Runtime Modules

Release 8.1.3

November 2013

IKAN Solutions N.V.
Kardinaal Mercierplein 2
B-2800 Mechelen
BELGIUM

Copyright © 2013, IKAN Solutions N.V.

No part of this document may be reproduced or transmitted in any form or by any means, electronically or mechanically, for any purpose, without the express written permission of IKAN Solutions N.V.

MetaSuite, MetaStore Manager, MetaMap Manager and Generator Manager are trademarks of IKAN Solutions N.V.

Table of Contents

CHAPTER 1
About This Manual

Runtime modules are subroutines that will be called at the execution of the main program. If any change is necessary on a subroutine the main program hasn't to be recompiled.

1.1. Related Publications

The following table gives an overview of the complete MetaSuite documentation set.

Release Information	Release Notes 8.1.3
Installation Guides	• BS2000/OSD Runtime Component • DOS/VSE Runtime Component • Fujitsu Windows Runtime Component • MicroFocus Windows Runtime Component • MicroFocus UNIX Runtime Component • OS/390 and Z/OS Runtime Component • OS/400 Runtime Component • VisualAge Windows Runtime Component • VisualAge UNIX Runtime Component • VMS Runtime Component
User Guides	• INI Manager User Guide • Installation and Setup Guide • Introduction Guide • MetaStore Manager User Guide • MetaMap Manager User Guide • Generator Manager User Guide
Technical Guides	• ADABAS File Access Guide • IDMS File Access Guide • IMS DLI File Access Guide • RDBMS File Access Guide • XML File Access Guide • Runtime Modules • User-defined Functions User Guide

If you are unfamiliar with MetaSuite, the following technical description provides you with a brief overview.

The MetaSuite System

MetaSuite is designed for data retrieval, extraction, conversion and reporting. It includes a workstation-based graphical user interface and a mainframe runtime component.

MetaSuite Database Interfaces	MetaSuite can access data from a number of database management systems, using the same commands, program structure and retrieval techniques used for non-database files. Each database interface is available as an optional enhancement to the base product.
MetaMap Manager	MetaMap Manager is the MetaSuite tool used to define models. Such models are intuitively built by describing overall program specifications, input file definitions (data and process) and target file definitions (data and process).
MetaStore Manager	MetaStore Manager is a tool that provides metadata maintenance and documentation services.
Generator Manager	The Generator Manager is the system administration tool.All kinds of basic functionalities and customization possibilities are supported by this tool.

CHAPTER 2

MetaSuite Runtime Modules

All modules handled by this manual are called "Runtime Modules". MetaSuite Runtime Modules are modules called by the generated MetaSuite programs. They are called during runtime. IKAN Solutions delivers the sources, and those sources must be compiled and linked before use

All runtime modules have a name beginning with "MS" and ending with a version number of 3 digits. Example: MSIMS815. Since this manual contains a general overview of the functionalities, the version numbers in the program names have been omitted.

2.1. MSA2U The SYS-ASCII-UNICODE functionality

This module performs a conversion from an ASCII-based character field into a Unicode-based character sequence This module is called when the SYS-ASCII-UNICODE function is used. The program contains a technical information block.

The same module is also used to perform the conversion from UTF-8 to UTF-16 setting the codepage to 1208. She is part of the SYS-UTF8-ASCII functionality.

2.2. MSASC The SYS-ASCII functionality

This module performs a conversion from an EBCDIC based character field into an ASCII based character sequence. This module is called when the SYS-ASCII function is used. The program contains a technical information block.

2.3. MSBIN The SYS-BINARY functionality

This module performs a conversion from a hexadecimal sequence into a character field. This module is called when the SYS-BINARY function is used. The program contains an information block about the usage.

2.4. MSCAL The calculation module

This module calculates the value of a string expression. If the expression is not numeric, the SYS-NOT-NUMERIC flag will be set.

This module is called for interpreting string expressions in delimited files, in XML files, and the SYS-NUMVALIDATE function uses this runtime module as well. The program contains a technical information block.

2.5. MSCAU The calculation module (Unicode)

The Unicode variant of MSCAL.

2.6. MSCCX Concatenation runtime module

This module performs string concatenations. This module is called for executing the AND function.

Parameters

Only one parameter:

```
01  LV-CC-PARMS.
    02 LV-CC-COUNT PIC S9(4) BINARY .
    02 LVT-LENGTH-TABLE.
    03 LVT-LENGTH OCCURS 16 TIMES
           PIC S9(4) BINARY .
    03 FILLER PIC X(4096).
```

2.7. MSCIX Runtime parameter I/O module

This module opens, reads and closes the file with link-name "PPTIPT". This file is used for transferring runtime parameters to the main program.

Parameters

```
01  CIX-PARMS.
    02 CIX-FUNCTION PIC S9(4) BINARY.
    02 CIX-STATUS PIC S9(4) BINARY.
    02 CIX-TEXT.
       03 FILLER PIC X(255).
```

CIX-FUNCTION

```
GET = 0,
OPEN = 1,
CLOSE = 3,
HOLD = 8. (do nothing)
```

CIX-STATUS

```
UNOPENED = 0.
OPENED = 1
CLOSED = 2.
EOF = 4.
```

CIX-TEXT: result of the GET function.

Comment lines and empty lines will be skipped.

2.8. MSDBX De-blanker runtime module

Numeric data has both leading and trailing blanks stripped. Alphanumeric has trailing blanks stripped, and the resulting strings are concatenated in the user supplied buffer.

Parameters

```
01  DBX-OUTBUF.
    02 DBX-OUT-BYTE OCCURS 32767
                    INDEXED BY DB-OUT-IDX.
       03 FILLER PIC X.
01  DBX-OUTLEN PIC S9(5) BINARY.
01  DBX-OTYPE.
    02 DBX-OTYPE-RED PIC X.
    02 DBX-SEPARATOR PIC X.
01  DBX-STRCT PIC S9(5) BINARY.
01  DBX-S01.
    02 FILLER PIC S9(5) BINARY.
    02 FILLER PIC X(4096).
01  DBX-S02.
    02 FILLER PIC S9(5) BINARY.
    02 FILLER PIC X(4096).
01  DBX-S03.
    02 FILLER PIC S9(5) BINARY.
    02 FILLER PIC X(4096).
```

2.9. MSDBU De-blanker runtime module (Unicode)

The Unicode variant of MSDBX.

2.10. MSDCX Date arithmetic runtime module

Program to calculate the difference between 2 DATE fields, and to add or subtract days from a certain date.

Parameters

```
01  DCX-PARMS.
    02 DCX-FUNCTION PIC S9(8) BINARY.
    02 DCX-JDATE PIC S9(8) BINARY.
    02 DCX-DORD PIC S9(8) BINARY.
    02 DCX-RESULT PIC S9(8) BINARY.
```

DCX-FUNCTION

1 = ADD days to J-DATE

2 = SUBTRACT days to J-DATE

3 = Difference in days between DORD and JDATE.
JDATE is assumed to be more recent than DORD.

DCX-JDATE

Format: YYYYDDD with YYYY=year, DDD=julian day.

