Version 5.22a

CCA Software Pty Ltd

ADASTRIP User Guide

Copyright Notice

Trademark Acknowledgments

ADABAS and NATURAL are trademarks of SOFTWARE AG of Germany and North America

MVS, OS390, z/OS, JES and DB2 are products of IBM Corporation, USA. MSP, MSP/AE and MSP/EX are products of Fujitsu Japan. ORACLE is the registered trademark of Oracle Corporation.

Requirements for Confidentiality

CCA Software Pty Ltd
PO Box 423
Blackburn Vic 3130
Australia

ABN: 35 060 664 057

Phone: +61-3-9894 0055
Fax: +61-3-9894-0039

Email: info@ccasoftware.com
Web: www.ccasoftware.com

Version 5.22a, November 2021

Table of Contents

Chapter 1

INTRODUCTION

Overview

ADASTRIP is a highly efficient data extraction utility that makes a single pass of relevant parts of an ADASAV database backup dataset or of an actual ADABAS database [e.g. the container files], and produces sequential output files. Each output file contains decompressed fields from a specified ADABAS file. The output file can be normalized if it contains either of the repeating type field structures - PE groups or MU fields.

ADABAS version compatibility

ADASTRIP V5.xx works with ADABAS Versions 6, 7.x, 8.1 and 8.2x input including full ADASAV backups, file ADASAV backups, online ADASAV backups or by directly reading of the database container files.

ADASTRIP V5.04 is the first version that truly supports the new MUPEX options for expanded MU and PE's, and also spanned record functionality provided by ADABAS 8.1x. - In fact ADASTRIP can be used to extract spanned records from an ADASAV backup.

QDUMP incremental backups are no longer supported, in fact QDUMP does not support ADABAS 8 processing and support for QDUMP has been withdrawn, except where existing license agreements are in force, up until ADABAS 7.4.

Selection Criteria

Comprehensive selection criteria enable output files to contain almost any desired subset of database 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. The use 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. Alternatively, ADASTRIP 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 SQL Server, MYSQL, DB2 or ORACLE. The extracted data is a sequential dataset with a record format of FB or VB. CSV output can be created using one of our standard optional user exits, which are can be purchased separately.

Parameters

ADASTRIP control parameters use a simple syntax, and report any errors clearly. ADASTRIP output reports give full statistical details of all the extracts as well as reports on the file definition tables being used (these FDT's are taken from the ADASAV tape or directly from the database when reading the container files). Fields that are to be extracted or used in selection criteria are highlighted in the FDT listing.

Automatic Generation of Parameters and Jobs by STRIP-IT

CCA has produced a GUI interface suitable for running on any standard Windows PC (including Windows 7) which when configured can be used to generate complex ADASTRIP parameters and JCL for FTP to the mainframe. This removes the users need to understand the parameter language for ADASTRIP but the still needs to understand the file structures/DDM's. This is available as an optional extra for a small upgrade fee.

Efficiency

ADASTRIP is designed for efficiency and is written wholly in Assembler language. For optimal run-time efficiency, selection criteria are pre-compiled into machine code. ADASTRIP uses BSAM where needed, OS390 or z/OS features where available, and multi-tasking to reduce run times by parallel processing.

Potential uses

A common use of ADASTRIP has been to extract large amounts of data cheaply for overnight batch processing, but there are many more potential applications, which save time and processing costs, such as:

- 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, ADASTRIP, 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.

Some of the potential uses are described in more detail Chapter 6 - OPERATING HINTS. (Page 39).

Chapter 2

OPERATIONS

Operation JCL

Operation JCL is similar to the sample JCL found in the Installation Library in members STRJCL1 through STRJCL9. The JOBLIB or STEPLIB should include the ADASTRIP load library. Note: because of the dynamic load structure of ADASTRIP, all the members of this library must be accessible, not just the top-level module (STRIP) which is mentioned in the EXEC card.

Skeleton JCL will resemble the following:

```
//jobcard
//STRIP500  EXEC  PGM=STRIP,PARM='XXXX......'
//STEPLIB   DD    DSN=xxx.ADASTRIP.LOAD,DISP=SHR
//*
//STDUMP1   DD    DSN=ADABAS.DB.DUMP1,DISP=SHR
//STPRINT   DD    SYSOUT=*
//STMESS    DD    SYSOUT=*
//STPARM    DD     *
........control parameters....
/*
//EXTRACT1  DD    DSN=DBA.STRIP.EXTRACT1,DISP=(NEW,CATLG),
//    DCB=(LRECL=9996,BLKSIZE=10000,RECFM=VB),
//    UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE)
//*
```

The JCL above is the **minimum** required to run ADASTRIP. If such features as online backups, multiple input volumes etc. are required, then additional DD cards will be necessary. Refer to Chapter 3 - PARAMETERS (Page 13) for a description of the additional JCL statements.

Note - The "PARM=XXXX...." is mandatory, unless a zap has been applied. This parameter specifies a product coded license number, which is supplied by your local distributor. Please contact your distributor for further details – see the CCA Software website http://www.ccasoftware.com.au/.

Input Datasets

Following is a summary of the input DD cards applicable to ADASTRIP:

Input Datasets	Mandatory or Optional	Description
STPARM	Mandatory	Contains the input parameters defining the processing rules for ADASTRIP.
STASSO	Optional	Must be used if (and only if) the input is direct from the database. This DD card defines the ASSO portion of the database.
STDATA	Optional	Must be used if (and only if) the input is direct from the database. This DD card defines the DATA portion of the database.
STDUMP1	Mandatory	Must be used if the input is from an ADASAV backup. Additional input datasets may be used by supplying STDUMP2 through STDUMP9. Useful if the ADASAV backup was written to more than one drive. **Note** - this dataset cannot be used in conjunction with STASSO or STDATA.
STDUMP2 through 9	Optional	Used to define additional datasets created during a backup to multiple drives.
STPLOG	Optional	Must be used whenever PLOGGING is used. Specifies the protection log active during an online ADASAV backup. DD cards for SORT processing described below must be supplied:

```
//SORTWK01  DD    SPACE=(TRK,(10,10),RLSE,UNIT=SYSDA
//SORTWK02  DD    SPACE=(TRK,(10,10),RLSE,UNIT=SYSDA
//SORTWK03  DD    SPACE=(TRK,(10,10),RLSE,UNIT=SYSDA
//SORTOUT   DD    DSN=&&TEMP,SPACE=(TRK,(10,10),RLSE,
//                UNIT=SYSDA
//SORTOUTS  DD    SYSOUT=*
//SORTDIAGS DD    DUMMY
```

