be executed during the output stage of a script. In a SORT script, DURING OUTPUT commands are run on the records

being written to the sorted output file. If you are familiar with SORT exits, this is similar to E35 processing or the COBOL OUTPUT-PROCEDURE clause. If you have deleted a record DURING INPUT, that record is not processed by DURING OUTPUT.

## AFTER OUTPUT

Commands in this timing are executed at the end-of-job. For example, you could print totals, update a control file, and so on.

## The @#\_TIMING System Variable

The system variable @#\_TIMING is not intended for use by users, but you can use it in your script in any command where you could use a 1-byte character field. The five values currently assigned to @#\_TIMING are

B = Before input  
I = During input  
O = During output  
A = After output  
E = End of job

It is very highly recommended that you do not attempt to change @#\_TIMING because the results are unpredictable but probably unpleasant. In a non-SORT script, @#\_TIMING never has the value *I*.

## Scope of timing commands

The default timing is “DURING OUTPUT,” meaning that if you do not enter a timing command in your script or if one or more commands are entered before the first timing command, the command is executed during the output stage.

A timing command stays in effect until the next timing command or until the end of the script is encountered.

You can enter more than one timing command of any type—you may want to do that to improve readability.

You cannot enter a timing command in the middle of an IF group or in the middle of a PROC.

## Examples of Timing Commands

1. Read a record from a control file before sorting a file.

```
SORT ACCOUNT BY NAME
BEFORE INPUT
READ CTLFILE KEY 'HEADER'
DURING OUTPUT
  Remainder of commands
```

- Sort the ACCOUNT file, leaving out any records with a negative amount. Display the total AMOUNT at the end of the job.

```
SORT ACCOUNT BY NAME
DURING INPUT
SKIP    AMOUNT < 0
DURING OUTPUT
ADD AMOUNT TO TOTAMT
AFTER OUTPUT
DISPLAY `TOTAL AMOUNT IS ` TOTAMT
```

- Same as example 2 but all done DURING OUTPUT (which is less efficient because all the records we don't want take part in the sort). Note that the SKIP command tells Data-Miner not to write the record or perform arithmetic on it—it does NOT tell it to ignore the record in logical commands (IF, SKIP, ONLY).

```
SORT ACCOUNT BY NAME
  * THE NEXT COMMAND IS NOT REALLY NEEDED AS IT IS THE
  * DEFAULT TIMING
DURING OUTPUT
SKIP AMOUNT <0
  ADD AMOUNT TO TOTAMT
AFTER OUTPUT
DISPLAY `TOTAL AMOUNT IS ` TOTAMT
```

- Copy the file INACCT to OUTACCT. Skip all the records for negative amounts or with a name of OBAMA. Display the total AMOUNT from the OBAMA records. (There are many ways to do this.) Note that all the “DURING OUTPUT” commands are unnecessary but have been put in to show that you can put them in if you like.

```
COPY
INPUT=VSAM  FILENAME=INACCT
* FIELD DEFINITIONS GO HERE
OUTPUT=VSAM FILENAME=OUTACCT DURING OUTPUT
IF NAME = `OBAMA`
  ADD AMOUNT TO T1
ENDIF
DURING OUTPUT
SKIP    AMOUNT<0 OR NAME=`OBAMA`
AFTER OUTPUT
DISPLAY `TOTAL OBAMA AMOUNT IS ` T1
```

## **TRACE Command**

TRACE lets you print trace messages to show what commands your script has executed. TRACE can be turned on and off during the execution of your script, allowing you, for example, to trace only for record 12345678.

The format of the TRACE command is

```
TRACE { ON | OFF | FLOW }
```

There is no default value for the operand—you must specify a value.

TRACE ON turns on all tracing—every command you execute will produce a trace line.

TRACE OFF turns tracing off until another TRACE ON or TRACE FLOW is encountered or for the rest of the job

TRACE FLOW turns on tracing only for those commands that affect the logic flow through your script such as IF, GOTO and PERFORM

Note that ENDIF is never actually executed and so will not be traced. Also, the command shown in the trace may not be the command that you wrote if Data-Miner has decided to translate your command to one that it considers better. This may happen if you enter a command in a non-Data-Miner language that Data-Miner recognizes.

## **TRACE Example**

In this example, statement 13 turns on tracing. Note that line 14 displays in the trace as a SET command because Data-Miner changes all ADD, SUBTRACT, etc., instructions to arithmetic expressions.

```
13 TRACE ON
14 ADD 1 TO JAMES
15 IF K2 = '4' GOTO TYPE4
16 ENDIF
17 DISPLAY JAMES @#_TIMING
/* Other statements follow

DMIN-506 >>> STATEMENT AT START OF SCRIPT ABOUT TO EXECUTE
DMIN-505 >>> STATEMENT      13 TRACE ABOUT TO EXECUTE
DMIN-505 >>> STATEMENT      14 SET ABOUT TO EXECUTE
DMIN-505 >>> STATEMENT      15 IF ABOUT TO EXECUTE
DMIN-505 >>> STATEMENT      17 DISPLAY ABOUT TO EXECUTE
      52      O
DMIN-506 >>> STATEMENT AT END OF SCRIPT ABOUT TO EXECUTE
```

## **Notes**

1. The Add in line 14 displays in the trace as a SET command because Data-Miner changes all ADD, SUBTRACT, etc., instructions to arithmetic expressions.