DCX-DORD

For functions 1 and 2	DORD must contain a number of days
For function 3	DORD must contain a date in the form YYYYDDD.

DCX-RESULT

After using functions 1 and 2	RESULT contains a date in the form YYYYDDD.
After using function 3	RESULT contains the number of days.

2.11. MSDCU Date arithmetic runtime module (Unicode)

The Unicode variant of MSDCX.

2.12. MSDMX SQLCODE handling module

This module gets the message text from an SQLCODE. If the SQLCODE should cause the program to stop, the message text (SQLWARN1,2,3…) will be displayed before stopping.

Variants of this module have been developed for other database systems.

`MSDMX`	DB2 (z/OS & OS-390, VMS)
`MSD2X`	DB2/2 (Windows)
`MSD4X`	DB2/400 (AS/400 & OS/400)
`MSD6X`	DB2/6000 (RS/6000 UNIX)

MSDVX	DOS/VSE
MSFIJ	ODBC/Fujitsu/Windows
MSIFX	INFORMIX
MSIGX	INGRES
MSODB	ODBC/Micro Focus/Windows
MSORX	ORACLE
MSRDB	Relational database in general
MSSEX	SESAM BS2000
MSSQX	SQL server
MSSYX	SYBASE
MSTRX	Teradata

Parameters

```
01  SYS-X.
    03 SYS-SQL-VERB PIC S9(4) BINARY.
    03 SYS-HALT-ALL PIC S9(4) BINARY.
    03 SYS-SQL-AUTO PIC S9(4) BINARY.
    03 SYS-RETURN-CODE PIC S9(9) BINARY.
    03 USR-RETURN-CODE PIC S9(9) BINARY.
    03 SYS-RESTART PIC X(8).
01  SQLCA SYNC.
    05 SQLCAID PIC X(8).
    05 SQLCABC PIC S9(9) BINARY.
    05 SQLCODE PIC S9(9) BINARY.
    05 SQLERRM.
       49 SQLERRML PIC S9(4) BINARY.
       49 SQLERRMC PIC X(70).
    05 SQLERRP PIC X(8).
    05 SQLERRD OCCURS 6 TIMES PIC S9(9) BINARY.
    05 SQLWARN.
       10 SQLWARN0 PIC X.
       10 SQLWARN1 PIC X.
       10 SQLWARN2 PIC X.
       10 SQLWARN3 PIC X.
       10 SQLWARN4 PIC X.
       10 SQLWARN5 PIC X.
       10 SQLWARN6 PIC X.
       10 SQLWARN7 PIC X.
       10 SQLWARN8 PIC X.
       10 SQLWARN9 PIC X.
       10 SQLWARNA PIC X.
    05 SQLSTATE PIC X(5).
```

2.13. MSDTX Date conversion runtime module

This module converts dates from one format to another.

Parameters

```
 01  DTX-PARMS.
* Input Date
     02 DTX-IX PIC X(10).
     02 DTX-I9 REDEFINES DTX-IX PIC 9(10).
* Output Date
     02 DTX-OX PIC X(10).
     02 DTX-O9 REDEFINES DTX-OX PIC 9(10).
* Input Date Type
     02 DTX-ITYPE PIC 9(2).
* Output Date Type
     02 DTX-OTYPE PIC 9(2).
     02 DTX-FMT PIC XXX.
* return Status
* -4 = SYS-INVALID-DATE
* -1 = SYS-NOT-NUMERIC
* 0 = OK
     02 DTX-S PIC S9(4) BINARY.
* Date default century
     02 DTX-CEN PIC S9(4) BINARY.
* Date century end
     02 DTX-END PIC S9(4) BINARY.
```

Type of date

DT9-MMDDYY 01	DT9-DDMMYY 02	DT9-MMDDYYYY 03
DT9-DDMMYYYY 04	DT9-YYDDD 05	DT9-YYYYDDD 06
DT9-YYMMDD 07	DT9-YYDDMM 08	DT9-YYYYMMDD 09
DT9-YYYYDDMM 10	DTS-MMDDYY 11	DTS-DDMMYY 12
DTS-MMDDYYYY 13	DTS-DDMMYYYY 14	DTS-YYDDD 15
DTS-YYYYDDD 16	DTS-YYMMDD 17	DTS-YYDDMM 18
DTS-YYYYMMDD 19	DTS-YYYYDDMM 20	DTX-MMDDYY 21
DTX-DDMMYY 22	DTX-MMDDYYYY 23	DTX-DDMMYYYY 24
DTX-YYDDD 25	DTX-YYYYDDD 26	DTX-YYMMDD 27
DTX-YYDDMM 28	DTX-YYYYMMDD 29	DTX-YYYYDDMM 30

The DT9 type of dates are zoned numeric, ten digits long. Alignment is done to the right. The DTX type of dates have the same structure as the DT9 type of dates, but the alignment is done to the left (alphanumeric alignment). The DTS type of dates are also alphanumeric and they have a separator.

2.14. MSDTU Date conversion runtime module (Unicode)

The Unicode variant of MSDTX.

2.15. MSE2U The SYS-EBCDIC-UNICODE functionality

This module performs a conversion from an EBCDIC-based character field into a Unicode-based character sequence. This module is called when the SYS-EBCDIC-UNICODE function is used. The program contains a technical information block.

2.16. MSEBC The SYS-EBCDIC functionality

This module performs a conversion from an ASCII based character field into an EBCDIC based character sequence. This module is called when the SYS-EBCDIC function is used. The program contains a technical information block.

2.17. MSENX File concatenation

This program is used in Windows and UNIX environments for simulating the OS/390 DD-concatenation. Sourcefile-names can be put one after another, delimited by semicolons.

Example: PPTF01 = fileid01dat;fileid02.dat;fileid03.dat

This sourcefile-string is passed to MSENX and this program returns the next sourcefile-name. At the end, the program-variable "FILES-NBR" becomes zero.

2.18. MSEOJ End of Job - module

This program can be customized for post-program processing. You should set the MTL option OPTION-EXPOSED-VARIABLES to "ON", else this module will not be called.

2.19. MSHEX The SYS-HEXADECIMAL functionality

This module performs a conversion from a character field into a hexadecimal sequence. This module is called when the SYS-HEXADECIMAL function is used. The program contains a technical information block.

2.20. MSHFX Hexadecimal formatter runtime module

Subroutine that produces printable hex from a passed character string. calls assembler subroutine "mshxx.." for hex to clear-text conversion, and "mspfx..." for printing. Note that program does not use run-time message formatter.

2.21. MSHXT Hexadecimal formatter runtime module

Run-time module to convert a string to horizontal hexadecimal dump format without translation.

2.22. MSHXX Hexadecimal formatter runtime module

Run-time module to convert a string to a horizontal hexadecimal dump format and then translates the input to printable characters.

2.23. MSIMS Inquiry of the IMS PCB

The IMS application interface block routine. This module does an inquire of the IMS program control block

The CEETDLI and PLITDLI interfaces keep track of calls to and returns from IMS. If you use CBLTDLI or ASMTDLI from a COBOL program and you are using IMS Version 4 Release 1 or IMS Version 3 Release 1 with PTF UN49280, the coordination between Language Environment and IMS condition handling is the same as that provided by CEETDLI.