STSTEMP	Optional	Required for spanned record support when reading an ADASAV input file. The RECFM=VB and the block size should be chosen large enough to accommodate an ADASAV block (though large blocks for this file are not yet supported - SORT won't handle large blocks anyway). The space allocation should be large

		enough to accommodate all spanned records on the ADASAV for all files at once that you wish to extract (extraction doesn't start for any file until all spanned records, for all files, have been copied from the ADASAV into STSTEMP
STDELTA	Optional	Used to point to the Delta save file. This feature is no longer available. The DD card is only available in ADASTRIP Version 3.01 or later and is mutually exclusive to STQDUMP and STPLOG. It must be accompanied by JCL cards for SORT, as above.

All other DD cards are output files, which must be specified by parameters in the STPARM dataset.

General Discussion

If the input is to come from an ADABAS backup, then DD cards STDUMP1 through STDUMP9 must be used. If there is only one dataset in the backup then it can be allocated any of these. ADASAV multi-drive backup datasets can be randomly allocated STDUMP1 through 9. ADASTRIP will work out the order of these input datasets by comparing the first record in each dataset. Do not use DUMMY for any input DSN.

If the input is from ADASAV backup, but you wish to use fewer drives, then you can concatenate several ADASAV datasets onto a single STDUMPx DD. A very important stipulation when concatenating is that each STDUMPx DD **MUST** have the component datasets sequenced correctly to preserve ascending RABN order within a dataset and to avoid overlapping RABN ranges between different datasets.

If the input is from ADASAV backup, but you wish to use more drives and achieve greater parallelism, you can allocate explicit volumes, (leaving out volumes that will not be needed). The order of RABNS found on any STDUMPx DD must ascend in order, and the RABN ranges on any STDUMPx DD must not overlap any other STDUMPy DD $(1 \le x < y \le 9)$.

Normally, each ADASAV backup dataset will be allocated to a STDUMPx DD (without concatenation or volume specification).

The BUFNO parameter

The STDUMPx datasets are read by QSAM and can have a BUFNO value specified explicitly in the JCL (DCB=BUFNO=x). Depending on your exact operating system, the blocksize of your ADASAV tape/cart, and the number of input datasets, you may find that a BUFNO around 30 will produce the fastest tape/cart processing. We suggest that you have a large BUFNO on datasets containing DATA and a small BUFNO (e.g. 5) on datasets containing only ASSO. If a BUFNO is not specified, ADASTRIP will default to a large value that will be appropriate for DATA but wasteful for ADASAV datasets containing only ASSO.

PLOG input

A protection log can be input to the ADASTRIP job. A protection log should only be used when the input ADASAV dataset was produced from an online (multi-user mode) ADASAV backup, and should contain the protection log blocks written during the ADASAV. The protection log must be input to DD name STPLOG.

When the PLOG DD input file is present, ADASTRIP will invoke the operating system sort (using the E15 input interface). The extracted RABNS will be sorted and written to DD SORTOUT, from where they will be merged with the input ADASAV. Hence, the presence of the PLOG DD means that the DD's needed by the sort process must also be present (refer to the sample JCL).

Direct database processing

If you want to run ADASTRIP directly against your database, then you must not use any STDUMPx or STPLOG input DDs. Instead, you must use both STASSO (for the Associator dataset) and STDATA (for the data storage dataset). Normally you would use this option when an MPM is not active, or when the files being "stripped" have been locked.

It is also possible to run against an actively updated database, but (as with any batch processing of active data) **integrity cannot be guaranteed**. For many purposes this may not matter, but remember, an unluckily timed block split may cause duplicate records. The DCB=BUFNO JCL parameter will have no effect on these datasets. The NCP JCL parameter could have an effect because BSAM is used for these datasets, but we recommend that you allow ADASTRIP to take its own defaults.

When processing directly against the database container files the user must ensure that the physical layout of the DDASSORx and DDATARx datasets exactly matches that which is defined in the database. ADASTRIP uses the Operating System information to read the data, this allows ADASTRIP to perform processing at such high efficiencies and speeds. If there is a difference between what the operating systems knows as a container file and what ADABAS does this can lead to ADASTRIP incorrectly reading invalid blocks that are part of the dataset container file but not necessarily in use by the database.

NC Field Processing

When an input field is an NC field, ADASTRIP now precedes that field in the output with an SQL significance indicator as follows:
For all valid fields the indicator is x'0000'.
For ("invalid", "null", "not filled in") fields, the indicator is x'FFFF'.

The implication is that TEST/NTEST's on these fields MUST now be specified as hex tests and the byte numbering starts at the first byte of the significance indicator. This is to enable the client to specify that they wish to select on fields that either are, or are not "real" nulls. (This of course for NC fields only. Other fields are not affected).

Output Datasets

The DD name STPRINT must be included and would normally be directed to SYSOUT. Note that this output will always include a one page documentation summary, which may be of use if parameter errors etc. are encountered, and the ADASTRIP User Guide is not readily available.

The DD name STMESS is optional. The dataset allocated to STMESS will only be opened if an ABEND occurs. It will contain a few pages of specially formatted diagnostic information, which will help the ADASTRIP product support team determine the cause of the ABEND.

With ADABAS 8 and the introduction of **'spanned'** records in the database there is a need to support possibly very large logical LRECL's, the output datasets in this case are termed **'segmented'** output and are fully described in the appropriate ADABAS V8 Utilities Manual [ADACMP].

Extract Files

A DD card will be needed for each of your extract files. The DD name and dataset names are chosen by you and must be unique within the step. The DD name in the JCL must also correspond with the DD name that you allocate in the first word of the parameter card set for the extract dataset. The extract dataset must contain sufficient space for the expected number and size of the extract records.

If the DISP is NEW, then the output format of the extract file must be defined. The DCB parameters LRECL, BLKSIZE and RECFM must be included (no defaults are taken except that LRECL will default to "BLKSIZE=4" if RECFM=VB). Please note, in versions prior to ADASTRIP V3.02, BLKSIZE=0 may not be specified.

