

Version

4.2.1

Treehouse Software, Inc.

ADAMAGIC User Guide



A Product of CCA Software Pty Ltd

Copyright Notice

Copyright 1994 - 2011 CCA Software Pty Ltd.

All rights Reserved.

Trademark Acknowledgements

ADABAS and NATURAL are trademarks of SOFTWARE AG of Germany and North America. Solaris is a product of Sun Microsystems and HP-UX is a product of Hewlett Packard Corporation of the USA. Any other trademarks referred to herein are the property of their respective owners.

Requirements for Confidentiality

This document contains trade secrets and proprietary information of CCA Software Pty Ltd. Reproduction and/or modification of this document without the written approval of CCA Software Pty Ltd. is prohibited. Use of this document is limited to licensed users of ADAMAGIC or those given specific written permission of CCA Software Pty Ltd.

THE SOFTWARE WHICH IS DESCRIBED IN THIS DOCUMENT IS SUBJECT TO LIMITATIONS ON USE, RELEASE, DISCLOSURE AND DUPLICATION AND TO REQUIREMENTS FOR CONFIDENTIALITY, PROTECTION AND SECURITY WHICH ARE SET OUT IN THE SOFTWARE LICENSE AND MAINTENANCE AGREEMENTS.

Treehouse Software, Inc.
2605 Nicholson Road, Suite 230
Sewickley, PA 15143 USA
Phone: 724.759.7070
Fax: 724.759.7067
e-mail: tsi@treehouse.com
<http://www.treehouse.com>

Version 4.2.1 rev 355, 3rd February 2011
This is the production release.

Table of Contents

INTRODUCTION.....	1	FIELD AND OFIELD	33
FUNCTIONAL OVERVIEW.....	1	TEST.....	35
VERSION COMPATIBILITY.....	4	<i>Literals</i>	37
PROCESSING OVERVIEW.....	5	NTEST.....	40
ADAMAGIC COMMAND LINE	5	RULE.....	41
ADAMAGIC INPUT SOURCES.....	5	LENGTH	41
ADAMAGIC OUTPUT SOURCES	5	TYPE	42
PARAMETER AND ENVIRONMENT		<i>Data conversion Rules and Restrictions</i>	
VARIABLE SUMMARY	7	42
INPUT DATA.....	8	<i>D type</i>	44
OUTPUT DATA	9	INDEX.....	44
COMPRESSED OUTPUT FILE PROCESSING		FORMAT.....	45
CONSIDERATIONS.....	10	LIMIT	45
RESTART CONSIDERATIONS	12	STARTAT	46
OPERATIONS	13	ARCHIVE.....	46
INPUT AND CONTROL PARAMETERS	13	DISCLAIMER.....	47
CONTROL PARAMETERS.....	14	CREATE ADABAS FILES	49
GENERAL PARAMETER SYNTAX.....	15	ADAMAGIC EXITS.....	51
MAGDUMP.....	16	INSTALLATION OF USER EXITS.....	52
ASSONN.....	16	USAGE CONSIDERATIONS	53
DATANN.....	17	EXAMPLES	55
MAGDEVICE.....	17	FILE TRANSFER	55
MAGTYPE	18	COMPRESSED OUTPUT	56
MAGPROCESS	18	EXTRACT OUTPUT.....	56
MAGMODEL.....	19		
MAGSCRIPT.....	19		
MAGDTA	19		
MAGDVT	20		
MAGFDU	20		
FILE	20		
NEWFILE.....	21		
DBID.....	21		
SKIPREC	22		
NUMREC	22		
DLLDIR.....	22		
PARAMETER EXAMPLES.....	25		
EXTRACT AND/OR REFORMAT			
COMMANDS	27		
GENERAL COMMAND SYNTAX.....	27		
RECORD AND FIELD SELECTION			
COMMAND SUMMARY	28		
SELECTION COMMANDS	28		
NAME.....	28		
FILE	29		
FTYPE	29		
NORMALISE	30		

Introduction

Functional Overview

ADAMAGIC reads mainframe (MVS, MSP or VSE) ADASAV datasets, UNIX/WINDOWS ADABCK backup (cartridge(s) or disk files) or a UNIX/WINDOWS ADABAS database to extract data from one or more selected ADABAS files. Comprehensive selection criteria enable output files to contain almost any desired subset of dumped records. If required, more than one extract can be taken from any ADABAS file (using different selection/extraction criteria). Regardless of how much output is requested, ADAMAGIC makes only one pass of the input data.

Output files created by ADAMAGIC may be in several formats.

1. **Format 1:** is compressed UNIX ADABAS form, formatted as DTA (compressed data) and DVT (compressed descriptor values) files as written by ADACMP. Corresponding to each pair of DTA and DVT files, an ASCII text file is written containing input parameters suitable for the ADAFDU utility. ADAMAGIC can optionally be run to simply extract the ADAFDU information without extracting any actual data, or to extract FDU and DTA but not DVT.
2. **Format 2:** is “flat” file format. That is, the data will be decompressed and written to one or more physical sequential files. The output can have either fixed or variable length records. There are no separators between fields in a record.
3. **Format 3:** is .CSV format. That is, the fields in each record will be decompressed and delimiter separated suitable for input to a program such as Microsoft Excel.

Files produced in format 1 contain all fields from the input data. The only controls over which records are written are that you may skip some records at the start of an ADABAS file and stop writing after a specified number of records have been output. The default is to copy all input records to the output files.

The records and fields written to format 2 and format 3 files are controlled by comprehensive selection criteria, which enable the output files to contain almost

INTRODUCTION

any desired subset of the input records. If required, more than one extract can be taken from any ADABAS file (using different selection/extraction criteria). The extracted data can be the entire decompressed record, suitable for reloading using ADACMP. Alternatively, any subset of the fields in each record can be nominated for extraction. Allowing the possibility of length overrides and field occurrence limits provides additional flexibility. If field occurrence limits are provided then it is easier to process PE and MU fields in the extract using languages such as NATURAL or COBOL. Alternatively, ADAMAGIC can normalize the output to eliminate PE/MU occurrences by building multiple output records. This feature is very useful for loading SQL-based databases, such as DB2 or ORACLE. The output files are sequential with either fixed or variable length records. ADAMAGIC control parameters use a simple syntax, and report any errors clearly.

ADAMAGIC itself imposes no theoretical limit to the number of format 2 and format 3 files that can be created. However, if you request too many extract files your system may not have enough memory or allow enough open file descriptors to cope with your request.

Version 4 of ADAMAGIC will run in a UNIX environment or a Microsoft Windows /2000/2003/XP/VISTA environment. In this document, unless specifically stated otherwise, the information applies to both the UNIX and WINDOWS environments.

ADAMAGIC is useful to the UNIX or WINDOWS ADABAS Database Administrator for the following tasks:

- Convert mainframe ADABAS databases to UNIX;
- Easily access historical data;
- Populate test databases;
- Archive old data;
- Extract large amounts of data cheaply for overnight batch processing;
- Data export - where extracts of defined sets of data are required for input to another DBMS, SAS, or transmission to remote sites;
- Sequential processes - i.e. wherever extensive batch processing of ADABAS files is performed and contention with online users is to be avoided;
- Data extract - where the ability to rerun or referential integrity are required without disturbing online operations;

- Periodic summary reports - e.g. monthly or quarterly summaries are to be produced which may require reruns, e.g. cross-tabulation reports;
- Improved performance - if cost of processing is an important consideration, ADAMAGIC, with its efficient sequential processing and parallel extracts, offers optimal performance characteristics;
- Ad hoc analysis and data mining activities - across historical or current database data;
- File reorganizations - where files are required for input into a file reorganization utility such as CCA's ADAREORG.

Optionally, a UNIX script or NT batch file can be invoked synchronously at the end of each output file to perform additional processing, such as to execute ADAFDU and ADAMUP utilities.

ADAMAGIC uses standard input parameters similar to other ADABAS utilities, which can be input via the console if ADAMAGIC is run in foreground.

A parameter is needed for each file to be selected. If required, the file number that is output can be different from the file number in the input.

Under UNIX, normally ADAMAGIC will be executed from a script, with parameter input redirected to come from the script itself, and console output redirected to a file. The normal way of executing ADAMAGIC under NT would be from a batch file with parameter input coming from the file named by the PARMFILE environment variable. Console output also may be redirected to a file. If the ADASAV input is on tape, mount messages will be sent to **stderr**, so you should not redirect **stderr** if mount messages are to be actioned.

A special compressed "model" input file is needed if you are producing format 1 output files. This is simply any compressed DTA file from your UNIX or NT database (typically devoid of data). The sole purpose of this file is to enable ADAMAGIC to obtain version information from the header record.

ADAMAGIC can be used to convert all application data files from an EBCDIC ADABAS database to equivalent files in a UNIX or NT ASCII ADABAS database. Note that you would not normally convert system files.

WARNING: If your input data is from an MVS database and contains floating point (G format) fields, these will be converted to UNIX or WINDOWS format. However if they are not normalized MVS floating-point fields the conversion result will be unpredictable, in a future version the output fields will be set to zero or empty, as appropriate with a warning message.

Version Compatibility

UNIX/WINDOWS - ADAMAGIC Version 4 applies to UNIX and WINDOWS ADABAS Version 3 or later. ADABAS versions prior to that release are not supported by ADAMAGIC.

Mainframe - ADAMAGIC Version 4 applies to mainframe ADABAS version 6 (or later) databases, including v7.4.x. ADABAS versions prior to that mainframe release are not supported by ADAMAGIC.

PROCESSING OVERVIEW

ADAMAGIC command line

To invoke ADAMAGIC, use the following command line.

```
adamagic [-v]
```

The `-v` switch is optional and means produce verbose messages. If this switch is set, ADAMAGIC will send messages to the statistics file (stdout) to let you know what it is doing. If ADAMAGIC detects that it is running in a console environment, `-v` is set automatically. However, ADAMAGIC is not always able to detect a console so the `-v` switch is provided for you to force verbose messages.