MSIMS makes use of ASMTDLI, the IMS assembler module.

Parameters

```
01  F00-IMS-PCBNAME PIC X(8).
01  F00-IMS-STATUS-CODE PIC XX.
01  F00-IMS-PCB POINTER.
```

2.24. MSINI Start of Job - module

This program can be customized for pre-program processing. You should set the MTL option OPTION-EXPOSED-VARIABLES to "ON", else this module will not be called.

2.25. MSLID PPTLID formatter module

This module reads the file with link-name "PPTLID" and puts the first record of that file into the only parameter.

Parameters

```
01  LV-LID-REC.
    02 LID PIC X(255).
```

2.26. MSLOG Moves all the PPTLOG information into a relational table structure.

This executable program (not a module!) writes messages from PPTLOG to a relational database. PPTLOG is the logical name for the logging file that is produced when running a script that was generated by MetaSuite. PPTLOG contains about the same information as PPTLST, in a less readable form for humans, but in a more readable form for computers.

The executable program MSLOG can be used in post-processing.

The SQL instructions to create the tables that are fed by this program, can be found on the installation CD in the "MetaStore" folder. (pptlog.sql) Several log files can be concatenated (i.e. put together, one after the other) as input file for PPTLOG.

For more information about the PPTLOG content, please refer to Description of the PPTLOG Records (page 17). For more information about the LOG database tables, please refer to Description of the LOG database tables (page 23).

2.27. MSLSX Runtime messages formatter module

This module writes messages on PPTLOG. Every action that has to triggered, has to call MSLSX.

PPTLOG contains plus minus the same information as PPTLST, in a less readable form for humans, but in a more readable form for computers.

PPTLOG is used for Post-processing reasons (statistics, error-treatment,...)

MSLOG (see above) is a program that reads PPTLOG and writes the content in relational database tables.

For more information about the PPTLOG content, please refer to Description of the PPTLOG Records (page 17).

2.28. MSPFX Runtime messages I/O module

The PPTLST (internal name = APDLST) file is made by this module.

Parameters

```
01  PFX-PARMS.
    02 PFX-FUNCTION PIC S9(4) BINARY.
    02 PFX-STATUS PIC S9(4) BINARY.
    02 PFX-INT PIC S9(4) BINARY.
    02 PFX-PRINT.
       03 PFX-PRINT80 PIC X(80).
       03 FILLER PIC X(52).
    02 PFX-ID PIC X(9).
```

Same functionalities as MSCIX.

Functions: Open, Close, Write. Status: Opened, Closed, Unopened, …

2.29. MSRNX Random number generator runtime module

This module generates a random number. You'll have to provide a random-seed. This can be a time-stamp or something else.

```
01  RANDPARM.
    02 RANDSEED PIC S9(8) BINARY.
    02 RANDVAL PIC S9(8) BINARY.
```

The second time, the random-seed will be changed.

2.30. MSRST Restart generator

This module simulates the IMS restartability module, but can be used in order to maintain restartability.

2.31. MSSED Sesam - Date editor

Sesam database. This module converts binary dates into display format.

2.32. MSSES Sesam - Date editor

Sesam database. This module converts timestamps to format YYYYMMDDHHMMSSNNN.

2.33. MSSET Sesam - Time editor

Sesam database. Time convert to HHMMSSNNN displayable.

2.34. MSSPX Runtime parameter formatter module

Reads the PPTIPT file and does the interpretation of the PPTIPT records. Moves the runtime parameter values into the proper zones. Makes use of MSCIX and MSLSX.

2.35. MSSZX Sample size runtime module

Sample data module.

2.36. MSTDX Date and time runtime module

Moves current date to the parameter.

Parameters

```
01  TDX-PARMS PIC X(21).
```

Format

```
01  Intrinsic-Date.
    03 New-Date Pic 9(8) Value Zeroes.
    03 New-Time Pic 9(8) Value Zeroes.
    03 Filler Pic X(5).
```

2.37. MSTDU Date and time runtime module (Unicode)

The Unicode variant of MSTDX.

2.38. MSTMX Temporary name routine

This function should be replaced by a C- tmpname() call.

2.39. MSTRM The SYS-TRIM functionality

This module trims the leading spaces from a string. This module is called when the SYS-TRIM function is used. The program contains a technical information block.

2.40. MSTRU The SYS-TRIM functionality (Unicode)

The Unicode variant of MSTRM.

2.41. MSU2A The SYS-UNICODE-ASCII functionality

This module performs a conversion from a Unicode-based character sequence into an ASCII-based character field. This module is called when the SYS-UNICODE-ASCII function is used. The program contains a technical information block.

2.42. MSU2E The SYS-UNICODE-EBCDIC functionality

This module performs a conversion from a Unicode-based character sequence into an EBCDIC-based character field. This module is called when the SYS-UNICODE-EBCDIC function is used. The program contains a technical information block

2.43. MSXDP Hexadecimal Debug functionality

This module is used by the DEBUG statement in order to issue an hexadecimal dump of a National field.

APPENDIX A
PPTLST

A.1. Description of the PPTLST file

Both PPTLST and PPTLOG files contain information about the processed source and target files. The PPTLST file however, is written in a more convenient form for the customer, and the PPTLOG file is easier to handle for computers.

1. The first part of the PPTLST file, the PROGRAM INFORMATION SECTION consists of generator information: runtime version, time that the program was written.

```
------------------------------------------------------------
---            PROGRAM INFORMATION SECTION                 ---
------------------------------------------------------------

MetaSuite Runtime 07.02.02
Program : P3265R   Version : 0001
                   Written : 2007-01-22 16:15:59
```

2. The RUNTIME MESSAGES SECTION contains different kinds of information that was produced during the run, for instance:
   - Records with invalid content are excluded by the system. Those records are listed here by means of a hexadecimal dump. The reason of the exclusion is listed also. In the next appendix we will give more information about how to interprete a dump.
   - Runtime parameters can be set by means of the command line or the PPTIPT file. Those settings are confirmed by a message in this listing.
   - This part also contains the values that have been entered by the MANUAL-INPUT functionality.

```
------------------------------------------------------------
---              RUNTIME MESSAGES SECTION                  ---
------------------------------------------------------------

E    NON-NUMERIC DATA IN FIELD       EMPLOYEE-NUMBER
     OF INPUT FILE                   BOS-DB2_Department_Employee
     Within Source      Record Number          1
     POSITION WITHIN RECORD                    1
E    DATE FORMAT INVALID IN FIELD    DATE-OF-HIRE
     OF INPUT FILE                   BOS-DB2_Department_Employee
     Within Source      Record Number          1
     VALUE AT TIME OF ERROR              999999

POSITION          HEX DUMP OF RECORD IN ERROR
                             1                   2          1          2
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 12345678901234567890
   00001 58585858585831585839393939393958585858 XXXXXX1XX999999XXXXX
   00021 58585858585858585858585858585858585858 XXXXXXXXXXXXXXXXXXXX
   00041  TO 00200 SAME AS ABOVE                  XXXXXXXXXXXXXXXXXXXX
   00201 58585858                                XXXX

All runtime parameter values were retrieved
from the restart buffer.
```