On extract files the BUFNO and NCP DCB parameters should not be specified, since ADASTRIP's BSAM buffer pool manager ignores them. RECFM can only be FB or VB. Nothing else is permitted. We strongly recommend using VB format.

The LRECL must be big enough for the extracted record size. This size is the sum of the "standard length" for each field, (including PE/MU occurrences) plus 1 byte per PE group plus 1 byte per MU field. If VB format is used, 4 bytes must also be allowed for the record descriptor word. (Note that with VB format, LRECL can be set larger than needed but if FB format is being used, any excess in LRECL will be wasted space). Maximum LRECL allowed by ADASTRIP is 30K bytes. A RECFM of FB should not normally be used where variable PE/MU occurrences are used.

Note: - ADASTRIP parameters can be used to force fixed occurrences and to override "standard" length. Where a fixed occurrence is being forced, the 1 byte for the PE/MU index must still be allowed for. It will contain the number of occurrences in the extract record (minimum of one). (See also the explanation of the INDEX card on page 22, regarding different record lengths depending on options chosen) If PE/MU normalizing is in effect, the record length will be constant and there will be no embedded counts.

If the special FORMAT COBOL parameter is in effect, then 1 additional byte must be allowed for the PE/MU indexes instead of the normal 1 byte.

A maximum of 200 output datasets is allowed. If you are using a large number of output datasets, make sure there is no dataset that has a blocksize significantly larger than the others, as this will cause wasted space in ADASTRIP's shared BSAM output buffer pool.

Note that if ADASTRIP is run with MODE SHORT (see parameter card documentation), then it is not necessary to include the extract file DDs as they will not be opened. If you are using user exits, then the final call (end of data) will be made to the exits when MODE is set to SHORT.

A STPARM dataset must always be included (usually in-stream data). This dataset must contain card images (ADASTRIP only refers to the first 72 bytes). The data in these cards must conform to the rules in Chapter 3 - PARAMETERS (see page 13).

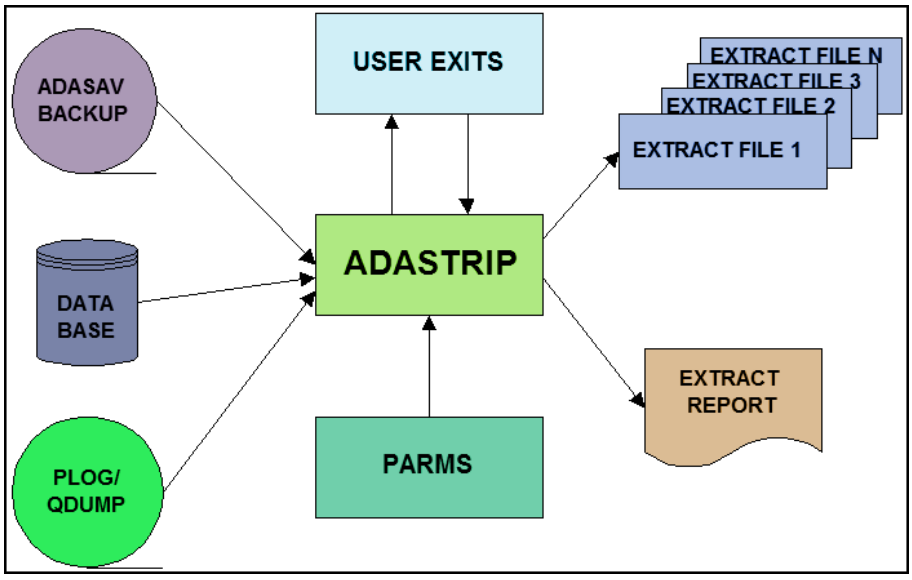

Figure 1: ADASTRIP operation

Region size considerations

A minimum of 8 megabytes should be specified. This will usually be sufficient to run ADASTRIP, however this may not be the case where a large number of extract files are being produced, and/or a large number of buffers are being used. A standard message (029I) displays the number of bytes remaining below the 16MB line before ADASTRIP allocates its shared BSAM output buffer pool.

It is recommended that you ensure that sufficient space has been allowed for this pool. The correct amount will depend on block sizes and other factors, but as a rough guide you should increase the region size if the message indicates less than 200,000 bytes in which to build the buffer pool.

Chapter 3

PARAMETERS

General Syntax Rules

The rules for general syntax are as follows:

Parameter cards with "*" in column 1 are ignored and can be used for informative comments.

Each card consists of a number of "words" (where a word is a string of symbols preceded or followed by any number of delimiting blanks).

Blanks are the only word delimiters and cannot be embedded in a word (except in the special case of the "TEST" card - see below - where a blank can be embedded in the quoted literal constant string).

Any card with a fixed number of words can have a comment added after the final blank.

The only type of card that can be continued is the "TEST" card. This card can have a "+" in column 72 to indicate the continuation. If the card contains a hexadecimal constant or string (e.g. X'0A0B'), then the continuation can only occur between whole bytes.

Parameter cards can appear in any order, providing the appropriate continuation card immediately follows a continued card.

All parameter data, except possibly comments and character constants, must be in upper case.

Keywords are only validated up to the last letter matching the standard form of the keyword, e.g. "FIELDS" will be equivalent to the standard keyword "FIELD".

Except for the "MODE" card (see below), the first word of a parameter card must be the DD name of the extract file to which it refers.

Parameter Card Format

MODE

```
MODE SHORT
or
MODE DUMP
or
MODE UPDATE
```

Optional:

SHORT - causes the Associator part of the dump to be read and FDT's to be printed out, but no extract data to be written.

DUMP - causes a "0C3 abend with dump" if there is error during record expansion, which would otherwise simply abort processing on that extract file.

UPDATE - This parameter ensures that DATA is extracted from the auxiliary input. ASSO is still taken from both STDUMPn and auxiliary input (e.g. STPLOG etc). ***Since it is impossible to determine which records in a changed RABN have changed, all records from any changed RABN (that is selected) are output***. This card only has an effect when used with STDUMPn & auxiliary input, i.e. not when ADASTRIP is run directly against the database.