ADAMAGIC input sources

- control statements (stdin)
- operator replies (stdin - reopened)
- a file specified by the PARMFILE environment variable
- some parameters optionally can be specified via environment variables
- model DTA file
- one of:
 - ◆ mainframe ADASAV dataset
 - ◆ UNIX/ WINDOWS ADABCK files
 - ◆ UNIX/ WINDOWS ADABAS database

ADAMAGIC output sources

- messages (stdout)
- reports (stdout or user specified file)

P R O C E S S I N G O V E R V I E W

- operator prompts (stderr)
- FDU, FDU, FDU ...
- DTA, DTA, DTA ...
- DVT, DVT, DVT ...
- Zero to many user specified files to hold records selected by selection specification parameters.

Parameter and Environment Variable Summary

File	Environment Variable or Parameter	Storage Medium	Additional Information
Backup	MAGDUMP	disk, tape	foreign ADASAV or ADABCK
	MAGDEVICE		Backup device (tape/disk)
	MAGTYPE		Backup system (UNIX/WINDOWS/mvs).
	MAGPROCESS		Specifies whether DAT, DVT and/or FDU files will be produced.
Model compressed	MAGMODEL	disk	output of ADACMP or ADAULD, only need the header.
<none>	MAGSCRIPT	disk	optional executable, DBID & file number as arguments
Compressed data	MAGDTA	disk	file number will be suffixed
Compressed descriptors	MAGDVT	disk	file number will be suffixed
ADAFDU statements	MAGFDU	disk	file number will be suffixed
Control statements	stdin (or PARMFILE file)		Control parameters cannot be supplied via environment variables.
ADAMAGIC messages and reports	stdout		Reports may be sent to a disk file instead of stdout.
Mount messages	stderr		
Extract file(s) specifications	your parameter name(s)	in the parameter	optional, as many as needed

		input	
--	--	-------	--

Input Data

The main input can be one, and only one, of the following types of file:

- a mainframe MVS (OS390, z/OS), MSP/EX or VSE ADASAV backup;
- a UNIX or WINDOWS backup;
- a UNIX or WINDOWS ADABAS database.

The main input is specified with the environment variable or parameters MAGDUMP or ASSOnn/DATAnn. MAGDUMP and ASSOnn/DATAnn are mutually exclusive.

The UNIX or WINDOWS (ADABCK) backup input can be on disk or single-tape medium such as an EXABYTE cartridge written by ADABCK (off-line, single drive, part or full backup). An ADABAS database must be on disk.

Where main input is an ADASAV backup written by a mainframe ADABAS system, it must have absolutely no translation (i.e. not converted to ASCII) and may be on media such as ½” 9-track tapes which can be read by your UNIX or NT system or on disk.

EXABYTE cartridges are also suitable for mainframe backup data.

The mainframe tape should preferably have IBM standard labels, although unlabelled tapes are acceptable (using "MAGDEVICE = utape" parameter).

Because ADAMAGIC needs to read past the “tape marks” which separate the mainframe labels and data, the tape device must exhibit BSD behavior (e.g. SOLARIS systems may need a device with “b” suffix, such as /dev/rmt/1b.)

If you are reading mainframe backup data from disk, the file must not contain any labels; it must be an exact image of the IBM ADASAV VB format records, including the binary record length (rdw or record descriptor word) which is part of the VB format. The block boundaries will normally disappear, along with the binary block length (bdw or block descriptor word) when you transfer the data as binary. If, however, you manage to transfer to a UNIX/WINDOWS file with the bdw intact, you should use MAGDEVICE = bdisk. (See MAGDEVICE parameter for explanation).

Note that ADAMAGIC does not need to read all the ASSO part of the input. ADAMAGIC only reads the first part of the ASSO which contains the GCB, FCB and FDT information. Therefore, where input data is on multiple tape volumes, second and subsequent tape volumes containing only ASSO data can be omitted from the input, as can volumes containing data for unselected files. Tape volumes must be mounted in the correct (ascending RABN, ASSO then DATA) sequence.

Even disk-based input can exist in multiple files - you will be prompted at run time if ADAMAGIC needs to obtain more data than is in the main disk file. There will be no prompting for more data when the main input is an ADABAS database. You must specify all necessary ASSOnn and DATAnn parameters.

Auxiliary input is a small disk file containing a sample ULDDTA as written by ADAULD. This file is used to provide version information for writing the headers of compressed output files and for determining the byte order for binary fields in the output file(s). This file is associated with the environment variable or parameter MAGMODEL.

Output Data

For each file, an FDU, DTA and DVT file may be written. ADAMAGIC will suffix each of the filenames with a 4-digit file number (leading zeros will be inserted if necessary). The FDU file can be edited and used as input to ADAFDU if needed. The DTA and DVT files are indistinguishable from files produced by the UNIX ADAULD utility.

Note that these DTA and DVT files cannot be written to tape - ADAMAGIC may need to write to many files “concurrently”, as the file blocks on the input medium are intermingled, and writing to tape would therefore require more tape drives than is usual. Error messages are written to standard output. File statistics are written to standard output.

Compressed Output File Processing Considerations

The notes in this section apply only if you are producing output DTA, DVT and/or FDU files.

Mainframe ADASAV input will normally be on ½” non-cartridge magnetic tape reels, as this is the lowest common denominator (although other media types can be used so long as you can write them on the mainframe and read them on your UNIX/WINDOWS system). Consequently, volume mounting is an issue.

ADAMAGIC will write a message to **stderr** where an ADASAV tape mount is needed (one message per tape volume). These prompts should be answered by the operator entering OK after the correct tape is mounted, or ABORT if the tape cannot be mounted.

When ADAMAGIC requests the “next” tape, you can skip a volume if you know that it is not needed (e.g. any volume containing only ASSO information, except the first volume, does not need to be mounted). Tape volumes must be loaded in the same sequence as they were written.

ADAMAGIC does not communicate with the operator when a UNIX/WINDOWS ADABCK tape is the input medium, but expects the tape to be pre-mounted.

ADAMAGIC can also accept ADASAV dumps that have been transferred in binary from a disk on the mainframe to a disk on the UNIX/WINDOWS system. This format requires that the record descriptor words on the mainframe in VB format be faithfully copied to the UNIX/WINDOWS file. See explanation of the MAGDEVICE parameter (Page 17) for more information on this.

ADAMAGIC drops all field and file security information when it builds its output compressed files.

ADAMAGIC will not generate hyper descriptor or phonetic descriptor values, so you should invert the file later if you use these descriptors.

ADAMAGIC does not support the rarely used “virtual field” facility of ADABAS.

Remember that the checkpoint and system files should not be processed through ADAMAGIC, as they will have no meaning to UNIX/WINDOWS ADABAS.

This version of ADAMAGIC does not process version 4 backups, nor will it process online backups. Partial backups (e.g. selected file backups) can be processed. By using partial backups, you may be able to speed processing by running two ADAMAGIC sessions in parallel.

If you are using disk as the input medium, the UNIX 2-Gigabyte file size limit can be sidestepped by using partial backups. Another method is to arrange for the input to be on multiple disk files, e.g. by copying MVS tape volumes to individual disk files before transferring with FTP.

If you have embedded hex values in a field of type A, they will not pass correctly through the EBCDIC to ASCII translation. This is not a problem where the input is a UNIX/WINDOWS ADABCK dump.

If you use over length alpha fields (e.g. by having a DDM conflicting with the FDT, or by using length override explicitly in NATURAL), ADAMAGIC will issue a warning, but will pass the data through to the compressed output files with the over length retained. Note that such files could not subsequently be processed by the ADADCU utility if this was ever needed.

Remember that U type fields are stored in a longer internal format for UNIX/WINDOWS ADABAS, so allow for this when calculating block sizes for your database (although it is somewhat offset by a shorter representation of non-NU nulls in UNIX/WINDOWS ADABAS).

Also note that the ADAFDU input parameters generated by ADAMAGIC may need extra safety margins manually added to the DATA and ASSO size specifications before using them. (ADAMAGIC allocates 10% above the amount of data actually in use).

ADAMAGIC will need a substantial amount of disk space to store the compressed output files, up to the same amount of space used by the database itself. Of course, you will need a lot less space than any other method of data transfer (which would have to use uncompressed files). It is possible to save even more space by using the "MAGPROCESS = short" parameter. This causes ADAMAGIC to NOT write DVT files. You will need to invert the descriptors using ADAINV if you use the "short" option.

If you reorganize your mainframe database before transferring the data with ADAMAGIC, you will make it simpler to operate ADAMAGIC and coordinate with ADAFDU and ADAMUP runs. This is because files will tend to be completed one after another in order, instead of being intermingled on the tapes. A side effect is that ADAMAGIC will use less memory to run.

When transferring a large database, it is useful to note that you can begin ADAMAGIC processing on UNIX/WINDOWS before the ADASAV process on the mainframe completes - just load tapes for ADAMAGIC as soon as they are written on the mainframe.

Although ADAMAGIC has been exhaustively tested, it is important that you test the validity of transferred data yourself. Judicious use of ADAVFY, plus inspection of random records in detail, plus execution of programs to reconcile totals, plus user testing is recommended. If you should happen to unload a file that was originally loaded from ADAMAGIC, you will notice that the descriptor byte counts and occurrences can be compared with similar counts and occurrences output by ADAMAGIC. However, you should not expect the DTA and DVT files to be identical because:

- (1) ADAMAGIC will set a different timestamp in the header;
- (2) ADAMAGIC will probably use empty field counts in a more efficient way, and
- (3) ADAMAGIC produces DVT values for a PE group in a different but equivalent order.

Restart Considerations

If an ADAMAGIC execution failed (e.g. because of a tape IO error), any files which had been completed and had their statistics written to standard output will not need to be done again in any rerun of ADAMAGIC. ADAMAGIC does not synchronize with the ADABAS system in any way, therefore it is never necessary to reset the DIB or restart the nucleus.

If you use the MAGSCRIPT option you may cause restart complications. If you run ADABAS utilities with an ADAMAGIC end of file script, you should be very cautious. Note that ADAMAGIC will cease processing until the synchronous completion of your script. If your script executes an asynchronous activity, you should have a mechanism to ensure that any ADABAS utility runs should proceed in serial fashion.