3. The third part is optional. In case of a program interruption and restart, it contains RESTART counters, for instance:
   - the number of source and target records before previous run was interrupted
   - the number of exclusions before previous run was interrupted
   - the number of produced paths before previous run was interrupted
   - the I/O error message

```
------------------------------------------------------------
---            RESTART INFORMATION SECTION                 ---
------------------------------------------------------------

SOURCE FILE INFORMATION
----------------------
SourceFile                          EMP-DB2_Department_Employee
 Version                            00000
 Error Code                         0000000000
 INPUT READ                                 40
 INPUT PATHS PROCESSED                      40
Record                              EMP-EMPLOYEE_PLUS
 INPUT READ                                  5

REPORT FILE INFORMATION
----------------------
ReportFile                          T01-RESULT
 Version                            00000
 Error Code                         0000000000
 READ                                       40
 PROCESSED                                  24
 EXCLUDED BY USER RULE                      16
 DETAILS PRODUCED                           24
```

4. The fourth part contains INPUT-OUTPUT counters, for instance:
   - the number of source and target records
   - the number of exclusions
   - the number of produced paths
   - the I/O error messages

```
------------------------------------------------------------
---          INPUT-OUTPUT INFORMATION SECTION              ---
------------------------------------------------------------

SOURCE FILE INFORMATION
----------------------
SourceFile                          EMP-DB2_Department_Employee
 Version                            00000
 Error Code
 INPUT READ                                 50
 INPUT PATHS PROCESSED                      50
Record                              EMP-EMPLOYEE_PLUS
 INPUT READ                                  6

REPORT FILE INFORMATION
----------------------
ReportFile                          T01-RESULT
 Version                            00000
 Error Code
 READ                                       50
 PROCESSED                                  32
 EXCLUDED BY USER RULE                      18
 DETAILS PRODUCED                           32
```

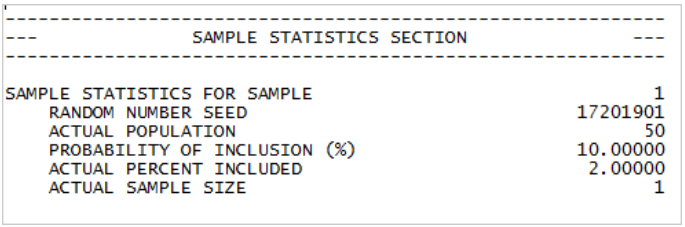
Important remark: In case of a restart run, the restart data is restored using the restart buffer. The file and record counters are restored to the value that they had before the program break. If the program breaks twice, then the next restart buffer data will contain higher values than the first restart buffer data.

Finally, the INPUT-OUTPUT section should contain **at least** the number of records that is read and written by a normal run. In most cases the result will be identical, but we can not guarantee that the last

record before the break won't be treated twice. More information about the technical background can be found in the *MetaMap Manager User Guide*, in the "RESTART" chapter.

5. The Sample statistics section contains the number of records that are included and excluded due to the usage of the SAMPLE command. The sampling parameters and the consequences of this sampling are described in this section.

   More information about these parameters can be found in the *MetaMap Manager User Guide* (search for the SAMPLE statement).



```
------------------------------------------------------------
---              SAMPLE STATISTICS SECTION                 ---
------------------------------------------------------------

SAMPLE STATISTICS FOR SAMPLE                              1
    RANDOM NUMBER SEED                             17201901
    ACTUAL POPULATION                                    50
    PROBABILITY OF INCLUSION (%)                  10.00000
    ACTUAL PERCENT INCLUDED                        2.00000
    ACTUAL SAMPLE SIZE                                    1
```

6. The RUNTIME INFORMATION SECTION is written at the end of the PPTLST file. This section contains:
   - The Program User Exit Status. This is the error number that was assigned to SYS-RETURN-CODE by the user.
   - The Program System Exit Status, which is the error number that was assigned to SYS-RETURN-CODE by the system. (I/O error number, for instance)
   - Message. This message explains the Program System Exit Status.
   - Start Run, which is the time stamp of the program load.
   - End Run, which is the time stamp of the program exit.
   - Run Time, which is the difference between the two previous time stamps.

```
------------------------------------------------------------
---              RUNTIME INFORMATION SECTION               ---
------------------------------------------------------------

Program user exit status                    0
Program system exit status                  0
  Meaning : No system message
Start Run : 2007-01-22 16:16:11
End Run   : 2007-01-22 16:16:14
Run-time  :          0 00:00:03
```

APPENDIX B

Description of the PPTLOG Records

Every PPTLOG record starts with an identifier of 2 characters. This identifier can have the following values :

IDENTIFIER	Description of the record
"V "	Generator version information – normal run.
"VR"	Generator version information – restart run.
"ID"	Program identity information.
"SF"	Sourcefile information.
"SR"	Sourcerecord information.
"TF"	Targetfile information.
"RC"	Returncode information.

The MDL file containing a brief description can be found in the `LOGDATA.MDL` file in the MDL directory. This definition might be very useful. Below you will find a sample PPTLOG file.

```
Version record :
ID/Version
Example : V 08.01.03
Program identity information :
|Program start time |Name |Vnr|Generate time|... ID2002030712060398+0100EX0
000120020307120531...
...Log Identifier 1 |Log Identifier 2
... Exercise 0
Sourcefile information :
IDTnn| SourceFile Name |IRecRCnt|...
SFI01----employee-master 000000000...
...ErrExcl-|UserExcl|RecPrCnt|IIBC----|IIBEX---|IIBUX---|...
...000000000000000000000000000000000000000000000000000000...
...ISWC----|IPRC----|IPEX----|IPUX----|IPBC----|IPBEX---|...
...000000000000000053000000003000000000000000000000000000...
...IPBUX---|InpProc-|OutWrCnt|NumErr--|LimitErr|SubscErr|...
...000000000000000050000000000000000006000000000000000000...
...DateErr-|CompErr-|
...000000003000000000
Sourcerecord information :
Tnn| SourceRecord Name |RecPC |RecUX |
SR 01----EMPLOYEE-DATA 000000050000000000
```

B.1. The Version Record

This record contains information about the runtime system that produced the records.

The first parameter, the "Record Type", can have the value "V" or "VR". The first value means that the following records are records from a normal run. The "VR" value means that the records that will follow contain data from the restart buffer. The restart buffer is dumped first before the "normal run" data are gathered…

The second parameter, the runtime version, can be found on position three with a length of eight characters.

Record example:

```
----+----1
VR07.02.02
```

MDL description:

```
ADD RECORD LV-V-REC SIZE 10
ADD FIELD V-RT POSITION 1 SIZE 1 TYPE CHARACTER RULE Record Type
ADD FIELD V-VERSION POSITION 3 SIZE 8 TYPE CHARACTER RULE Runtime Version
```

B.2. The Program Identity Information record

This record contains program specific information, like program name, the version number, the time that the program was generated, and the time that the program was started. Some user information can be stored in the log identifiers. The program can fill log identifier 1; log identifier 2 contains the program description.