Example:

```
MODE SHORT
```

DAYLIGHT

```
DAYLIGHT
```

Optional: This parameter should be used when ADASTRIP is run immediately after the change to daylight saving. Run against a dump file and a plog that was generated prior to the daylight saving changeover. Note - this parameter is to be the only parameter on this card, and should start in column 1. This parameter is valid with either ADABAS V7 or V8.

Example:

```
DAYLIGHT
```

FIELDTAB

```
FIELDTAB nn
```

Optional: In this version of ADASTRIP, the FIELDTAB parameter has been added which allows the amount of space assigned to internal FDT tables to be assigned.

The number 'nn' represents the number of Megabytes that will be assigned to the table, there must be at least one space between the parameter and the 'nn'. This parameter was introduced to allow the customer to adjust the space assigned to cater for large FDT's as indicated by e.g. the message:

FILE 00167 DD EXTR001 - OVERFLOW IN FDT

When ADASTRIP finishes it prints a message showing how much was actually used:

010% OF FIELDTAB SPACE USED

Note anything less than 1% used shows up as 0%. This parameter is valid with either ADABAS V7 or V8.

Example:

```
FIELDTAB 12
```

A total of 12 megabytes will be allocated to the internal FDT tables.

NOLENCHK

```
NOLENCHK
```

Optional: This parameter turns off default length checking for all extracts. The end result is that no warning is given on extract records that exceed the defined output LRECL and those records will be silently truncated. The LRECL checking of normalized output records is handled in the same way. This parameter is only valid for ADABAS V8 and above.

When this parameter is specified, a warning will be printed at the end of processing specifying the DDnames for which the record length was exceeded.

```
WARNING: DD <DDname> LRECL FOR THIS FILE IS TOO SMALL
```

The NOLENCHK parameter can be permanently set off by the zap that is supplied in the install library.

If this parameter is **not set** ADASTRIP will terminate processing when a LRECL length overflow error occurs.

When NOLENCHK (or the zap) is used, records that were truncated during the run, result in a warning message being output both in the STPRINT output & among the console messages upon completion of Adastrip. Previous versions only put a message in STPRINT.

Example:

```
NOLENCHK
```

RETCODE

```
RETCODE
```

Optional: This parameter will allow the user to specify any single digit return code when an index overflow condition occurs. Of course this value will usually be zero, but any value 0-9 may be specified. The default value remains 4. For the moment the RETCODE value is only used for INDEX overflow warnings, however this may be extended to cover other warnings.

The syntax is per the following example. Note that any number of spaces may separate RETCODE and the value. This parameter is only valid for ADABAS V8 and above.

Example:

```
RETCODE n
```

FILE

```
DDname FILE nnn
```

Mandatory (one per extract file). The extract file is built from fields in ADABAS file nnn (1-5000). Example:

```
DDEXTR01 FILE 158
```

FILE – FAST sub-parameter

Optional (one per extract). This sub parameter changes ADASTRIP processing to reduce CPU utilization required when more records are rejected, rather than accepted.

```
DDEXTR01 FILE 158
DDEXTR01 FAST
DDEXTR01 ……..
```

NORMALISE

```
DDname NORMALISE
```

Optional (maximum one per extract file). This specifies that ADABAS recurring data (PE/MU fields) will be separated out into multiple records. A PE with five occurrences each, holding an MU with three occurrences in the ADABAS input record, will generate 5*3 = 15 output records (in the absence of selection criteria).

Any non-repeating data will be duplicated in each of the 15 output records. Data in the PE but not the lower level MU will be duplicated in each of the 3 records for that PE occurrence (in this example). A special pseudo field of ## (see FIELD parameter on page 17) can be used to output the physical PE/MU indexes along with the ISN as a physical key if needed.

In addition, a special test (NTEST - see page 20) can be used on normalized data. A bonus with normalized data is that the field order on output can be specified (see page 18).

Usually there would be at most one PE containing one MU represented in the fields specified for output. However, ADASTRIP will deal with multiple PEs or MUs by considering them to be "tied" by their index value. A warning message will be issued if the number of occurrences for the PEs/MUs do not match. For example, if there are two MUs (say M1 and M2) under the same PE or not under a PE, then M1(1) and M2(1) values will appear together on a certain record, M1(2) and M2(2) will appear together on the next record and so on. If M2 has more occurrences than M1, then the excess will be dropped and a warning message generated. Note that it is very rare to see an ADABAS file design which "ties" fields together in this way.

Example:

```
DDEXTR02 NORMALISE
```

FIELD

```
DDname FIELD field-spec field-spec field-spec ....
```

where field-spec is one of the following forms:

"**xx**" or "**xx**(**i**-**j**)" or "**xx**(**i**-**j**,**k**-**l**) or "**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 (1 through 16381 for LA fields);

i is less than or equal to **j**, **j** is less than **k**, **k** is less than or equal to **l**, **l** is less than **m**, **m** is less than or equal to **n**.

"field-spec" cannot contain embedded blanks.

Mandatory (one or more per extract file). The field's **xx** are extracted from the records in the ADABAS file associated with this extract. If the field-spec contains the optional bracketed sub-field specification then **i,j,k,l** etc. designate which bytes of the compressed ADABAS 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).

The field order is irrelevant (fields are always extracted in physical order). The "pseudo" field of ## can be used to represent the ISN, which is considered to be the first physical field. The "wild field" designator of ** can be used to represent all fields except the ISN being extracted. The ## field is the only field which can meaningfully appear with the wild field. Note that fields must be elementary fields. Group fields (including PE group fields) are not permitted on this card.

For ADABAS expanded files, the first ISN of each expanded file in the expanded file chain is reported as MINISN + 1 for the current component file.

If you are normalizing PE/MU occurrences [non-MUPEX], then the ## pseudo field has a slightly different meaning. It will be 6 bytes long (not 4) and will contain the binary ISN followed by two 1-byte binary occurrence values. These values represent the PE or MU not contained in a PE in the 1st byte and an MU contained in a PE in the 2nd byte. If there are no PEs or MUs, then both byte indexes will be zero. If there is no MU contained in a PE, then the 2nd byte will be zero.

If however MUPEX is defined the ## pseudo field will be 8 bytes long and will contain the ISN followed by two 2 byte binary occurrence fields. . These values represent the PE or MU not contained in a PE in byte 1-2 and an MU contained in a PE in the bytes 3-4. If there are no PEs or MUs, then both byte indexes (1-4) will be zero. If there is no MU contained in a PE, then the bytes 3-4 will be zero.