OPERATIONS

Input and Control Parameters

The following control parameters are available:

C	MAGDUMP	=	path name
C	ASSOnn	=	path name
C	DATAnn	=	path name
O	MAGDEVICE	=	device name
O	MAGTYPE	=	system name
O	MAGPROCESS	=	option
O	MAGMODEL	=	path name
O	MAGSCRIPT	=	path name
O	MAGDTA	=	path name
O	MAGDVT	=	path name
O	MAGFDU	=	path name
C	FILE	=	number
O	NEWFILE	=	number
O	DBID	=	number
O	SKIPREC	=	number
O	NUMREC	=	number
O	LLDIR	=	path name
O	MAGX5FDT	=	path name
O	FLAGS	=	option

Where **C** = required but conditions apply and **O** = optional

As well as the control parameters listed above, there are several other input commands which may be used interactively:

- HELP** or **?** displays a help panel containing parameter summaries.
- SHOW** or ***** displays current parameter values.
- !** followed by a command executes a shell command.
- QUIT** abandons the interactive session.
- START** terminates the parameter phase and initiates execution.

There are also record selection parameters which are explained in the next chapter.

Control Parameters

Control parameters can be supplied to ADAMAGIC in up to four different ways. Parameters may be supplied via environment variables, entered on the command line, typed in at the console and/or supplied in a file named by the PARMFILE environment variable. Environment variables are read first and may be overridden by parameters from the command line and then from the console or the PARMFILE file. If input is supplied both via the PARMFILE environment variable, and via command line input redirection, then the PARMFILE input takes precedence.

Note that if environment variables are set but not required, they must be unset before running ADAMAGIC, otherwise errors may occur. One way to do this, under UNIX, is to run a script similar to the following:

```
unset ASS01
unset ASS02
unset ASS03
unset ASS04
unset ASS05
unset ASS06
unset ASS07
unset DATA1
unset DATA2
unset DATA3
unset DATA4
unset DATA5
unset DATA6
unset DATA7
../src/adamagic < $1
```

A script like the one above will unset the environment variables only for the execution of the script and not effect the environment variables set in your shell session.

NOTE: Set PARMFILE to the name (or full or relative path as desired) of the file that contains ADAMAGIC's run time parameters. This will cause ADAMAGIC to read the specified file instead of asking for its parameters from the console. If PARMFILE is not set, ADAMAGIC will read any environment variables and then issue a console prompt to ask for any further parameters (or overrides).

It is probably usual to use "SET PARMFILE=<filename/path>" in an WINDOWS environment. Otherwise you will get a (probably) unwanted console prompt.

Examples of PARMFILE values:

```

SET PARFILE=magicparms.txt
SET PARMFILE=c:\mydir\magicparms.txt
SET PARMFILE=..\parmdir\magicparms.txt

```

Syntax NOTE: Under WINDOWS, file names and parameter values are not case sensitive. Under UNIX, parameter values (such as MAGDEVICE) are not case sensitive but file names and paths are case sensitive. For either operating system, parameter values may be specified in mixed case. Eg. **MAGDEVICE=tape** or **MAGDEVICE=Tape**

Also, note that UNIX systems use / as a directory delimiter and WINDOWS uses \ as a directory delimiter in a path specification. If you copy an example parameter set from one system to the other, remember to change the slashes.

General Parameter Syntax

- Any record in the input that starts with a hash (#) or an asterisk (*) is a comment record and is ignored.
- Blank lines are ignored.
- Each record consists of a number of "words" (where a word is a string of symbols preceded or followed by any number of delimiting blanks).
- For control parameters, the type of record is decided by the first word in the record.
- Blanks are the only word delimiters and cannot be embedded in a word (except in the special case where a blank can be embedded in a quoted literal constant string).
- Comments can be added after the last significant word on any record by preceding the comment with a semicolon (;).
- Records cannot be continued. Ie. All parameters and values for a keyword must appear in the same record.
- Records can appear in any order (except for the hierarchical order of FILE and its subordinates).
- Case is not normally significant. It may be significant within user specified values that are used for comparison with fields in the database.

MAGDUMP

MAGDUMP = path name

Optionally override any MAGDUMP environment variable. It must specify the file name (or tape device name) of a mainframe EBCDIC ADASAV database backup or a UNIX/WINDOWS ASCII ADABCK database backup being input to ADAMAGIC. This is the main input to ADAMAGIC. See also the associated parameters MAGDEVICE and MAGTYPE below.

Conditionally required - MAGDUMP is required if the primary input is an ADABAS backup file. It must be omitted if the primary input is an ADABAS database.

ASSOnn

ASSOnn = path name

Optionally override any ASSOnn environment variable(s). It must specify the file name of the ASSO portion of the UNIX/WINDOWS database being input to ADAMAGIC. This is the main input to ADAMAGIC. See also the associated parameters MAGDEVICE and MAGTYPE below.

Replace **nn** with a sequential number, starting at **1** and incrementing by 1 for each subsequent ASSO file. Leading zeros are optional. Eg. ASSO1 ASSO02. You must specify as many ASSOnn parameters as there are physical files in your database's Associator. You must also specify the ASSOnn files in ascending RABN sequence.

If you are extracting information from the main database as defined by your database administrator, you may find that the ASSOnn environment variables are created when you log in. If that is the case you will not need to specify any ASSOnn parameters.

Note: The database must be stopped or the files to be extracted must be locked before reading a database directly, otherwise data integrity cannot be guaranteed.

Conditionally required - ASSOnn is required if the primary input is an ADABAS database. It must be omitted if the primary input is an ADABAS backup.

DATAnn

DATAnn = path name

Optionally override any DATAnn environment variable(s). It must specify the file name of the DATA portion of the UNIX/WINDOWS database being input to ADAMAGIC. This is the main input to ADAMAGIC. See also the associated parameters MAGDEVICE and MAGTYPE below.

Replace **nn** with a sequential number, starting at **1** and incrementing by 1 for each subsequent DATA file. Leading zeros are optional. Eg. DATA1 DATA02. You must specify as many DATAnn parameters as there are physical files in your DATA. You must also specify the DATAnn files in ascending RABN sequence.

If you are extracting information from the main database as defined by your database administrator, you may find that the DATAnn environment variables are created when you log in. If that is the case you will not need to specify any DATAnn parameters.

Note: The database must be stopped or the files to be extracted must be locked before reading a database directly, otherwise data integrity cannot be guaranteed.

Conditionally required - DATAnn is required if the primary input is an ADABAS database. It must be omitted if the primary input is an ADABAS backup.

MAGDEVICE

MAGDEVICE = device name

Optionally override any MAGDEVICE environment variable. It must be **tape**, **utape**, **disk**, **bdisk**, or **db** and specifies the device type of the primary input file.

Tapes can be standard IBM labeled (**tape**) or IBM unlabelled (**utape**). If multi-volume tape input is used, ADAMAGIC will write a mount request to stderr at the end of each tape reel.

Disk files produced by FTP from the mainframe normally will not contain “block descriptor words”. If you have a method of transferring mainframe ADASAV information to UNIX/WINDOWS which preserves the block descriptor word (see next paragraph) you must use **bdisk** instead of **disk**.

Disks can be “multi-volume” too - ADAMAGIC will prompt for a new filename if the required data is not all found on the main disk input file. Disk input must

OPERATION

not contain tape label data. If MAGTYPE is “**UNIX**” (see below) only **disk** or **tape** is allowed for MAGDEVICE and data is, of course, ASCII (without IBM labels!).

Optional - If omitted, ADAMAGIC assumes **disk** unless the MAGDUMP file name starts with “/dev/”, in which case it assumes **tape**.

Also see [File Transfer in Chapter 7](#) for further information.

MAGTYPE

MAGTYPE = system name

Optionally override any MAGTYPE environment variable. It must be **MVS**, **UNIX**, or **NT** [for WINDOWS] and identifies the type of system on which the ADABAS backup was produced or on which the ADABAS database resides.

Optional - If omitted, ADAMAGIC assumes MVS unless MAGDEVICE=db, in which case the environment variable “OS” is checked. If the value of “OS” contains the letters “NT” then MAGTYPE defaults to NT, otherwise MAGTYPE=UNIX is assumed.

MAGPROCESS

MAGPROCESS = option

Optionally override any MAGPROCESS environment variable. It must be **full**, short, **fdu** or **none**.

When **full** is specified, the FDU parameters as well as the DTA and the DVT are produced for each selected file. When **short** is specified, the DVT is not produced. When **fdu** is specified, neither the DTA nor the DVT is produced. The **full** option should be used if you intend to load the output on to a database. The **short** option might be used if you only wanted to decompress the DTA output. The **fdu** option is useful if you just need to obtain file definitions or to check the contents of the backup.

Optional - If omitted the default is **none**.

MAGMODEL

MAGMODEL = path name

Optionally override any MAGMODEL environment variable. It must be the file name of a compressed DTA file produced by your UNIX/WINDOWS ADAULD utility. It need not contain any data because it is only read to obtain version and byte order information about the machine for which the output files are intended. That information is contained within the first hundred or so bytes in the file. The default is the file “magmodel” in the current directory. (For Unix users, both “magmodel” and “MAGMODEL” will be tried.)

Conditionally required – A **magmodel** file is required, but you may omit the parameter if you are happy with the default.

MAGSCRIPT

MAGSCRIPT = path name

Optionally override any MAGSCRIPT environment variable. It must be the file name of a UNIX script or NT batch (.bat) file or executable program that takes two arguments. ADAMAGIC will execute this command as a system call at end-of-data for an output compressed file. The two arguments passed are the **DBID** and the **file number** of the output compressed file.

Optional - The default is no script.

MAGDTA

MAGDTA = path name

Optionally override any MAGDTA environment variable. It specifies the "base" filename for the output DTA file. ADAMAGIC will append a 4-digit file number to this base filename to distinguish multiple DTA files written. (Eg. MAGDTA=tstdta and a file number of 21 will produce the output file named “tstdta0021”.)

DTA files will not be used if the MAGPROCESS option is set to **fdU** or **none**.

Optional – The default base name is “dta” in the current directory. If **magprocess=none** this parameter is ignored.

OPERATION

MAGDVT

```
MAGDVT = path name
```

Optionally override any MAGDVT environment variable. It specifies the "base" filename for the output DVT file. ADAMAGIC will append a 4-digit file number to this base filename to distinguish multiple DVT files written. (Eg. MAGDVT=tstdvt and a file number of 21 will produce the output file named "tstdvt0021".)

DVT files will not be used if the MAGPROCESS option is set to **short**, **fdv** or **none**.

Optional – The default base name is "dvt" in the current directory. If **magprocess=none** this parameter is ignored.