Record example:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----
8----+----9----+--
|Program start time |Name |Vnr|Generate time|Log Identifier 1 |Log Identifier 2
ID2002030712060398+0100EX0 000120020307120531 Exercise 0
```

MDL description:

```
ADD RECORD LV-ID-REC SIZE 113
ADD FIELD ID-RT POSITION 1 SIZE 2 TYPE CHARACTER RULE Record Type
ADD FIELD ID-DATE POSITION 3 SIZE 21 TYPE CHARACTER RULE Program Run Date
ADD FIELD ID-PGM POSITION 24 SIZE 8 TYPE CHARACTER RULE Program Name
ADD FIELD ID-VER POSITION 32 SIZE 4 TYPE CHARACTER RULE Program Version
ADD FIELD ID-WRITTEN POSITION 36 SIZE 14 TYPE CHARACTER RULE Program Version
ADD FIELD ID-LID1 POSITION 50 SIZE 32 TYPE CHARACTER RULE Log Identifier - Part 1
ADD FIELD ID-LID2 POSITION 82 SIZE 32 TYPE CHARACTER RULE Log Identifier - Part 2
```

B.3. The "Source file Information" record

The "Source file Information" record contains the source file name and lots of file specific counters like read counters, errors counters, user exclusion counters, processed counters… both for initial processing as for input processing

Record example:

```
----+----1----+----2----+----3----+----4----+----5----+----6
IDTnn| SourceFile Name |IRecRCnt|…
SFI01----employee-master 000000000…
5----+----6----+----7----+----8----+----9----+----0----+----1
…ErrExcl-|UserExcl|RecPrCnt|IIBC----|IIBEX---|IIBUX---|…
…000000000000000000000000000000000000000000000000000000000…
-+----1----+----2----+----3----+----4----+----5----+----6
…ISWC----|IPRC----|IPEX----|IPUX----|IPBC----|IPBEX---|…
…000000000000000053000000003000000000000000000000000000000…
--6----+----7----+----8----+----9----+----0----+----1----+
…IPBUX---|InpProc-|OutWrCnt|NumErr--|LimitErr|SubscErr|…
…000000000000000050000000000000000006000000000000000000…
---+----2----+----3
…DateErr-|CompErr-|
…000000003000000000
```

MDL description:

```
ADD RECORD LV-SF-REC SIZE 230
ADD FIELD SF-RT POSITION 1 SIZE 2 TYPE CHARACTER RULE Record Type
ADD FIELD SF-TYPE POSITION 3 SIZE 1 TYPE CHARACTER RULE SourceFile processing type
ADD FIELD SF-FN POSITION 4 SIZE 2 TYPE ZONED UNSIGNED RULE SourceFile Number
ADD FIELD SF-FNAME POSITION 6 SIZE 36 TYPE CHARACTER RULE SourceFile Name
ADD FIELD SF-IIRC POSITION 42 SIZE 9 TYPE ZONED UNSIGNED RULE Initial records Read
Count
ADD FIELD SF-IIEX POSITION 51 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Error Exclu-
sions
ADD FIELD SF-IIUX POSITION 60 SIZE 9 TYPE ZONED UNSIGNED RULE Initial User Exclu-
sions
ADD FIELD SF-IIPC POSITION 69 SIZE 9 TYPE ZONED UNSIGNED RULE Initial record Pro-
cessed Count
ADD FIELD SF-IIBC POSITION 78 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Buffers con-
structed Count
ADD FIELD SF-IIBEX POSITION 87 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Buffers
Error Exclusions
ADD FIELD SF-IIBUX POSITION 96 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Buffer Ex-
cluded
ADD FIELD SF-ISWC POSITION 105 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Extract
Record Write Count
ADD FIELD SF-IPRC POSITION 114 SIZE 9 TYPE ZONED UNSIGNED RULE InPut record Read
Count
ADD FIELD SF-IPEX POSITION 123 SIZE 9 TYPE ZONED UNSIGNED RULE Input Error Exclu-
sions
ADD FIELD SF-IPUX POSITION 132 SIZE 9 TYPE ZONED UNSIGNED RULE Input User Exclusions
```

```
ADD FIELD SF-IPBC POSITION 141 SIZE 9 TYPE ZONED UNSIGNED RULE Input Buffers con-
structed Count
ADD FIELD SF-IPBEX POSITION 150 SIZE 9 TYPE ZONED UNSIGNED RULE Input Buffer Error
Exclusions
ADD FIELD SF-IPBUX POSITION 159 SIZE 9 TYPE ZONED UNSIGNED RULE Input Buffer User
Exclusions
ADD FIELD SF-IPPC POSITION 168 SIZE 9 TYPE ZONED UNSIGNED RULE Input Processed Count
ADD FIELD SF-ORWC POSITION 177 SIZE 9 TYPE ZONED UNSIGNED RULE Initial Output Record
Written Count
ADD FIELD SF-NEC POSITION 186 SIZE 9 TYPE ZONED UNSIGNED RULE Numeric Errors
ADD FIELD SF-LEC POSITION 195 SIZE 9 TYPE ZONED UNSIGNED RULE Limit Errors
ADD FIELD SF-SEC POSITION 204 SIZE 9 TYPE ZONED UNSIGNED RULE Subscript Errors
ADD FIELD SF-DEC POSITION 213 SIZE 9 TYPE ZONED UNSIGNED RULE Date Errors
ADD FIELD SF-CEC POSITION 222 SIZE 9 TYPE ZONED UNSIGNED RULE Computation Errors
```

B.4. The "Source record information" record

The" Source record information" record contains both for INIT as for INPUT processing the number of processed records and the number of excluded records by the user.

Record example:

```
----+----1----+----2----+----3----+----4----+----5----+----6
Tnn| SourceRecord Name |RecPC |RecUX |
SR 01----EMPLOYEE-DATA 000000050000000000
```

MDL description:

```
ADD RECORD LV-SR-REC SIZE 77
ADD FIELD SR-RT POSITION 1 SIZE 2 TYPE CHARACTER RULE Record Type
ADD FIELD SR-TYPE POSITION 3 SIZE 1 TYPE CHARACTER RULE SourceFile processing type
ADD FIELD SR-FN POSITION 4 SIZE 2 TYPE ZONED UNSIGNED RULE SourceFile Number
ADD FIELD SR-RECN POSITION 6 SIZE 36 TYPE CHARACTER
ADD FIELD SR-INIT-PC POSITION 42 SIZE 9 TYPE ZONED RULE Init Procedure Processed
record Count
ADD FIELD SR-INIT-UX POSITION 51 SIZE 9 TYPE ZONED RULE Init Procedure User Exclu-
sions
ADD FIELD SR-INPT-PC POSITION 60 SIZE 9 TYPE ZONED RULE Input Procedure Processed
record Count
ADD FIELD SR-INPT-UX POSITION 69 SIZE 9 TYPE ZONED RULE Input Procedure User Ex-
clusions
```

B.5. The "Target File information" record

The" Target file information" record contains the number and name, and various counters both for INPUT as for OUTPUT processing, SORT and REPORT counters, and the different type of system errors (overflows, date errors, computational errors, subscript errors, limit errors).

Record example:

```
----+----1----+----2----+----3----+----4----+----5----+----6
Tnn| TargetFile Name |Read----|Process-|…
TFT01T01-datawarehouse 000000050000000050…
-6----+----7----+----8----+----9----+----0----+----1----+
…ErrExcl-|UserExcl|SortWrit|ReporDet|ReporTot|LimitErr|…
…00000000000000000000000000000000005000000000000000000…
--+----2----+----3----+----4----+----5----+----6
…SubscErr|DateErr-|CompErr |OFlowTot|
…000000000000000000000000000000000000
```

MDL description:

```
ADD RECORD LV-TO-REC SIZE 149
ADD FIELD TO-RT POSITION 1 SIZE 2 TYPE CHARACTER RULE Record Type
ADD FIELD TO-TYPE POSITION 3 SIZE 1 TYPE CHARACTER RULE TargetFile Type
ADD FIELD TO-TN POSITION 4 SIZE 2 TYPE ZONED UNSIGNED RULE TargetFile Number
ADD FIELD TO-TNAME POSITION 6 SIZE 36 TYPE CHARACTER RULE TargetFile Name
ADD FIELD TO-IPRC POSITION 42 SIZE 9 TYPE ZONED UNSIGNED RULE Input Read record
Count
ADD FIELD TO-IPPC POSITION 51 SIZE 9 TYPE ZONED UNSIGNED RULE Input Processed record
Count
ADD FIELD TO-IPEX POSITION 60 SIZE 9 TYPE ZONED UNSIGNED RULE Input Error Exclusions
ADD FIELD TO-IPUX POSITION 69 SIZE 9 TYPE ZONED UNSIGNED RULE Input User Exclusions
ADD FIELD TO-OSWC POSITION 78 SIZE 9 TYPE ZONED UNSIGNED RULE Output Sort records
Written Count
ADD FIELD TO-ORDC POSITION 87 SIZE 9 TYPE ZONED UNSIGNED RULE Output Report Details
written Count
ADD FIELD TO-ORTC POSITION 96 SIZE 9 TYPE ZONED UNSIGNED RULE Output Report Totals
written Count
ADD FIELD TO-LEC POSITION 105 SIZE 9 TYPE ZONED UNSIGNED RULE Limit Errors
ADD FIELD TO-SEC POSITION 114 SIZE 9 TYPE ZONED UNSIGNED RULE Subscript Errors
ADD FIELD TO-DEC POSITION 123 SIZE 9 TYPE ZONED UNSIGNED RULE Date Errors
ADD FIELD TO-CEC POSITION 132 SIZE 9 TYPE ZONED UNSIGNED RULE Computation Errors
ADD FIELD TO-OEC POSITION 141 SIZE 9 TYPE ZONED UNSIGNED RULE Overflow Error Count
on totals
```

B.6. The "Return code information" record

The "Return code information" record contains user and system return code, and a time stamp at the time that the file was closed.

Record example:

```
----+----1----+----2----+----3----+----4----+----5----+----6
Returncode User System Timestamp program end
RC 0 0|20020307120604l3+0100
```

MDL description:

```
ADD RECORD LV-EX-REC SIZE 43
ADD FIELD EX-RT POSITION 1 SIZE 2 TYPE CHARACTER RULE Record Type
ADD FIELD EX-RCU POSITION 3 SIZE 10 TYPE CHARACTER RULE user return code
ADD FIELD EX-RCS POSITION 13 SIZE 10 TYPE CHARACTER RULE System return code
ADD FIELD EX-DATE POSITION 23 SIZE 21 TYPE CHARACTER RULE Program Run End Date
```

APPENDIX C

Description of the LOG database tables

C.1. The "RPROGRAM" table

Field Name	Data Type	Description
ID_NUMB	Number	Identification number. Every program run, this number increases by one
PAR_NUMB	Number	Record number within the current program log
ID_RTVER	Text	MetaSuite generator version
ID_PGM	Text	Program name
ID_VER	Number	Program version
ID_DATE	Date/Time	Program start time
ID_WRITTEN	Date/Time	Program generation time
EX_RCU	Text	User return code
EX_RCS	Text	System return code
EX_DATE	Date/Time	Program stop time
ID_LID1	Text	SYS-APPLICATION-GROUP value
ID_LID2	Text	SYS-APPLICATION-NAME value

C.2. The "RSRCFILE" table

Field Name	Data Type	Description
ID_NUMB	Number	Identification number. Every program run, this number increases by one
PAR_NUMB	Number	Record number within the current program log
SF_TYPE	Text	Sourcefile processing type
SF_FN	Text	Source file number
SF_PREF	Text	Source file prefix
SF_NAME	Text	Source file name
SF_IIRC	Number	Initial records read count
SF_IIEX	Number	Initial error exclusions
SF_IIUX	Number	Initial user exclusions
SF_IIPC	Number	Initial record processed count
SF_IIBC	Number	Initial buffers constructed count
SF_IIBEX	Number	Initial buffers error exclusions
SF_IIBUX	Number	Initial buffer excluded
SF_ISWC	Number	Initial extract record write count
SF_IPRC	Number	Input record read count
SF_IPEX	Number	Input error exclusions
SF_IPUX	Number	Input user exclusions
SF_IPBC	Number	Input buffers constructed count
SF_IPBEX	Number	Input buffer error exclusions
SF_IPBUX	Number	Input buffer user exclusions
SF_IPPC	Number	Input processed count
SF_ORWC	Number	Initial output record written count
SF_NEC	Number	Numeric error count
SF_LEC	Number	Limits error count
SF_SEC	Number	Subscripts error count
SF_DEC	Number	Date error count
SF_CEC	Number	Computational error count

C.3. The "RSRCRECORD" table

Field Name	Data Type	Description
ID_NUMB	Number	Identification number. Every program run, this number increases by one
PAR_NUMB	Number	Record number within the current program log
SR_TYPE	Text	Sourcefile processing type
SR_FN	Text	Source file number
SR_PREF	Text	Source file prefix
SR_NAME	Text	Source record name
SR_INI_PC	Number	Initial processed count
SR_INI_UX	Number	Initial user exclusions
SR_IPT_PC	Number	Input processed count
SR_IPT_UX	Number	Input user exclusions

C.4. The "RTGTFILE" table

Field Name	Data Type	Description
ID_NUMB	Number	Identification number. Every program run, this number increases by one
PAR_NUMB	Number	Record number within the current program log
TO_TYPE	Text	Targetfile type
TO_TN	Text	Targetfile number
TO_PREF	Text	Targetfile prefix
TO_NAME	Text	Targetfile name
TO_IPRC	Number	Input read record count
TO_IPPC	Number	Input processed record count
TO_IPEX	Number	Input error exclusions
TO_IPUX	Number	Input user exclusions
TO_OSWC	Number	Output sort records written count
TO_ORDC	Number	Output report details written count
TO_ORTC	Number	Output report totals written count
TO_LEC	Number	Output limit errors
TO_SEC	Number	Output subscript errors
TO_DEC	Number	Output date errors
TO_CEC	Number	Output computation errors
TO_OEC	Number	Output overflow error count on totals

How to interprete a DUMP