If used, this ## field can make a unique key and allow you to trace the parent ADABAS record.

The order of fields specified in FIELD cards is important if you are normalizing (i.e. if you have a NORMALISE parameter for the same DDname). When normalizing, the fields will appear on the output in the specified order. However, if you are not normalizing, then the order of the fields on this card is irrelevant because the fields will appear on the output record in their ADABAS physical order.

When a field has been designated for output, it will be flagged as "OUT(**n**)" on the ADASTRIP FDT listing with **n** being the number of bytes which will be written out for this field.

TEST

```
DDname TEST x test-word
```

Optional (up to 32 per extract file). Used to define tests that will be used during record selection. "x" represents the test identifier which must be alphabetic (A-Z) or numeric (1-6). This test identifier must appear in at least one RULE card. The "test-word" is a string that defines a comparison test and is structured as follows:

```
"operand.operator.literal" (e.g. "AB.EQ.C'FRED'")
```

Where:

"operand" 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;

"literal" is of the form C'AB8' (alpha constant, including numeric) or P'123' (packed decimal constant) or X'000001010AFFFF' (hexadecimal constant).

The field **ff** is an ADABAS field and its standard length (after modification by any "LENGTH" cards) 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.

The length of a hexadecimal literal (after conversion of 2**n** hex digits into **n** bytes) must be exactly equal to **j**-**i**+1. Note that hexadecimal literals can be continued onto other card(s) as necessary. Normal constants cannot be continued.

Tests involving "type G" fields (i.e. ADABAS version 5 floating point) must be made with a knowledge of the hexadecimal representation of these numbers since ADASTRIP does not allow a floating point literal to be used.

Note that there is not a special literal constant form to represent unpacked numeric. This is because unpacked numeric is just a subset of normal character constants. Currently, negative numbers are not supported in unpacked form. To overcome this problem, use a type override to convert to packed form.

Packed decimal constants may be either positive or negative, e.g.

P'-123', P'+123' or simply P'123' (implied positive).

Normally, when a test is applied against a field that has multiple occurrences (PE/MU) 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 19), in order to force testing of ALL occurrences.

Note -- former versions of ADASTRIP supported a different syntax, whereby (for example) AA+1(2) was equivalent to AA(2-3). This obsolete form is still supported (but not in combination with the "/*" specification). Example:

```
DDEXTR01 TEST C BC.GT.C'1000'
```

NTEST

```
DDname NTEST x test-word
```

Optional (up to 32 per extract file). Used to define tests to be used during record selection. "x" represents the test identifier which must be alphabetic (A-Z) or numeric (1-6). This test identifier must appear in the RULE card.

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

```
"operand.operator.literal " (e.g. "AB.EQ.C'FRED'")
```

Where:

"operand" is "**ff**" or "**ff**(**i**-**j**)" (**ff**=field, **i**=1st byte, **j**=2nd byte);

"operator" may be EQ,NE,GT,GE,LT, or LE;

"literal" is of the form C'AB8' (alpha constant, including numeric) or P'123' (packed decimal constant) or X'000001010AFFFF' (hexadecimal constant).

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

Example:

```
DDEXTR02 NTEST A BD.EQ.C'POLICY'
```

RULE

```
DDname RULE a+b+c+...
```

Optional (up to 30 per extract file). "a", "b", "c", etc. represent test identifiers defined in the TEST (or NTEST) card. If only one test identifier is present, then no "+" is required. There must **NOT** be any blanks surrounding the "+" symbol. The meaning of this card is that record selection will occur if ALL the comparison tests mentioned in this card 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 cards are ORed for a given extract file).

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 card.

LENGTH

```
DDname LENGTH nnn aa bb cc ....
```

Optional. **nnn** (1-253) is a length valid for the type of the fields aa, bb, cc – if a field is marked as an LA (Long Alpha) the length range is (1-16381).

The effect 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 standard length. If truncation occurs, a warning message will be written. Example:

```
DDEXTR01 LENGTH 20 BC
```

TYPE

```
DDname TYPE x aa bb cc ....
```

Optional. The fields aa, bb, cc, etc. will be converted to the data type "x". "x" can only be "U" (unpacked) or "P" (packed). The data type of the fields being changed must then be "P" or "U" respectively. The effect is as if the ADABAS FDT contained this type. This card can be used, for example, where unpacked data would be too long for arithmetic operations, or where packed data is preferred for brevity. Where a field is affected by both a TYPE and a LENGTH card, the TYPE card is applied first.

INDEX

```
DDname INDEX nnn aa bb cc ....
```

Optional. The fields aa, bb, cc, etc. must be either PE group fields or MU fields. **nnn** must be a valid index (1-191 for a PE group in ADABAS V6 and 1-191 for an MU field, with MUPEX support the valid range is 1-64k values). INDEX cards override the number of occurrences found in the dump 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 output field(s) will be preceded by a 1-byte count that reflects the number of occurrences in the output record (not the input). If occurrences are dropped then, the output messages will contain a warning to that effect. 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.

```
DDname INDEX nnn aap bbp ccp ....
```

With the addition of flag "p", where "p" can be V, M, or B, a different output format can be obtained.

V - stands for **variable**. It results in no padding when the INDEX card is used (i.e. variable length records are produced). This in turn requires that a variable blocked [RECFM=VB] format be used.

M - stands for **message**. ADASTRIP puts out a message each time the index limit is exceeded, specifying field name, ISN, INDEX, and the number of occurrences.

B - stands for **both** V and M.

f or F - stands for fixed length. 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 shown, V, M, or B is appended to the name of the ADABAS field on the INDEX parameter card (e.g. AAV, BBM).

Warning: the use of B and M will cause an overhead in processing time and messages output to the system console may cause an overflow to the WTO buffer, therefore the B and M flags should be used with care.

Consider using the NORMALISE card if you do not like using PEs or MUs.

FORMAT

```
DDname FORMAT xxxxxxxx
```

Optional. Currently, the only allowed value for xxxxxxxx is COBOL. This will have the effect of forcing output indexes (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.