MAGFDU

```
MAGFDU = path name
```

Optionally override any MAGFDU environment variable. It specifies the "base" filename for the output FDU file. ADAMAGIC will append a 4-digit file number to this base filename to distinguish multiple FDU files written. (Eg. MAGFDU=tstfdv and a file number of 21 will produce the output file named "tstfdv0021".)

The FDU files written by ADAMAGIC contain suggested ADAFDU control statements which could be used when defining the file to UNIX or NT. It is important that you review these statements (e.g. for appropriate space values) before using them with ADAFDU. ADAMAGIC will generate space values expressed in megabytes, based on the space actually in use for each file plus an allowance of an extra 10%.

FDU files will not be used if the MAGPROCESS option is set to **none**.

Optional – The default base name is "fdv" in the current directory. If **magprocess=none** this parameter is ignored.

FILE

```
FILE = number
```

This parameter may be entered multiple times (once per file selected from the primary input). It establishes a “processing mode” under which other file specific parameters are interpreted.

The parameters that will be associated with a preceding FILE parameter are listed below (NEWFILE, NUMREC etc.).

Note - If these FILE-dependent parameters are positioned BEFORE any FILE parameters, they will be taken as GLOBAL DEFAULTS that will apply to ALL subsequent FILE parameters.

Conditionally required – This parameter is required if you want to create DTA, DVT and/or FDU output files. Otherwise it may be omitted.

NEWFILE

```
NEWFILE = number
```

Use this parameter to renumber an ADABAS file written to a DTA, DVT or FDU file. This is the file number that will be used in the output. This file number will be used as the suffix to the DTA, DVT and FDU output file names, will be embedded in the DTA and DVT headers and used in the **FILE** parameter in the FDU file. It is not possible for ADAMAGIC to write 2 files with the same NEWFILE, even if they are for different DBIDs.

This parameter is only used if you are producing DTA, DVT or FDU output files.

Optional – Default is the same number used as input (i.e. the previous FILE parameter).

DBID

```
DBID = number
```

Use this parameter to renumber an ADABAS database written to a DTA, DVT or FDU file. This is the target database-id that will be used in the output. This DBID will be embedded in the DTA and DVT headers and used in the **DBID** parameter in the FDU file. It will also appear as an argument passed to any MAGSCRIPT command (followed by NEWFILE argument). Different FILES may have different DBIDs as long as the file number is unique.

This parameter is only used if you are producing DTA, DVT or FDU output files.

OPERATION

Optional – Default is the input database-id from the primary input file.

SKIPREC

```
SKIPREC = number
```

This specifies the number of records (i.e. ISNs) that will be skipped before commencing to write records to DTA and DVT for the currently active FILE number.

Note that record order is as found on the input (RABN sequence) - which may not be the same as logical order on the database.

Optional – Default is to not skip any records.

NUMREC

```
NUMREC = number
```

This specifies the maximum number of records (i.e. ISNs) that will be written for the currently active FILE number.

Note that record order is as found on the input (RABN sequence) - which may not be the same as logical order on the database.

Optional – Default is to not skip any records.

DLLDIR

```
DLLDIR = path name
```

This specifies the directory path where the Adamagic exits may be found. The ddl/shared-object file will have different names, depending on the OS on which Adamagic is being run, as follows:

Windows/NT	libAdamagic_exits.dll
Solaris	libAdamagic_exits.so
Linux	libAdamagic_exits.so
HPUX	libAdamagic_exits.0

The directory path must not include the ADAMAGIC exits shared library name.

With the exception of Windows/NT, the file names in the table above may be symbolic links. The path name must be a single directory (folder) without trailing (back)slash. DLLDIR may be specified either as an environment variable or in the parameter cards. If it is not specified Adamagic will do an OS dependent search as follows:

Windows/NT	<ol style="list-style-type: none"> 1. The directory from which Adamagic loaded. 2. The system directory. 3. The 16-bit system directory. 4. The Windows directory. 5. The current directory. 6. The directories that are listed in the PATH environment variable.
Solaris	<ol style="list-style-type: none"> 1. LD_LIBRARY_PATH 2. crle determined default path (contains either /usr/lib or /usr/lib/64)
Linux	<ol style="list-style-type: none"> 1. LD_LIBRARY_PATH 2. The list of libraries cached in /etc/ld.so.cache. 3. /lib, followed by /usr/lib
HPUX	<ol style="list-style-type: none"> 1. /usr/lib

MAGX5FDT

MAGXFDT = path name

MAGX5FDT specifies the name of a file containing field redefinition rules. This can only be supplied via an environment variable. Field redefinitions can only be supplied for type A (alpha) fields. These redefinitions will be processed if:

- the source field type is A
- the MAGTYPE is MVS. In other words, only if an MVS to open systems conversion is being done.

OPERATION

One or more field redefinition rules can be added via a filename defined in a file supplied in the environment variable MAGX5FDT. These redefinition rules allow alpha fields to be treated as binary or packed etc, thus skipping EBCDIC to ASCII conversion if the input data is from MVS.

Redefinition rules are of the form:

```
file.fnrlfield=len,type:redéf_elements
```

For example:

```
111.15\AA=30,A:10,A;7,B;10,A
```

where fields are:

- dbid, e.g. 111
- file number, e.g. 15
- 2-character field name, e.g. AA
- total field length, e.g. 30
- existing field type, e.g. A
- 1 - n redefinition elements

Redefinition elements are of the form:

```
len,type[;]
```

For example (from above):

```
7,B;
```

where:

- this element is 7 bytes long
- this element is binary
- ; indicates there is another element to follow

There are some caveats for use of these:

- the total of redefined fields must equal the total field length
- redefinition element lengths are in bytes. Do not confuse this with digits in packed fields.

Sample code implementing this is also supplied in the sample user exit 5 in libAdamagic_exits.cpp and libAdamagic_exits.c. This exit code may be customised if desired.

Note that the sample code also uses an environment variable MAGX5FDT. If you supply user exit 5 and this environment variable, Adamagic will call UEX5 and will not use the internal redefinition facility.

FLAGS

The FLAGS parameter at present takes only one argument, REVLBIN, which stands for "REVerse Little-endian BINary fields", meaning that little-endian binary fields will be reversed to match the SAG default. This is contrary to previous ADAMAGIC behaviour. If you do not supply this parameter then processing defaults to the previous behaviour.

Example:

```
FLAGS = REVLBIN
```

Parameter Examples

```
# = indicates a comment line
MAGDUMP=/adabas/ebcdic/dump99
MAGDEVICE = disk                ;default
MAGTYPE = MVS                    ;default
MAGPROCESS = FULL                ;default
MAGDTA = /adabas/new/dta-f       ;file no will be suffixed
MAGDVT = /adabas/new/dvt-f       ;file no will be suffixed
MAGFDU = /adabas/new/adafdu.input ;file no will be suffixed
DBID = 59                        ;general default for following files..
FILE = 11
FILE = 12
NEWFILE = 102                    ;thus file (12) will be renumbered to 102
FILE = 13
DBID = 120                       ;file 13 will become file 13 in DB 120
FILE = 14
```


Extract and/or Reformat Commands

This chapter deals with how to select records from the primary input and create sequential output files with decompressed records, possibly altering or ignoring some data along the way. Extract and select commands may be coded instead of or as well as the control parameters described in the previous chapter.

With the commands described in this chapter you will be able to supply the names of your output files, specify whether you want fixed length or variable length output records, select the records you want, change the type of some fields (eg. unpacked to packed) and/or alter the data in specific fields.

Note that extract commands cannot be supplied via Environment variables.

General Command Syntax

In addition to the syntax specified in the Control Parameters chapter, selection command records have the following syntax.

- The first word in a selection record associates it with all other records for the same output file. That first word is restricted to a maximum of 100 characters and must not contain any spaces.

Record and Field Selection Command Summary

The following selection commands are available:

extract-name	NAME	file-name
extract-name	FILE	nnn
extract-name	FTYPE	file-type rdw /delim-char
extract-name	NORMALISE	{ short long } ;spelling may be NORMALIZE
extract-name	FIELD	field-spec field-spec ...
extract-name	OFIELD	field-spec field-spec ...
extract-name	TEST	x test-word
extract-name	NTEST	x test-word
extract-name	RULE	a+b+c...
extract-name	LENGTH	nnn aa bb cc ...
extract-name	TYPE	x aa bb cc ...
extract-name	INDEX	nnn aa bb cc ...
extract-name	FORMAT	xxxxxxx
extract-name	LIMIT	nnnnnn
extract-name	STARTAT	nnnnnn
extract-name	ARCHIVE	file-name

Selection Commands

NAME

extract-name	NAME	file-name
--------------	-------------	-----------

This command names the output file. You may specify only a file name and that file will be created in the current directory. Alternatively you may specify a relative path and file name or an absolute path and file name. If a file exists with the same name and path it will be overwritten.

If you supply a **NAME** and a **FILE** command and do not supply any other commands for the same `extract-name`, all records from the specified file will be extracted and decompressed into the specified output file. `extract-name` is case sensitive and must not be longer than 255 characters.

Required – The **NAME** and **FILE** commands are the only commands that you must supply if you want to create a decompressed file.

FILE

```
extract-name FILE nnn
```

File specifies the file number from the primary ADABAS input source that must be read to find the data for this `extract-name`.

Required – The **NAME** and **FILE** commands are the only commands that you must supply if you want to create a decompressed file.

FTYPE

```
extract-name FTYPE file-type { [rdw]
                                [delimiter-character]}
```

The **FTYPE** command describes the attributes of the output file for this `extract-name`.

The parameters are positional and must be specified in the order shown in the command template above. `file-type` must be specified. The second parameter is optional.

`file-type` can have the values **flat** or **csv**.

flat indicates that the fields in the output record should have their standard length, with leading zeros or trailing blanks inserted if necessary. There will be no separator between fields. Fields whose input data is longer than the standard length will be truncated. If truncation occurs a warning message will be written to the statistics report.

If there are no PE or MU fields then the output record length will be constant. Also, if you specify the **normalise** command or specify an **index** command for every variable occurrence field, the output record length will be constant. In every other case the output record length will be variable and depends upon how many occurrences there are of PE or MU fields in the input record.