```
E   NON-NUMERIC DATA IN FIELD        EMPLOYEE-NUMBER
    OF INPUT FILE                    ----employee-master
    Within Source      Record Number          1
    POSITION WITHIN RECORD                    1
E   NON-NUMERIC DATA IN FIELD        ANNUAL-SALARY
    OF INPUT FILE                    ----employee-master
    Within Source      Record Number          1
    POSITION WITHIN RECORD                   16
E   DATE FORMAT INVALID IN FIELD     DATE-OF-HIRE
    OF INPUT FILE                    ----employee-master
    Within Source      Record Number          1
    VALUE AT TIME OF ERROR              999999
POSITION          HEX DUMP OF RECORD IN ERROR
                             1                   2          1          2
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 12345678901234567890
   00001 58585858585831585839393939393958585858 XXXXXX1XX999999XXXXX
   00021 58585858585858585858585858585858585858 XXXXXXXXXXXXXXXXXXXX
   00041  TO 00200 SAME AS ABOVE                  XXXXXXXXXXXXXXXXXXXX
   00201 58585858                                 XXXX
```

D.1. Step 1

Print out the PPTLST file with the DUMP

```
E    NON-NUMERIC DATA IN FIELD       EMPLOYEE-NUMBER
     OF INPUT FILE                   ----employee-master
     Within Source      Record Number          1
     POSITION WITHIN RECORD                    1
E    NON-NUMERIC DATA IN FIELD       ANNUAL-SALARY
     OF INPUT FILE                   ----employee-master
     Within Source      Record Number          1
     POSITION WITHIN RECORD                   16
E    DATE FORMAT INVALID IN FIELD    DATE-OF-HIRE
     OF INPUT FILE                   ----employee-master
     Within Source      Record Number          1
     VALUE AT TIME OF ERROR              999999
POSITION          HEX DUMP OF RECORD IN ERROR
                             1                   2          1          2
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 12345678901234567890
   00001 58585858585831585839393939393958585858 XXXXXX1XX999999XXXXX
   00021 58585858585858585858585858585858585858 XXXXXXXXXXXXXXXXXXXX
   00041  TO 00200 SAME AS ABOVE                  XXXXXXXXXXXXXXXXXXXX
   00201 58585858                                 XXXX
```

D.2. Step 2

Print out the MDL files that correspond to the file names in the DUMP (marked in violet).
In this particular case, we will have to print out the MDL definition of the employee-master file.

```
ADD FILE employee-master VERSION 38 TYPE SEQUENTIAL FIXED 204 BLOCK 204 LABEL STANDARD MODE ASCII
ADD RECORD EMPLOYEE-DATA#39934 SIZE 204 RULE Employee record containing sensitive personal data
ADD FIELD EMPLOYEE-NUMBER#39935 POSITION 1 SIZE 5 TYPE ZONED
ADD FIELD DEPARTMENT#39936 POSITION 6 SIZE 1 TYPE ZONED UNSIGNED RULE Working department
ADD FIELD PAY-CODE#39937 POSITION 7 SIZE 1 TYPE CHARACTER RULE paycode 1 = weekly 2 = monthly
ADD FIELD JOB-TITLE-CODE#39938 POSITION 8 SIZE 2 TYPE ZONED UNSIGNED
ADD FIELD DATE-OF-HIRE#39939 POSITION 10 SIZE 6 TYPE ZONED UNSIGNED DATE 'MMDDYY'
ADD FIELD ANNUAL-SALARY#39940 POSITION 16 SIZE 9 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD PAY-RATE#39941 POSITION 25 SIZE 9 TYPE ZONED DECIMAL 2 UNSIGNED
ADD FIELD EMPLOYEE-NAME#39942 POSITION 34 SIZE 15 TYPE CHARACTER
ADD FIELD STREET-ADDRESS#39943 POSITION 49 SIZE 20 TYPE CHARACTER
ADD FIELD CITY-ADDRESS#39944 POSITION 69 SIZE 15 TYPE CHARACTER
ADD FIELD STATE-CODE#39945 POSITION 84 SIZE 2 TYPE CHARACTER
ADD FIELD ZIP-CODE#39946 POSITION 86 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD SOCIAL-SECURITY-NUMBER#39947 POSITION 91 SIZE 9 TYPE ZONED UNSIGNED
ADD FIELD SECURITY-CLEARANCE-CODE#39948 POSITION 100 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD VOLUNTARY-DEDUCTIONS#39949 POSITION 105 SIZE 10 TYPE CHARACTER OCCURS 10
ADD FIELD VOL-CODE#39951 OF VOLUNTARY-DEDUCTIONS#39949 POSITION 1 SIZE 1 TYPE CHARACTER
ADD FIELD VOL-AMOUNT#39952 OF VOLUNTARY-DEDUCTIONS#39949 POSITION 2 SIZE 9 TYPE ZONED DECIMAL 2
```

D.3. Step 3

Determine the record.

In the DUMP, you certainly have noticed the line

```
Within Source      Record Number          1
```

This means that the first record of your input file contains invalid data.

Important note: This is the first physical record in the input file, and NOT the first logical record definition in the mdl file.

D.4. Step 4

Determine the field and the kind of error.

The field name and the kind of error can be found in the lines that start with the letter "E"

```
E   NON-NUMERIC DATA IN FIELD       EMPLOYEE-NUMBER
E   NON-NUMERIC DATA IN FIELD       ANNUAL-SALARY
E   DATE FORMAT INVALID IN FIELD    DATE-OF-HIRE
```

D.5. Step 5

Investigate the field properties in the MDL file.

If you have the field name, you can derive the field type, the position and the length of the field using the MDL definition.

```
ADD FIELD EMPLOYEE-NUMBER#39935 POSITION 1 SIZE 5 TYPE ZONED
ADD FIELD DATE-OF-HIRE#39939 POSITION 10 SIZE 6 TYPE ZONED UNSIGNED DATE 'MMDDYY'
ADD FIELD ANNUAL-SALARY#39940 POSITION 16 SIZE 9 TYPE ZONED DECIMAL 2 UNSIGNED
```

D.6. Step 6

Verify the value of the field in the record dump.

As an example we consider the field ANNUAL-SALARY. This field is 9 bytes long (SIZE 9), type ZONED, with two digits behind the decimal point. As you can see in the MDL file and in the DUMP lines, this field starts on position 16.