EXIT

```
DDname EXIT xxxxxxxx
```

Optional. "xxxxxxxx" must be the name of an executable module which is in a library concatenated to STEPLIB or otherwise accessible to the system loader. This module will be loaded into memory at the start of the ADASTRIP session and given control when a record for this DD name is to be output. It is also given control before closing the output file. The user exit can expand, contract, or otherwise alter the output record, providing the record length remains bounded by the LRECL value in the DD card in your JCL, and it can further impose selection criteria on the record. For coding details, examine the sample exit (STRI6UX1). 16 bytes of data are reserved in the main ADASTRIP control block for the use of your exits, but note that all your exits must share this area. You might have one exit per output file, with multiple input tasks concurrently executing any exit at any instant.

Your user exit must be coded re-entrantly . If running under z/OS, it will gain control in 31-bit mode with some of the parameter addresses above 16MB - you should return control in 31-bit mode. We recommend using Assembler language. Do not attempt to write to the output dataset. Do not attempt to write to your own dataset unless you are aware of the methods by which concurrent tasks can access the same dataset. Avoid doing "expensive" operations (even GETMAINS) unless you know that the exit will not be called many times. If you wish to use COBOL, be aware that simple COBOL routines produce a large environmental overhead and that you will need to write special Assembler stubs which will establish a COBOL environment and load your COBOL programs.

LIMIT

```
DDname LIMIT nnnnnnnnn
```

Optional. "nnnnnnnnn" is a number from 0 to 999999999. When this number of records has been written to the extract file specified by DDname, 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 dataset will not be opened. The default is 999,999,999.

LET

```
DDname LET VF=AA(i-j)
```

The LET statement is for use with the eSTRIP exit. It makes it possible to test on sub-fields within eSTRIP, by creating a new [virtual] field within ADASTRIP which is a sub-field of an existing field in the input record. eSTRIP can then test on this new field.

VF is the virtual field (this field does not exist in the FDT prior to definition in the LET statement and then only for the duration of the extract). It is appended to the end of the FDT and therefore any record that exists in memory. The field is available for processing in a user exit as if it exists in the 'real' FDT. This field will not be output to the extract file.

AA is the example real field from which the content of the virtual field is extracted.

i-j i = start byte, j end byte, standard rules apply i <= j <= length of field. If only one number is supplied, this is used for i, and j is assumed to be end of field.

A maximum of one LET statement can be used to define 1 virtual field [per extract file]; each virtual field must have a different two character short name to those in the FDT.

SEGMENT

```
DDname SEGMENT
```

The SEGMENT statement is for use with files that have spanned records defined. This tells ADASTRIP to output the records as segments as defined in the ADABAS ADACMP Utility manual. These files can then be processed either by ADACMP or by other external programs which can use the appropriate ADABAS definitions – refer to the "Creating and Supporting ADACMP Headers" under ADACMP/Processing in the utilities manual; there is also a table showing field layout and descriptions), and also note to include DSECTS ADAH and ADAC in the ADABAS source library.

Sample Parameters

```
FRED  LIMIT 999
FRED  FIELDS ## AA BB CC(1-1, 10-20) DD
FRED  NORMALISE
FRED  LENGTH 20 A1
FRED  INDEX 05 ZZ
FRED  TYPE P N1
FRED  EXIT UX1
FRED  TEST A AA(1-5).LE.C'ABC'
FRED  TEST B BB(2-3/*).NE.X'012F'
FRED  TEST C BB.GT.P'0'
FRED  TEST D AA(/*).GE.C'AAA'
FRED  TEST H CC(1-12).EQ.X'00000000'      +
 X'0000000000000001'
FRED  NTEST 1 DD.EQ.C'Food'
FRED  RULE A+B
FRED  RULE A+C+D+1
GEORGE FILE 25
GEORGE FIELD **
GEORGE FIELD ## **
GEORGE FORMAT COBOL
ARTHUR FILE 8
ARTHUR FIELD X1 X2
ARTHUR FIELD X3 X4
```

Sample Extract

Assume that a financial institution maintains a 1,000,000 record master file of credit card holders, with the following fields:

```
G     01    AA    HOLDER-NAME
      02    AB    HOLDER-FIRST-NAME       A     20.0  NU
      02    AC    HOLDER-LAST-NAME        A     20.0  NU
      01    AD    HOLDER-CREDIT-RATING    A     1.0   NU
                  'E' = EXCELLENT
                  'G' = GOOD
                  'P' = POOR
P     01    BA    CREDIT-CARDS
      02    BB    CARD-NAME               A     20.0  NU
      02    BC    CARD-CURR-BAL           P     9.2   NU
      02    BD    CARD-LAST-STMT-BAL            P     9.2   NU
      01    S1    NAME-KEY                A     10.0  S
```

Extract requirements

Further, assume that the institution wishes to extract a listing of all card holders whose last statement balance was over $1,000.00 and whose credit rating was poor.

Possible solutions

With a file of 1,000,000 records, such an extract would normally require some effort by application programmers or the DBA. A possible solution would be to create a superdescriptor containing HOLDER-CREDIT-RATING and a new field, STMT-BAL-OVER-1000, which would be updated with the statement balance less $1,000.00. If the most common value for credit rating were 'blank', then the resulting index would not be overly large, and would allow the required batch process to read only those records required, via a READ LOGICAL on the above superdescriptor.

But what happens if the report is run rarely and no other processing can use this new superdescriptor? It becomes very difficult to justify a superdescriptor for one occasional batch report. Yet without it, the batch process must at the very least read the whole file either via READ PHYSICAL, or by unloading and decompressing the file with ADABAS utilities.

ADASTRIP solution

The ADASTRIP solution enables multiple datasets or files to be extracted with only one pass of the database, without requiring access to the database.

Assuming that the file in question is file 158, the following set of ADASTRIP parameters would suffice:

Extract parameters

```
DDEXTR01 FILE 158
DDEXTR01 FIELD AB AC BB BD
DDEXTR01 INDEX 20 BA
DDEXTR01 TEST A BD.GE.P'1000'
DDEXTR01 TEST B AD.EQ.C'P'
DDEXTR01 RULE A+B
```

Output record

The resulting extract would contain a record for each input record that contained a credit rating of 'P' and had at least one credit card with a last statement balance of $1,000.00 or more. Each output record would contain the credit card holder's first and last names, followed by a 1-byte binary count field containing the value 20 (for 20 occurrences to follow), then a repeating group of 20 occurrences of credit card name and the last statement balance for that card. If there were less than 20 credit card occurrences in the input record, then the output record will be padded out to 20 occurrences.