If you code the **rdw** keyword, **flat** records will start with an MVS style variable file Record Descriptor Word (RDW) to make it easy to determine the length of the record. An RDW is two bytes long and contains the length of the record (including the RDW). The record length is a two byte, unsigned binary number with the most significant byte appearing first in the record. (Eg. X'4973' means the record has a 2 byte RDW and 18799 bytes of fields for a total length of 18803 bytes).

CSV indicates that the fields in the output record should have the `delimiter-character` placed between fields. Each field will take up as little space as possible, leading zeros or trailing blanks will be removed. Fields whose input data is longer than the standard length will be output in their entirety, no truncation will take place.

All data will be converted to text format and every record will be terminated with a new line character (x'0A').

If the field being converted to character format is format **B** (binary) it will be treated as a binary number if the data length is 2 or 4. Any other length will cause the field to be treated as a bit string. In the case of binary numbers, the output will be the decimal equivalent. In the case of bit strings, the output will be a series of 0 and 1 characters.

`delimiter-character` may be coded. The `delimiter-character` may be any character except a digit (ie. must not be 0 through 9). If `delimiter-character` is omitted it will default to a comma.

Records in a **csv** format file must have the same number of delimiters in every record. Therefore, if you do not code a **normalise** command, ADAMAGIC will force the use of **normalise short**.

Optional – The default is **flat**.

NORMALISE

```
extract-name NORMALISE {[short]|[long]}  
  or  
extract-name NORMALIZE {[short]|[long]}
```

normalise specifies that ADABAS recurring data (PE/MU fields) will be separated out into multiple records.

When **normalise** is specified, ADAMAGIC will create a set of output records in such a way that all data values in the input record appear in an output record. An input record may contain zero or more MU fields, zero or more PE groups and those PE groups may contain zero or more MUs. PEs are only allowed at FDT level 01.

Any non-repeating data will be duplicated in each output record.

NORMALISE SHORT

The first output record will contain the first data value from each of the multiply occurring fields. The second output record will contain the second value, and so on. When there are no more values for a particular field, a null value will be output where that field should be. Processing does not stop until all values of all multiple fields have been output. No data is lost.

For example, say an input record has the FDT definition

```
01,AA,4,A
01,AB,30,A,MU
01,AC,PE
02,AD,6,A
02,AE,20,A,MU
```

and one input record has the values

```
AA=xyz
AB occurs 5 times and has values Tom, Joan, Alice, Anton,
Louise
AC occurs 3 times and has values AD=cook
AE (2 occ) = Darwin,
Brisbane
AD=assist
AE (1 occ) = Melbourne
AD=appren
AE (1 occ) = Adelaide
```

ADAMAGIC would produce the following **flat** output record. (NOTE: fields are in table format for clarity - ADAMAGIC would not insert the spaces or the record number.)

REC#	AA	ABocc	AB	AC	AD	AEocc	AE
1	xyz	5	Tom	3	cook	2	Darwin
2	xyz	5	Joan	3	cook	2	Brisbane
3	xyz	5	Alice	3	assist	1	Melbourne
4	xyz	5	Anton	3	appren	1	Adelaide
5	xyz	5	Louise	0			

(The fields ABocc and AEocc are the number of occurrences of fields AB and AE respectively.)

If the output format is **csv** then the same input record would produce the following result.

```
xyz,5,Tom,3,cook,2,Darwin
xyz,5,Joan,3,cook,2,Brisbane
xyz,5,Alice,3,assist,1,Melbourne
xyz,5,Anton,3,appren,1,Adelaide
xyz,5,Louise,0,,,
```

RECORD EXTRACTION

It can be seen from the above example that the number of occurrences of a PE or MU is inserted in the output record immediately before the value for that group or field. Occurrence numbers are greater than zero while there is data following them. If the data has run out, the occurrence number will be zero. In **flat** format, a zero occurrence number is not followed by any data for that group or field. However, **csv** files require the same number of delimiters in each record. Therefore, a zero occurrence number will be followed by the appropriate number of delimiters.

NORMALISE LONG

Like normalise short, the first output record will contain the first values of all multiply occurring fields. After that, all data other than the last occurring multiple field will be constant in output records, with the last multiple field changing until all its values have been output. Then the second last multiple field will move to its next value and again enough records to cover all values of the last multiple field will be written. The process will move backwards through all the multiple fields in the record. (Note that this is logically what happens. The records may not be written in the above sequence.) For example, if a record contains three MU fields (AA, AB and AC) with 2, 3 and 2 occurrences respectively, then the output would be

```
x'03010101' AA(1) AB(1) AC(1)
x'03010102' AA(1) AB(1) AC(2)
x'03010201' AA(1) AB(2) AC(1)
x'03010202' AA(1) AB(2) AC(2)
x'03010301' AA(1) AB(3) AC(1)
x'03010302' AA(1) AB(3) AC(2)
x'03020101' AA(2) AB(1) AC(1)
x'03020102' AA(2) AB(1) AC(2)
x'03020201' AA(2) AB(2) AC(1)
x'03020202' AA(2) AB(2) AC(2)
x'03020301' AA(2) AB(3) AC(1)
x'03020302' AA(2) AB(3) AC(2)
```

Each record will be prefixed by a series of bytes containing

1. The number of index bytes following this byte
2. A byte for each PE or MU field containing the occurrence number of each multiple field in this record. Occurrence numbers start at 1.

These values are shown as x'xxxx' in the above table. If **format cobol** has been specified, the count and indexes will be two bytes each.

A special pseudo field of ## (see **FIELD** command on page 33) can be used to output the ISN ahead of the physical PE/MU indexes. This combination can be used as a physical, unique key if needed.

In addition, a special test (NTEST - see page 40) can be used on normalised data. A bonus with normalised data is that the field order on output can be specified (see the discussion of using the **FIELD** command with **normalised** on page 34).

WARNING – **normalise long** can produce a huge number of output records.

Optional – maximum one per `extract-name`. The default for `shortlong` is **short**.

FIELD and OFIELD

```
extract-name FIELD field-spec field-spec ...
           or
extract-name OFIELD field-spec field-spec ...
```

The fields `field-spec` are extracted from the records in the ADABAS file associated with this extract.

`field-spec` takes one of the following forms:

- `xx`
- `xx(i-j)`
- `xx(i-j,k-l)`
- `xx(i-j,k-l,m-n)`

where:

- **xx** is an ADABAS elementary field short name
- **i,j,k,l,m,n** are integers in the range 1 through 253
- **i** is less than or equal to **j**
- **k** is less than or equal to **l**
- **m** is less than or equal to **n**
- byte ranges may overlap
- byte ranges may be out of byte sequence

If the same field name is specified more than once within the same `extract-name` the first specification will prevail.

The facility to have byte ranges out of sequence is useful for rearranging fields. (Eg. You may have a date field in `ddmmyyyy` format which is difficult to sort in chronological order. You could use the byte ranges `5-8,3-4,1-2` to rearrange the field into `yyyymmdd` format which sorts naturally.)

`field-spec` cannot contain embedded blanks.

RECORD EXTRACTION

If the `field-spec` contains the optional subfield specification then **i**, **j**, **k**, **l**, **m** and **n** designate which bytes of the decompressed field will appear (concatenated together) in the output. (For example, if LJ is an 8 byte field, "LJ" specifies all 8 bytes to be output, whereas "LJ(1-1,8-8)" specifies just two bytes output (the 1st and last bytes). Up to three ranges of bytes may be specified.) Note that subfield selection overrides any **length** specification for the same field, so the output field may be shorter than the value of **length**. (Eg. If you specify **length 8** and **field aa(2,3)** for a field that contains the character string "ABCDEF ", the resultant output will contain "BC" with a length of only 2 bytes.)

The field order is irrelevant in a **field** command. Fields are always extracted in physical (FDT) order. If you want to specify the order of the output fields use **ofield** (ordered fields) instead.

The "pseudo" field of **##** can be used to represent the ISN. Specify **##** as the field name if you want the ISN to appear in the output record. If field ordering is in effect the ISN will be placed within the other fields in the sequence specified.

The "wild field" designator of ****** can be used to represent that all fields, except the ISN, are to be included in the output record.

Note that fields must be elementary fields. Group fields (including PE group names) are not permitted on this command. Note that ordering by **ofield** is negated by the use of ******. ****** always causes fields to be written in physical order. You may use ****** as a shorthand notation to save specifying all field names if you want all fields written to the extract file and want to edit only a few fields. For example:

```
extract1 field **
extract1 field ab(5-8,3-4,1-2)
```

would write all input fields to the output file and rearrange field AB.

All fields required may be specified on one record. If you are extracting a lot of fields, the line could become very long and difficult to read. If desired, more than one **field** or **ofield** command may be coded. For **ofield**, fields will be taken in the order they appear in the input commands (i.e. start to finish, left to right). It is not an error to mix **field** and **ofield** commands for the same `extract-name`. If any **ofield** command is supplied for a given `extract-name` then the fields will be ordered.

When ADAMAGIC is processing field selection, it creates an internal FDT. So that the exact format of the output record can be determined, that internal FDT is written out to a file. The file name is the extract file name with ".fdt" appended. Eg. If you code `extract2 name extract2.data` then the internal FDT will be written into file `extract2.data.fdt`.

This file is used by FDUGEN, which also requires control statements. These are produced by Adamagic, and can be found in the file with “.inparm” suffix e.g. `extract2.data.inparm`. Note however that the parameter values in the “.inparm” file are taken from the input data, and take no account of the fact that there may be less records in the output file than there were in the input file. Hence it may be advisable to adjust these numbers down on occasion.

It is possible to select fields from within a PE and that selection may be in a different field order from the input record. Also, you may select fields from a PE, then select fields that are not in that PE, then select more fields from the same PE. When fields are selected from within a PE, ADAMAGIC attempts to maintain the PE structure, but that is not always possible. If you select more than one field from the same PE and intersperse other fields amongst them, the result will contain more than one PE with the same group name. For example:

The FDT for the input record contains (in part)

```
01,AQ,PE
02,AR,3,A,NU
02,AS,5,P,NU
02,AT,5,P,MU,NU
01,AH,6,U,DE
```

and you code the following **ofield** command

```
extract2 ofield at ah ar
```

then the output FDT will look like

```
01,AQ,PE
02,AT,5,P,MU,NU
01,AH,6,U,DE
01,AQ,PE
02,AR,3,A,NU
```

Optional – One or more **field** and/or **ofield** commands may be coded per `extract-name`. If omitted, all fields in the input record are written to the output record (equivalent to coding a `field-spec` of `**`) and output fields will be in the same sequence as they are read from the primary input file.