```
POSITION          HEX DUMP OF RECORD IN ERROR
                             1                   2           1         2
          1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 12345678901234567890
   00001 5858585858583158583939393939395858585858 XXXXXX1XX999999XXXXX
   00021 5858585858585858585858585858585858585858 XXXXXXXXXXXXXXXXXXX
   00041  TO 00200 SAME AS ABOVE                   XXXXXXXXXXXXXXXXXXX
   00201 58585858                                  XXXX
```

In the DUMP file, the record content at the time of the DUMP, has been written twice. On the left side of the page, the hexadecimal content of the record was written, and on the right side of the page, the character content was written.

Position 16 contains the first byte of nine, position 17 contains the second byte, and so on. Position 16 is filled with the hexadecimal value "58", corresponding to the letter "X".

The bytes that correspond to this field are marked in RED.

On a mainframe, the hexadecimal value has to be translated using the EBCDIC code table. On Unix or Windows the hex value has to be interpreted with the ASCII code table.

You can clearly see that the value "XXXXXXXXX" is not a numeric value. The message "NON-NUMERIC DATA IN FIELD ANNUAL-SALARY" is correct…

APPENDIX E
Unicode

E.1. What is Unicode

Unicode is a standard for representing characters as integers.

Unlike ASCII, which uses 8 bits for each character, Unicode uses 16 bits, which means that it can represent more than 65,000 unique characters. This is a bit of overkill for English and European languages, but it is necessary for some other languages, such as Chinese and Japanese.

Detail:

- Industry standard for coded character set - defined by Unicode Consortium and ISO
- Covers all commonly used characters in the world in one code page (vs. one "language" per ASCII, EBCDIC, or EUC code page)
- Characters: text, digits, special characters, symbols, control characters, ...
- Multiple Unicode encoding formats: UTF-8, UTF-16, UTF-32
- "Stateless" encoding: meaning of an encoding unit is self defining

E.2. UTF-8 and UTF-16 Encoding

- UTF-8 encoding unit: one byte (the byte-oriented form of unicode)
  - character : 1 to 4 encoding units
- UTF-16 encoding unit: two bytes
  - character: 1 or 2 encoding units
  - All characters defined in commonly used EBCDIC and ASCII code pages represented in one encoding unit
- Characters in Latin-1 (Western European) ASCII code page represented consistently
  - "A" = X"41" in UTF-8 or ASCII,
  - "A" = X"0041" in UTF-16

E.3. Unicode data type

- USAGE NATIONAL, Picture character N
- One UTF-16 encoding unit (2 bytes) per PICTURE N character
- "Character" defined in terms of PICTURE symbol positions, for reference modification, character counts, etc.

Example Copy Book COBOL

```
01  UNICO-REC.
    03 KLN-NR  PIC N(07) USAGE NATIONAL.
    03 FON-NM  PIC N(06) USAGE NATIONAL.
    03 EGN-KD  PIC N(1)  USAGE NATIONAL.
    03 VOOR-NM PIC N(1)  USAGE NATIONAL.
    03 ROEP-NM PIC N(1)  USAGE NATIONAL.
    03 MUT-DT  PIC X(10).
    03 MUT-AT  PIC S9(9) COMP.
```

Representation in MDL

```
DELETE FILE unicod
ADD FILE unicod VERSION 1 TYPE SEQUENTIAL FIXED 46 BLOCK 46 LABEL STANDARD
ADD RECORD UNICO-REC SIZE 46
ADD FIELD KLN-NR  POSITION 1  SIZE 14 TYPE CHARACTER
ADD FIELD FON-NM  POSITION 15 SIZE 12 TYPE CHARACTER
ADD FIELD EGN-KD  POSITION 27 SIZE  2 TYPE CHARACTER
ADD FIELD VOOR-NM POSITION 29 SIZE  2 TYPE CHARACTER
ADD FIELD ROEP-NM POSITION 31 SIZE  2 TYPE CHARACTER
ADD FIELD MUT-DT  POSITION 33 SIZE 10 TYPE CHARACTER
ADD FIELD MUT-AT  POSITION 43 SIZE  4 TYPE BINARY
```

E.4. Unicode Conversion

Intrinsic Conversion Functions

- FUNCTION DISPLAY-OF (national-data [ccsid])
  Returns alphanumeric (EBCDIC) representation of NATIONAL argument
- FUNCTION NATIONAL-OF (ebcdic-data [ccsid])
  Returns UTF-16 representation of EBCDIC (DISPLAY or DISPLAY-1) argument.

If ccsid is omitted, it defaults to the value from the CODEPAGE() compiler option. ccsid may represent an EBCDIC, ASCII, EUC or UTF-8 code page

Note: Use only one EBCDIC code page in a program.

Converting from Unicode

```
01 Unicode-Data pic N(20) usage national.
01 EBCDIC-Data pic X(20).
01 Greek-Data pic X(20).
01 UTF8-Data pic X(20).
01 Japanese-Data pic G(20) Usage Display-1.
```

Example:

1. `Move function Display-of (Unicode-Data) to EBCDIC-DATA`

   Converts UTF-16 (CCSID 1200) to EBCDIC-data represented in CCSID in effect

2. `Move function Display-of (Unicode-Data, 4971) to Greek-Data`

   Converts UTF-16 to EBCDIC Greek (CCSID 4971)

3. `Move function Display-of (Unicode-Data, 1208) to UTF8-Data`

   Converts UTF-16 to UTF-8 (CCSID 1208)

4. `Move function Display-of (Unicode-Data, 1399) to Japanese-Data`

   Converts UTF-16 to EBCDIC Japanese (CCSID 1399)

Converting to Unicode

```
01 Unicode-Data pic N(20) usage national.
01 EBCDIC-Data pic X(20).
01 Greek-Data pic X(20).
01 UTF8-Data pic X(20).
01 Japanese-Data pic G(20) usage display-1.
```

Example:

1. `Move function National-of(EBCDIC-Data) to Unicode-data`

   Converts EBCDIC-data represented in CCSID in effect via CODEPAGE

2. `Move function National-of (Greek-Data, 4971) to Unicode-Data`

   Converts EBCDIC Greek (CCSID 4971)data to UTF-16

3. `Converts UTF-8 (CCSID 1208)data to UTF-16`

   Converts UTF-8 (CCSID 1208)data to UTF-16

4. `Converts EBCDIC Japanese (CCSID 1399)data to UTF-16`

   Converts EBCDIC Japanese (CCSID 1399)data to UTF-16

E.5. Functions in MetaMap

The following functions are available in the Options group of MetaMap. These functions will call one or 2 modules to perform the necessary conversion.

Function	Module 1	Module 2
SYS-ASCII-UNICODE	MSA2U***	-
SYS-ASCII-UTF8	MSA2U*** Ascii to UTF16	MSU2A*** (1208) UTF16 to UTF8
SYS-EBCDIC-UNICODE	MSE2U***	-
SYS-EBCDIC-UTF8	MSE2U*** Ebcdic to UTF16	MSU2E*** (1208) UTF16 to UTF8
SYS-UNICODE-ASCII	MSU2A***	-
SYS-UNICODE-EBCDIC	MSU2E***	-
SYS-UTF8-ASCII	MSA2U*** (1208) UTF8 to UTF16	MSU2A*** UTF16 to Ascii
SYS-UTF8-EBCDIC	MSE2U*** (1208) UTF8 to UTF16	MSU2E*** UTF16 to Ebcdic