Output record length

The length of the output record is calculated as follows:

```
Length      = len(AB) + len(AC) + 1 + (20 * (len(BB) + len(BD)))
      = 20 + 20 + 1 + (20 * (20 + 6))
      = 561
```

Chapter 4

NORMAL OUTPUT

Documentation Summary

ADASTRIP printed output (on DDPRINT) always begins with a one-page documentation summary. This can save referring to the full documentation if you should encounter parameter syntax errors. This summary contains the copyright notice, the installed version number and the zap level.

Extract data ID

Subsequent page headings contain information to help identify the extracted data. The database id is obtained from the GCB (it is set to "???" if the dump was a selected file backup which has no GCB). The date and time of the original backup is printed, as well as the date and time at the start of the ADASTRIP run. If input is direct from the database, then the time at "backup" will be the same as the ADASTRIP time.

Parameter cards

The contents of the parameter cards are then printed, along with identifying sequence numbers. Any severe parameter errors detected on the first pass of the parameters may also follow here (and the run would end with condition code of 8).

File numbers

The next page contains a list of all file numbers that have been specified in the parameters, along with their respective filenames as recorded in the FCB records in the dump. This can be a useful check against inadvertently specifying an incorrect file number.

Parameter errors

Any parameter errors encountered during the final pass of the card input would appear on the next page. (See Chapter 5 for more details).

RABN map

The RABN "map" for selected files is printed next - this may be compared with information printed out by ADABAS utilities if required for cross-checking.

Extract FDTs

The FDT corresponding to each extract file is now printed, along with an indication of the number of ADABAS blocks contained in the dumped file and the maximum number of records that will be extracted. The FDT looks much like an ADAREP FDT printout, but with some notable differences:

- the ISN is represented as a field with name "##";
- an index is shown against occurring fields which have been forced to a fixed number of occurrences by parameters;
- group fields are omitted; options of "SEL" or "OUT" mean, respectively, that a field is mentioned in selection criteria or is an extracted field;
- PE group members are indented;
- "END" will appear in the options area if that is the last field to be scanned by ADASTRIP.

File messages

Finally, the output will contain messages about each file that has been extracted. The number of extracted records will be detailed as well as the number that were read.

Note - keep in mind that reading will have stopped if the maximum number of output records (as specified in a LIMIT parameter) was reached.

Warning messages

Warning messages may also appear. A warning that the file has been aborted indicates serious problems and should be investigated (this will cause a condition code of 8). A file will be aborted if an output record cannot fit in the record area or contains corrupt data.

A warning that an repeating field has had occurrences dropped will be caused by specifying a forced index value that is too small (you may choose to ignore this warning).

A warning about normalization indicates that multiple "tied" PEs or MUs were found but the indexes did not match.

A warning about field truncation means that a field, which was a selection field or a candidate for extraction, contained more bytes than indicated in the standard length. Note that the truncation message will appear even if the truncation has only occurred while scanning records that actually failed selection criteria. A subsidiary message will tell you which field was the last field to be truncated. You may choose to ignore truncation or avoid it by using the LENGTH parameter to override the standard length.

Chapter 5

ERROR MESSAGES

Overview

This chapter is divided into three sections:

Processing messages.

Parameter checking messages.

Abend diagnostic information.

The following messages, which may arise during general processing, will be written to the job log.

Each processing message is of the form:

```
ADASTRIP nnnX MMM...
```

where:

nnn is a unique message number

X is the message type:

I = information
Information (I) messages may appear at anytime during processing to provide information on processing status, etc.

W = warning
Warning (W) messages may appear at anytime but will often precede an abend or termination, and provide extra information that could help resolve the problems.

E = error
Error (E) messages are always followed by an abend or termination.

MMMM... is descriptive text

Processing Messages

Processing messages are those written to the job log during actual processing of the backup tapes.

Message No	Message text	Meaning/Action
001E	WRITER TASK ABNORMAL END	An internal subtask has terminated abnormally. There will be other messages that will allow determination of the error.
002E	STPRINT SYSOUT IS MISSING	The STPRINT DD card is missing. Supply the DD card and rerun the job.
003E	STDUMP INPUT FILE(S) MISSING	The STDUMPn input files are missing. Supply the correct ADASAV datasets and rerun the job.
004W	ABEND EXIT - NO SDWA	Formatted dump could not be produced.
005E	ABEND EXIT - STMESS CANT BE OPENED	ADASTRIP's ESTAE exit has been entered and it has attempted to write to the STMESS DD card, but it was not present or was DUMMY. Check all parameters and log messages. If the problem is not obvious, supply the STMESS DD card and rerun the job.
006I	ABEND EXIT ENTERED	This is an informative message, indicating that ADASTRIP's ESTAE exit has been entered. Examine output, JCL and formatted dump to locate error. Other messages will give more details as to the cause of the problem.
007I	CLOSING XXXXXXXX	Standard message indicating that ADASTRIP is closing DDname XXXXXXXX. No further user action is required.
008I	CLOSING XXXXXXXX	As above
009I	CLOSING XXXXXXXX	As above

Message No	Message text	Meaning/Action
010E	PLOG/QDUMP NOT ALLOWED WITH ADA V4	An attempt was made to use either QDUMP incremental or PLOG input files with an ADABAS Version 4 backup. This is not supported. Remove the offending DD cards, STPLOG or STQDUMP, and rerun the job.
011E	CANNOT OPEN SORTED EXTRACT (SORTOUT)	An abortive attempt was made to open the DD card SORTOUT. Check input JCL to see if the DD card is present and correct, rerun the job.
012E	INCORRECT PARMCARD SUPPLIED	The parameter card supplied on the exec step is invalid. Correct and rerun the job.
013E	CODEWORD DOES NOT MATCH THIS PRODUCT	The supplied codeword is incorrect and does not match this product. Supply the correct codeword and rerun the job. If there is still a problem, contact your local distributor.
014W	SOFTWARE LICENCE EXPIRES IN NN DAYS	The software license will expire in NN days. Contact your local distributor for a new codeword to renew your license.
015E	SOFTWARE LICENCE HAS EXPIRED	The software license has expired. Contact your local distributor for a new codeword. It is not possible to use ADASTRIP until this is done.
016E	INVALID CODEWORD SUPPLIED	The codeword supplied is not valid. Check that you have the correct codeword on the exec step parameter. If there is still a problem, contact your local distributor for a correct codeword.
017E	SOFTWARE NOT LICENCED FOR THIS CPU	The codeword supplied is not for this CPU. Supply the correct codeword and rerun the job.
018I	VOLUME SWITCH	This is an informative message only. Issued when multi-volume input is read.