2. The ENDIF does not display in the trace because Data-Miner does not execute any code for an ENDIF.
3. The data displayed by statement 17 (“52            0”) is displayed on the same print file as the trace because TRACE and DISPLAY both use the system print file for their output.

### ***Pre-Start Procedure***

A new parameter, BEGIN, is available on the AUTO command to name a PROC that is to be performed before the first record of the main input file is read. If there is no main input file then this procedure is run at the start of the script. This procedure is performed only once and so, if you want it to execute some commands several times, you must set up your own loop.

### **Example**

This command tells Data-Miner to read the ACCOUNT file automatically and to perform the DOFIRST procedure before it starts processing ACCOUNT.

```
AUTO INPUT ACCOUNT BEGIN DOFIRST
*   THE REST OF THE SCRIPT GOES HERE
PROC DOFIRST
* Start-up commands go here - read a control file etc.
ENDPROC
```

### ***End-of-File Procedure***

A new parameter, EOF, is available on the AUTO command to name a PROC that is to be performed when end of file is reached on the main input file. This procedure is performed only once, and so if you want it to execute some commands several times, you must set up your own loop.

### **Example**

This command tells Data-Miner to read the ACCOUNT file automatically and to perform the ENDIT procedure at end of file.

```
AUTO INPUT ACCOUNT EOF ENDIT
*   THE REST OF THE SCRIPT GOES HERE
PROC ENDIT
* END OF FILE COMMANDS GO HERE - DISPLAY TOTALS ETC
ENDPROC
```

**Note:** The AUTO command does not need either an EOF or BEGIN parameter. Both or either can be specified and they can be specified in either order.

## ***Difference between BEGIN / EOF and BEFORE / AFTER Timing Commands***

There is no difference at all between these commands—they merely give you a choice of how to do a job. Of the two, BEFORE INPUT is probably quicker and easier to write than a BEGIN PROC, but the choice is yours.

## ***Support for More Field Types***

The VAR field type allows the use of variable length character fields of up to 32,767 bytes. Data-Miner automatically adjusts the length of the field when a new value is placed in the field.

## ***STATS Command & Statistics***

STATS tells Data-Miner where you want the execution statistics printed , if at all.

The format is

```
STATS [ Printer | Console | Both | None ]
```

Only the first character of the parameter need be entered. “Printer” turns off “Console” and vice versa.

File IO statistics displayed at end of job now include extra information such as the file id from the DLBL or DD statement (for example, ACCOUNT) and the file name (for example, XYZCO.ACCOUNTS.MASTER). Statistics are broken down by the kind of IO done for each file. For example,

```
35 RECORDS PRINTED ON PRINTER
35 RECORDS DUMPED ON PRINTER
35 RECORDS ADDED TO OUTFILE JM.FIXBLK.OUTPUT
40 RECORDS READ FROM INFILE DMNR.KSDS.R10K
0 FIELD VALIDATION ERRORS
```

## ***CSI-SORT Support (z/VSE Only)***

CSI-SORT uses Data-Miner's I/O drivers. Enhanced BIM-EPIC support improves tape handling in Data-Miner and CSI-SORT.

The Data-Miner I/O drivers no longer perform an expiry code check when used by CSI-SORT.

## ***Changes in Execution Order***

Your existing scripts will continue to run their commands in the order that they run now except that INSERT commands are now executed when they are encountered rather than being saved for the end of the script. If you are running a script that loops automatically (for example, a COPY or a script with an AUTO command at the start), you may want to ensure that your INSERT commands come after a “BEFORE INPUT” or “AFTER OUTPUT” timing command to ensure that they do not get executed repeatedly.

## ***German Support***

Most of the Data-Miner commands can now be written in German. German and English can be freely mixed. German commands supported include

ABZIEHEN	BERICHT	MULTIPLIZIEREN
ADDIEREN	DEFINIEREN	PROZEDUR
ANDERS	DIVIDIEREN	PROZEDURENDE
AUSFUEHREN	DRUCKEN	VERSCHIEBEN
AUSFUEHRENNENDE	EINGABE=	WAEHREND
AUSGABE=	ENDE	WENN
AUSWAEHLEN	ENDEWENN	

## ***VSAM Performance***

VSAM performance has been improved, especially for ESDS output

## ***CPU Performance***

CPU utilization has been improved in most commands

## ***Maintenance***

All PTFs from earlier versions have been incorporated in the source code of version 7.1.