TEST

```
extract-name TEST x test-word
```

Used to define tests that will be used for record selection.

x represents the test identifier which must be alphabetic (A-Z) or numeric (0-9). The letters are not case sensitive. Ie. **A** is considered to be the same test as **a**. This test identifier should appear in at least one **RULE** command. The test identifier must be unique across all **test** and **ntest** commands.

RECORD EXTRACTION

The `test-word` is a string that defines a comparison test and is structured as follows:

`field-spec.operator.literal`

(e.g. `AB.EQ.C'FRED'`)

where:

- `field-spec` is **ff** or **ff(i-j)** or **ff(/*)** or **ff(i-j/*)**
(**ff**=field, **i**=1st byte, **j**=2nd byte, see below for **"/***)
- `operator` may be **EQ NE GT GE LT** or **LE**
or the following algebraic equivalents may be used
$$= \quad < \quad > \quad >= \quad < \quad <=$$
- `literal` is of the form
 - ◆ `C'AB8'` (character constant, including numeric)
 - ◆ `U'123'` (unpacked decimal constant)
 - ◆ `P'123'` (packed decimal constant)
 - ◆ `B'1??101'` (binary string)
 - ◆ `G'123.987'` (floating point number)
 - ◆ `F'92164'` (fixed point binary number)
 - ◆ `X'000001010AFFFF'` (hexadecimal constant)
 - ◆ `D'ddmmyyyy,01071999'` (date in `ddmmyyyy` format)
 - ◆ `EMPTY` (keyword meaning null, zero or blank)

The field **ff** is an ADABAS field and its standard length (after modification by any **length** commands) must be greater than or equal to **j** (if specified). If **i** and **j** are not specified the default is that **i**=1 (the 1st byte) and **j**=the field length (the last byte).

Bytes **i** through **j** will be compared against a specified constant. This constant should be **j-i+1** bytes long. If not, it will be padded with appropriate nulls/blanks/zeros or truncated as necessary so that a match can be made on **j-i+1** bytes.

Literals

The literal designator letter (C, P, etc.) may be specified in upper or lower case.

CHARACTER LITERALS

Syntax: `[+]C'<character-value>'`

The characters within a character literal are taken as they are typed and compared in a case sensitive manner. The string between the single quotes may contain any character desired.

If you want a case insensitive comparison to be performed, place a + (plus sign) in front of the **C**. eg. `+c'Smith'` will search for occurrences of “smith”, “SMITH”, “Smith”, and so on.

If you need a single quote within the string, place two single quotes together (eg. `C'Joe''s bike'`).

UNPACKED DECIMAL LITERALS

Syntax: `[-]U'<whole-decimal-number>'`

The string between the single quotes must contain only decimal digits.

Unacked decimal literals default to positive. If you want to specify a negative number, place a – (minus sign) in front of the **U**. Eg. `-U'500'`.

Note, it is valid to compare an unpacked numeric field with a character literal, provided all the characters in the literal and the field are numeric. Admagic will automatically make any conversions necessary and make the comparison in numeric mode.

PACKED DECIMAL LITERALS

Syntax: `[-]P'<whole-decimal-number>'`

The string between the single quotes must contain only decimal digits.

Packed decimal literals default to positive. If you want to specify a negative number, place a – (minus sign) in front of the **P**. Eg. `-P'500'`.

BIT STRING LITERALS

Syntax: `B'<string-of-1s-0s-?s>'`

RECORD EXTRACTION

The string between the single quotes must contain only ones, zeros or question marks. The question mark is used as a wild card when it does not matter whether the corresponding bit in the input data is a one or zero.

The **B** literal is used when you want to compare a field (of any format) with a bit string. If the data value has fewer bits than the number of characters in the literal, the data will be padded on the right with zero bits. If the literal contains fewer characters than the number of bits in the data value, the literal will be padded on the right with question marks. The comparison is made with the input data from left to right one bit at a time. Use a wild card (?) when you want to ignore a bit position.

To compare a binary format field as a number, use the **F** literal. In this case the length of the binary field must be either 2 or 4 bytes (16 or 32 bits). Note that binary (B format) fields are unsigned so are always positive. Therefore do not place a minus sign in front of the **F**.

FLOATING POINT LITERALS

Syntax: `[-]G'<decimal-number-may-contain-fraction>'`

The string between the single quotes must contain decimal digits and, optionally, one decimal point.

Floating point literals default to positive. If you want to specify a negative number, place a - (minus sign) in front of the **G**. Eg. `-g'500'`.

Tests involving “type G” fields (i.e. ADABAS version 5+ floating point) can be made using the `G'nn.mm'` floating point literal.

FIXED POINT BINARY LITERALS

Syntax: `[-]F'<whole-decimal-number>'`

The string between the single quotes must contain only decimal digits.

Fixed point binary literals default to positive. If you want to specify a negative number, place a - (minus sign) in front of the **F**. Eg. `-f'500'`.

HEXADECIMAL LITERALS

Syntax: `X'<hexadecimal-string>'`

The string between the single quotes must contain only digits 0 to 9, the letters A to Z (in upper or lower case) or commas.

The length of a hexadecimal literal (after conversion of 2n hex digits into n bytes) may be different from the length of the input field. If the lengths differ, the shorter will be padded on the right with hex zeros before the comparison is made.

Note that hexadecimal literals may be made more humanly readable by splitting them, at a byte boundary, with a blank or a comma. Eg. X'0134,AD3f,c456' is equivalent to X'0134AD3fc456'. Case is not significant.

Hexadecimal comparisons are performed left to right one byte at a time. No conversion or rearrangement of the data is performed.

DATE LITERALS

Syntax: D'<format-literal>, <date>'

The syntax of a date literal is **d'ddmmyyyy,01071999'** where ddmmyyyy specifies the format the date is stored in the database field. Use the same format to code the comparison date on the **test** command. The three components **dd**, **mm**, and **yyyy** are keywords. They may appear in any order. Eg. Specify yyyymmdd if your dates are stored with the year first, the month next and the day last. Dates are compared in chronological order regardless of the format in which they are stored.

Adamagic prefers 8 digit date fields. However, to cater for non-Y2K date fields (IE. the year is 2 digits not 4) a six digit field is allowed to be compared with a date literal. You must still code the date literal as above with 8 digits. The 6 digit input field will be padded with a calculated century. If the year is in the 70s, 80s or 90s then the supplied century number will be 19, otherwise it will be 20.

If an input field is null the date comparison is considered to be not true and the record will not be selected for output.

EMPTY

The special literal **empty** may be used to mean a null, blank or zero of a length appropriate to the field being tested. **Empty** may be in upper case, lower case or mixed case.

You can only compare a field to be equal to or not equal to **empty**.

GENERAL NOTES

1. Normally, when a test is applied against a field that has multiple occurrences (MU or fields within PE) then it will be considered "true" if ANY occurrence value is "true" with respect to that test. If you wish the test to yield a "true" value only where ALL occurrence values are "true", then you must add the /* (as specified in the syntax on page 36), in order to force testing of ALL occurrences.
2. TESTs are applied after applying any **field**, **length** or **type** specifications and before processing **normalise** or **index**.
3. If the format of the input data is not the same as the comparison literal, the input field will be converted to the literal format before making the comparison.
4. It is not valid to compare binary (format B), floating point (format G) or fixed point (format F) fields with a date literal.

Optional – A total of 36 **test** and **ntest** commands per `extract-name` may be specified.

NTEST

```
extract-name NTEST x test-word
```

The **NTEST** command, apart from the omission of the "/*" specification, is identical in syntax to the **TEST** command. **NTEST** stands for **N**ormalisation **T**est and means that the test is applied only after the basic record has been selected as a result of **TEST** commands. It is applied not to the ADABAS record but to EACH of the normalised records produced from any given ADABAS record. It enables you, in effect, to select which occurrences will be output. Obviously the "/*" is meaningless on normalised records which are "flat" records without any multiple occurrences.

Fields specified in an **ntest** command must be multi-valued (IE. each field must be either an MU or be a member of a PE group).

Optional – A total of 36 **test** and **ntest** commands per `extract-name` may be specified.

RULE

```
extract-name RULE a+b+c...
```

a, b, c, etc. represent test identifiers defined in a **TEST** (or **NTEST**) command. If only one test identifier is present, then no "+" is required. There must **NOT** be any white space surrounding the "+" symbols. The meaning of this command is that record selection will occur if **ALL** the comparison tests mentioned in this command are true. If some tests are false for a certain record under a particular **rule**, it is still possible for the record to be selected under another **rule**. (i.e. tests are ANDed on any one **RULE**, but multiple **RULE** commands are ORed for a given `extract-name`).

Note - where a **TEST** involves a field with multiple values (in a PE and/or MU), it will be considered to be true if it is true for any of the multiple values, unless the "/"* notation was specified in the **TEST** command.

Optional – As many **rule** commands as required may be specified for each `extract-name`.

LENGTH

```
extract-name LENGTH nnn aa bb cc ...
```

nnn (1-253) is a length valid for the type of the fields aa, bb, cc, etc. The minimum length of any field is 1 byte. The maximum length value (in bytes) allowed for each field format (type) is as follows.

- **A** (alpha) – 253;
- **U** (unpacked numeric) - 29;
- **B** (binary) – 126;
- **P** (packed decimal) - 15;
- **F** (fixed point binary) – length value must be exactly 2 or exactly 4;
- **G** (floating point) – length value must be exactly 4 or exactly 8.

The effect of **length** is as if the ADABAS FDT actually contained the "standard length" of **nnn** for those fields, and hence, the extracted decompressed field length.

Fixed length fields cannot have a length override.

During extraction, fields may be truncated or padded out to the specified length. If truncation occurs, a warning message will be issued.

RECORD EXTRACTION

Length will be ignored for CSV format output files because field lengths in CSV files are always as short as possible.

As many **length** commands as needed may be specified.

Optional – The default is to use the standard length as found in the FDT.

TYPE

```
extract-name TYPE x aa bb cc ...
```

The fields aa, bb, cc, etc. will be converted to the data type **x**.

x (the output data type) may have one of the values:

- **A** (alpha);
- **U** (unpacked numeric);
- **B** (binary);
- **D** (date type);
- **P** (packed decimal);
- **F** (fixed point binary);
- **G** (floating point).

Support for long alpha fields (LA Only) has been introduced in this release of ADAMAGIC.

Conversion from any data type to any other data type is permitted provided the data adheres to the following rules, with the exception of D type, see below. The output field length is the standard length from the FDT unless overridden by a **length** command.

Data conversion Rules and Restrictions

- **From A (alpha)** - Where an alpha (A) type field is converted to a numeric type field, all the characters must be numeric except for a possible leading plus or minus sign, leading and/or trailing spaces (which will be ignored), commas (which will be ignored) and one decimal point. If a decimal point is found and the output field is other than float, the fractional part will be lost (truncated) and a warning message issued.

If the output type is **B** then the converted number must fit into 32 bits (unsigned). If it doesn't, significant digits will be lost and a warning issued.

If the output type is **F** then the converted number must fit into a two or four byte, signed, binary number. If it doesn't, significant digits will be lost and a warning issued. If the length of the output field is not 2 or 4, 4 will be forced.

Conversion to output format **G** is limited to the first 100 characters. If the length of the output field is not 4 or 8, 8 (double precision floating point number) will be forced.

Significant digits beyond the size of the output field will be lost. If loss of significant digits occurs, a warning message will be written into the statistics report. If any non-numeric characters (other than those mentioned above) are found, the output field will be set to empty and a warning message written into the statistics report.

- **B (binary) to F (fixed point binary)** – Only the least significant two or four bytes are converted which may result in the loss of significant digits. If loss of significant digits occurs, a warning message will be written into the statistics report.
- **F (fixed point binary) to B (binary)** – A straight copy is performed. No conversion of the data is attempted. IE. The data is treated as a string of bits, not a number.
- **From P (packed)** – If the output type is of shorter precision than the number of digits in the packed value, high order significant digits will be lost. If loss of significant digits occurs, a warning message will be written into the statistics report.
- **From G (float)** – If the output type is of shorter precision than the number of digits in the float value, high order significant digits will be lost. If the output type is anything other than character (**A** or **LA**) then the fractional part of the number will be lost. If any loss occurs, a warning message will be written into the statistics report. **Warning:** Adamagic uses standard system routines for processing floating point values. In most cases the output value will be correct. However, the authors and suppliers of Adamagic take no responsibility for any inaccuracies introduced into your data due to the processing of floating point numbers.
- **From U (unpacked)** – Unpacked data is compatible with all other types. There are no restrictions when converting U type data.

This command can be used, for example, where unpacked data would be too long for arithmetic operations, or where packed data is preferred for brevity.

RECORD EXTRACTION

Where a field is affected by both a **TYPE** and a **LENGTH** command, the **TYPE** command is applied first.

As many **type** commands as needed may be specified.

D type

This type is used for a very specific conversion, i.e. from internal Date/Time to the format yy-mm-dd (unless a date-exit (UEX4) is specified). It is only applicable to csv output. Its purpose is to make internal Date/Times legible for humans. It applies only to Date/Time fields stored in Natural's **internal** format. Natural's other formats can easily be read as they are. Dates with year higher than 70 are 19.. dates, those with year lower than 70 are 20.. dates. UEX4 makes it possible to change the formatting of the date to anything desired (see Adamagic exits).

Optional – The default is to write output fields with the same type as their input.

INDEX

```
extract-name INDEX nnn aa[#] bb[#] cc[#] ...
```

The fields aa, bb, cc, etc. must be either PE group fields or MU fields. nnn must be a valid index (1-191). **INDEX** commands override the number of occurrences found in the input records.

If more occurrences are present than **nnn**, then the occurrences from **nnn**+1 onwards are ignored. If fewer occurrences than **nnn** are present, then additional null fields are output in order to make up the number. The recurring field(s) will be preceded by a 1 or 2 byte count (see **format cobol**) that reflects the number of occurrences in the output record (not the input). ADAMAGIC writes a message to the statistics report each time the index limit is exceeded, specifying field name, ISN, INDEX, and the number of occurrences.

The purpose of this feature is to allow "dumb" post-processing programs to use a fixed record layout. Incautious use may cause a lot of wasted space in the extract file.

The # after the field name is optional. If you append the flag “#” a different output format is obtained.

- stands for number of occurrences. When this option is used, the PE/MU entries will be padded out to the INDEX value with empty entries if necessary, however the count field will contain the number of entries that contain real data, as taken from the input record. For example, suppose INDEX 10 is specified for field AA and that for a given record the actual number of entries in the input record is only 3. Then the output will contain 3 entries from the input, plus 7 null entries, but the count at the front will still only be 3.

As many **index** commands as needed may be specified.

Consider using **normalise** instead of **index**.

Optional – The default is to write the same number of occurrences as are read.

FORMAT

```
extract-name FORMAT xxxxxxxx
```

The only allowed value for xxxxxxxx is COBOL. This will have the effect of forcing output indices (i.e. occurrence counters) for MU or PE fields to be 2 bytes binary instead of 1 byte. The output may more easily be post-processed by COBOL programs using OCCURS DEPENDING. This option will have no effect if the NORMALISE option is also specified for the file.

Optional – The default is to write a 1 byte occurrence number.

LIMIT

```
extract-name LIMIT nnnnnn
```

"nnnnnnn" is a number from 0 to very large (it must fit in a double precision floating point variable). When the number of records read from the primary input file exceeds this number plus the startat number minus 1, no further dump records will be tested against selection criteria for this extract, (i.e. no more output will be produced for this extract). If the limit is 0 then the extract file will not be opened.

If **normalise** is not specified then the number of output records will equal "nnnnnn". If **normalise** is specified then the number of output records is not predictable as it depends upon the number of occurrences of each multiply occurring field in the input data.

Optional – The default is to read and process all input records.

STARTAT

```
extract-name STARTAT nnnnnn
```

"nnnnnnn" is a number from 0 to very large (it must fit in a double precision floating point variable). One less than this number of records will be read from the primary input file but will not be tested against selection criteria or be written to the extract file. Applying the selection criteria and possible writing to the extract file will commence from the "nnnnnn"th record read.

NOTE that records are read in the physical order in which they appear in the database. That is not necessarily ISN sequence.

Optional – The default is to read and process all input records.

ARCHIVE

```
extract-name ARCHIVE file-name
```

This command names an archive file. The **archive** command works in conjunction with one or more **test** commands. If you specify an archive file, all records not selected by any **test** command will be written to the archive file. For example, you may code a test to select all records in your database that have a certain date field on or after July 1st 1999. All records that contain that date or later will be selected and written to the **NAME** command's file. All records with a date earlier than that specified will be written to the **ARCHIVE** file.

You may specify only a file name and that file will be created in the current directory. Alternatively you may specify a relative path and file name or an absolute path and file name. If a file exists with the same name and path it will be overwritten.

NOTE that records are read and written in the physical order in which they appear in the primary input file. That is not necessarily ISN sequence.

Also note that **archive** does not apply to **ntest** results.

Optional – The default is to ignore all records that are not selected by **test** commands.

DISCLAIMER

Although ADAMAGIC's record selection process has proven to be very robust and reliable, it is wise to check that decompressed output is as expected. This can be achieved by using the **STARTAT** and **LIMIT** commands to generate a test output file with a small number of records (e.g. **LIMIT 99**) and then browsing the output file with your favourite binary data viewer. You should also thoroughly check all messages in the statistics report, particularly those pertaining to loss of information. As with any data conversion exercise, it is also prudent to retain the original primary input file(s) (or a backup thereof) until satisfied that the extracted data has been tested and validated.

Here is a list of items that should be checked. The list is only a suggestion and is not intended to be complete.

- Are ISN's needed in the output?
- Ensure that all PE's and MU's have the correct counts.
- Are output record lengths correct?
- If a suitable application exists for the data, use it to check that the records (and fields) pass application edits and process as expected.
- Ensure that the output file is as expected in terms of the number of records and the contents of fields.

Create ADABAS Files

ADAMAGIC can read text format data and reformat it so that it can be loaded into an ADABAS database with ADAMUP. The input text file is in **CSV** (comma separated, variable length field) format such as produced by Microsoft Excel. This file must not contain any binary data. If the ADABAS record is to contain any multiple value fields or groups (MU or PE) then the number of occurrences must be indicated by a number preceding the multiple occurrences.

For example, if a record contains the field AT defined as 1,AT,5,P,MU and it has three occurrences with values 15, 25 and 186, then its input text data would look like

```
... , 3, 15, 25, 186, ...
```

The occurrence number is allowed to be zero, in which case no values follow it for that multi-value field. If an occurrence contains an empty value, the empty value will not appear in the output record and the number of occurrences will be reduced accordingly.

The CSV data is converted to ADABAS format according to the specifications in the FDT contained in the primary input file (magdump or ASSOn/DATAn). Only the ASSO part of the primary input is read.

The output files are compressed DTA, DVT and FDU files.

Text data input is signaled by the MAGPROCESS parameter having the value

```
magprocess = CSV(<csv-file-name>)
```

The csv-file-name is delimited by a right parenthesis, a blank or a semicolon.

If the run of Adamagic is to process input text data then no other functions are permitted. IE. Adamagic can process CSV text **or** produce extracts and/or compressed data, not both.

Example.

```
asso1 = AdamagicData\unix\ASSO1.007  
asso2 = AdamagicData\unix\ASSO2.007
```

PROCESSING TEXT DATA

```
asso3 = AdamagicData\unix\ASSO3.007
asso4 = AdamagicData\unix\ASSO4.007
data1 = AdamagicData\unix\DATA1.007
data2 = AdamagicData\unix\DATA2.007
data3 = AdamagicData\unix\DATA3.007
data4 = AdamagicData\unix\DATA4.007
magtype = UNIX
magdevice=db
magprocess = csv(AdamagicParms\csv.f15test.data)
magmodel = AdamagicData\unix\magmodel3.1.1.22unix
magdvt = csvdvt
magdta = csvdta
magfdu = csvfdu
file = 15
```

The above commands would cause Adamagic to read the text file "AdamagicParms\csv.f15test.data", convert its fields in accordance with the FDT for file 15 in the ASSO file(s) and create three, compressed output files named csvdvt0015, csvdta0015 and csvfdu0015. The output files are suitable for input to ADABAS utilities that can read ADAULD output.

Adamagic exits

Four possible exits can be specified for Adamagic. These allow various aspects of Adamagic operation to be customized to specific needs.

UEX1	This exit is called prior to writing an extract record. It may be used either to modify the record before output, or to tell Adamagic not to output the record at all, i.e. as a record selection exit.
UEX2	Called prior to writing a compressed dta record. It may be used either to modify the record before output, or to tell Adamagic not to output the record at all, i.e. as a record selection exit.
UEX3	This exit is selected by requesting a TYPE conversion of a field, to type S (for “Special”). It is therefore a field level exit. See comments in the sample source for restrictions on usage.
UEX4	This is a special date(time) formatting exit. It allows for the creation of any desired output format for date(time)s that are in Natural internal format
UEX5	<p>If this exit is present, it is only called for an A (alpha) format field during the conversion of an MVS input source to a Unix or Windows output target. The exit is given the opportunity to convert the field according to whatever rules it wishes, although sample code to use redefinitions is supplied. If the exit processed the field then it must perform EBCDIC to ASCII translation where required. The exit must return the field length to indicate successful processing, or 0 to indicate it has not processed the field. If it returns 0, ADAMAGIC will then perform standard EBCDIC to ASCII translation for the field.</p> <p>Support for redefinitions has also been incorporated into ADAMAGIC itself - see Chapter 3 Operations section MAGX5FDT for details. If all you need is redefinitions, then you can define a set of rules in a file and specify the filename in MAGX5FDT, and you will have all you need. However if you want to customise redefinition processing further then you can change UEX5 as desired</p>

UEX6	If this exit is present then it will be called to perform EBCDIC to ASCII translation. It allows customisation of the translate table used.
------	---

Installation of user exits

This distribution contains a zip file called Exits.zip which contains a copy of:

libAdamagic_exits.c	the source file containing all the sample user exits
libAdamagic_exits.h	the matching header file
so_<platform>	Compile scripts for each platform supported. It is advisable to use these to compile the exits.

Both the script file and the source file contain explanatory comment. For NT/Windows 2000, there is no script file. Instead the following files are supplied.

libAdamagic_exits.bpr	A Borland project file
libAdamagic_exits.cpp	The source (identical to the “.c” version).
libAdamagic_exits.h	The matching header file
libAdamagic_exits.def	A file which establishes the relationship between internal and external exit names.

If no Borland C++ compiler is available, then these will need to be adapted to suit whatever is available. Any or all of the exits may be in(ex)cluded. Adamagic tests for their presence, and calls them if they exist. Needless to say, Adamagic runs fastest when no exit is called so, by preference, do not include unnecessary exits.

A source file containing samples of all exits is included with Adamagic. It is called libAdamagic_exits.c(.cpp for Windows/NT). Once compiled (see below), the resultant executable module should have a system dependent name, such as specified in the table under DLLDIR, or at least a symbolic link of that name must exist, which points to the real file. The “lib” prefix is required by some OS’s if the module is to reside in the default system library for shared-objects (usually /lib).

The sample source is written in C. It may be compiled with any C compiler however GNU gcc was found to work well. A compile script has been included, which may be used to facilitate compilation and installation. The scripts and parameters for the various OSs are:

Windows/NT	A Borland project file (libAdamagic_exits.bpr - included) was used rather than a script.
Solaris	so_solaris with parameter 32 or 64 (depending on the machine)
Linux	so_linux
HPUX	so_hpux

Sample build scripts starting with 'so_' are provided that can compile and link the user exits into a shared library. These scripts should be used with care, and should be customised to user requirements. They should not be run as is unless checked by a system administrator, because they will attempt to insert the shared libraries in system library directories.

User exit shared libraries can be called from a user specified directory by ADAMAGIC by using the DLLDIR environment variable as described in Chapter 3. When run in this fashion it is not necessary to create any entries in system directories. If run in this way, then the build script can be greatly simplified. For example the so_solaris script could be replaced by a single command:

```
g++ -m32 -ansi -shared -D_LARGEFILE_SOURCE -
D_FILE_OFFSET_BITS=64 -o libAdamagic_exits.so -f PIC
libAdamagic_exits.c
```

If in doubt it is strongly recommended that you consult a system administrator familiar with building shared libraries.

Usage considerations

Comments within the sample source code make clear the various parameters taken by each exit, and how they are used. Only one copy of each exit is ever loaded, so each exit must be able to handle all the input it will receive.

Parameters passed to the exits make it possible to know for what they are currently being called, i.e. for which group of parameters cards. UEX1 receives a.o. the current extract name, UEX2 receives a.o. the current file number, and UEX3 receives a.o. the current field name and current extract name. When multiple extracts are done in the one Adamagic run, UEX1 will be called for all of them. When multiple compressed files are created in one run, UEX2 will be called for all of them etc.

If an exit exists, it will be called, if not, Adamagic will process data normally. Only those exits which are required should be compiled. The others should be commented out (in preference to deleting them from the source, as they may later be needed).

Careful use should be made of UEX3, if the file is subsequently to be compressed and loaded into a database, as ADABAS utilities may reject changes made if the resultant field format does not match that specified for the ADABAS field.

Adamagic copies the input field format unchanged to the output FDT when type S is specified, so it may be necessary to manually change the FDT before using it with Adabas, depending on what the exit has done to the field. Because U type fields are stored as packed fields under MVS, Adamagic converts them back into U type fields before passing them to this exit.

If UEX4 is present it is called instead of the normal formatting that would take place for a "D type" TYPE statement (i.e. yy-mm-dd). Note that it is **only** called for the **formatting** portion of the processing, hence can only be applied to fields that are stored in actual internal format.

Examples

File transfer

Transfer of ADASAV files from mainframe to Unix/Linux/NT/Windows 2000 via FTP can cause the Block Descriptor Word (BDW) of the MVS files to be lost if precautions are not taken to prevent it. One way to do this is to first create an unblocked file under MVS which contains the BDW from the ADASAV file. This is an example of the JCL which can be used to achieve this:

```
//DDCCA2G JOB MSGCLASS=X,CLASS=A,NOTIFY=DDCCA2
//GENER EXEC PGM=IEBGENER
//SYSUT1 DD DISP=SHR,RECFM=U,DSN=DDCCA2.ADASAV.F111240
//SYSUT2 DD DSN=DDCCA2.ADASAV.F111240U,DISP=(,CATLG),
//          UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),RECFM=U
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
```

Note that RECFM=U is set on **both** input and output files. The input file is a regular ADASAV dump (1 or more files), with actual attribute RECFM=VB, however in the JCL we tell it that the input file is RECFM=U. The output file may then be transferred by FTP to the Unix/Linux/NT/Windows 2000 box without loss of the BDW. When processed with Adamagic, MAGDEVICE=BDISK should be specified, which tells Adamagic that we are dealing with a blocked input file.

An alternative method involves use of the IBM FTP command “MODE B” which may either be included in the card deck of the FTP job on the mainframe, if this is used to “put” the file, or sent to the mainframe via “QUOTE MODE B” if the receiving machine is used to initiate the file transfer with a “get” command. A prerequisite for the “MODE B” method to work is that the block size, of the ADASAV dataset, not exceed 32760 bytes. When MODE B is used to transfer the file, MAGDEVICE may be either DISK or BDISK, both work equally well.

Compressed output

This parameter deck for Adamagic takes MVS input, and creates compressed output ready for ADAMUP.

```
FILE = 11
DBID = 12
SKIPREC = 0
NUMREC = 9999999999
MAGFDU = '../output/file11.fdt'
MAGDTA = '../output/file11.dta'
MAGDVT = '../output/file11.dvt'
MAGPROCESS = FULL
MAGDEVICE = BDISK
MAGTYPE = MVS
MAGMODEL = 'model-bigendian.dta'
MAGDUMP = ../input/DDCCA2.ADASAV.F111240U
```

The example assumes that a directory called “output” exists at the same level as the current directory. It will create a dta file, a dvt file, an fdt file, and an inparm file. The latter will be created in the same directory as the fdt file.

Since the model file is “big-endian” the output will be suitable for Adamup on a Unix machine (Intel machines are “little-endian”).

Before loading the files with ADAMUP, it may be necessary to use the fdt and inparm files with the Adabas utility ADAFDU to create a new file in the database, if it doesn't already exist.

Extract output

Extract files provide more flexibility than compressed files, but need to be compressed with ADACMP before they can be loaded into a database. Only extracts with FTYPE flat rdw are suitable for ADACMP. Other restrictions also exist, e.g. normalize makes files unusable for ADACMP as do certain options on the index command and the format command. Thus care needs to be taken that command usage will not alter the record/field layout such as to make it incompatible.

An example extract deck is:

```
# Select specific fields from all records in the input
# file.
# Rearrange the byte order of some fields.
# Write the fields in the sequence they are defined in
# the FDT.
#
magdump = ../input/DDCCA2.ADASAV.F111240U
magprocess = none
magdevice = bdisk
magtype = mvs
magmodel = 'model-bigendian.dta'
extract11 name ../output/extract11.flat
extract11 file 11
extract11 ftype flat rdw
#extract11 field ** ; this line would include all
#fields
extract11 field AA AC(1-5) AZ
extract11 field AT AH
extract11 field AI(3-4,1-2)
extract11 field AX(5-6,3-4,1-2)
extract11 field AS
#extract11 limit 4 ; limit output to 4 records
#extract11 normalise short ; either spelling
#extract11 normalize long ; works
#extract11 format cobol
#extract11 field ## ; Uncomment this line to include
#ISN in the output record
```

If the input data contains NC fields, then NULL_VALUE must be specified as a control card for ADACMP when compressing the data e.g.:

```
export CMPDTA=../output/extract11.dta
export CMPDVT=../output/extract11.dvt
export CMPIN=../output/extract11.flat
export CMPFDT=../output/extract11.flat.fdt
"$ADADIR/$ADAVERS/adacmp" >
../errors/extract11.cmperror << CMPARM_END
FDT
NULL_VALUE
RECORD_STRUCTURE=ILENGTH_PREFIX
CMPARM_END
```

end of document