Message No	Message text	Meaning/Action
019E	STPLOG/STQDUMP DD CANNOT BE OPENED	The supplied protection log or QDUMP dataset could not be opened. Check the dataset name, correct and rerun the job.
0020I	MERGING RECORDS FROM PROTECTION LOG	This is an informative message only. It indicates that protection log records are being merged with an online ADASAV backup. No further action is required.
0021I	MERGING RECORDS FROM QDUMP INCREMENTAL	This is an informative message only. Indicates that QDUMP incremental datasets are being merged with an ADASAV backup. No action is required.
0022I	EXTRACT/SORT PHASE COMPLETED NORMALLY	This is an informative message only. No action is required.
0023E	EXPECTED PART 2 RECORD NOT FOUND	The backup and protection log may not match. Check timestamps etc. to determine the cause.
0024E	UNEXPECTED PART 2 RECORD FOUND	Similar to last message. Check that backup and protection logs match.
0025E	EOF ON PLOG BEFORE SYN2/5 FOUND	Unexpected EOF condition on the protection log before the correct termination checkpoint was found. Check that plog and backup match, and that all plog datasets in that session were concatenated together. If not, correct and rerun.
0026E	EOF ON PLOG BEFORE SYN1/4 FOUND	Unexpected EOF on the PLOG was detected before the correct start checkpoint was found. Action, same as previous message.
0027E	EXTRANEOUS OR MISSING INPUT DD NAMES	There appear to be missing/extraneous DDnames required for input. Correct JCL and rerun the job.
0028E	READER TASK ABNORMAL END	The reader sub-task has abnormally terminated. Check additional messages

Message No	Message text	Meaning/Action
		for the reason.
0029I	XXXXXXXX BYTES MAX FOR OUTPUT BUFS	This is an informative message only. A maximum of XXXXXXXX bytes is available for output buffers. This can be increased via the REGION parameter if the region is very small.
0030E	XXXXXXXX ECB WAITS	No action.
0031E	XXXXXXXX IDLE WAITS	No action.
0032E	XXXXXXXX PUT WAITS	No action.
0033E	XXXXXXXX LOCK WAITS	No action.
0034E	LICENCE EXPIRY, CONTACT YOUR DISTRIBUTOR	The software license has expired. Contact your local distributor for a new codeword.
0035E	XXXXXXXX LRECL, BLOCK SIZE INCOMPATIBLE	The DDname XXXXXXXX DCB attributes are not compatible with the supplied blocksize. Correct and rerun.
0036E	ASSO GCB NOT FOUND	The general control block has not been found. Check that the correct volume for the ASSO and/or dataset name has been supplied.
0037I	VOLUME SWITCH	No action.
0038E	STPARM CANNOT BE OPENED	The STPARM dataset cannot be opened. Check that the correct parameter dataset name has been supplied. Correct and rerun the job.
0039E	OVERFLOW IN OUTPUT MVC TABLE	Overflow has occurred in the output move table. Too many sub-fields have been used. Split your processing into two runs.

Message No	Message text	Meaning/Action
0040W	ABORT R=rrrrrrrr+X(xxxx+yyyy) ddddddddd ff	There was a problem in decompression (e.g. a bad length on index value). rrrrrrrr is the data RABN, xxxx is the hexadecimal offset to the record in the RABN, yyyy is the hexadecimal offset to the field position in the record, ddddddddd is the DD name of the output file and ff is the ADABAS 2 character field name. Examine output.
0041E	PLOG/QDUMP SORT FAILED	The SORT for protection log or QDUMP incremental data has failed. Check the supplied JCL for errors and for additional messages on the joblog. If the error still isn't determined, try the SORTDIAGS DD card.
0042E	PLOG DOES NOT MATCH ADASAV BACKUP	The protection log supplied does not match the ADASAV backup. Correct and rerun the job.
0043W	WARNING: MU/PE INDEX EXCEEDED FOR FIELD xx ISN=nnnnnnnn INDEX=ppp NR OF OCCURRENCES=yyy	This message indicates that record number "nnnnnnnn" had more occurrences than were specified by the INDEX card ("ppp") - where xx is the ADABAS field name and yyy is the actual number of occurrences for the multi-valued field for the record. Refer to INDEX card in the parameter section of this manual.
044W	BUFFER SPACE VERY LOW	This message is produced when there are less output buffers than output files. You need to increase your region size.
045E	OPEN XXXXXXXXX FAILED	This message is produced when the Associator failed to open correctly.
046E	OPEN XXXXXXXXX FAILED	This message is produced when the data portion of the database failed to open correctly.
047E	OPEN XXXXXXXXX FAILED	This message is produced when the STDUMPnn failed to open correctly.

Message No	Message text	Meaning/Action
048E	RSIZE MUST BE AT LEAST 2500K	This message is produced when too small a region size is specified for the job step.
049E	CHKFREE FAIL, INFORM ADASTR16 SUPPLIER	This message is produced when ADASTRIP fails to successfully free the memory acquired when checking the region size.
0501	MERGING RECORDS FROM DELTA SAVE	This message is produced when Delta Save data is used as output.

Parameter Error Messages

The following error messages may arise during parameter checking. The messages are self-explanatory and are accompanied by the card number of the parameter in question. These messages are reproduced here so that you can judge the meaning of a particular error message in the context of error messages that cover the same area. After the first parameter error is discovered, analysis will stop and ADASTRIP will terminate with condition code 8.

MODE card errors	INVALID MODE INVALID MODE KEYWORD
Errors detected on any card except MODE	DDNAME NOT ASSOCIATED WITH A FILE INVALID KEYWORD FOLLOWING DDNAME
FILE card errors	MISSING FILE NUMBER NON-NUMERIC FILE NUMBER INVALID FILE NUMBER DUPLICATE DDNAME
FIELD card errors	NO FIELD NAMES ON FIELD CARD XX IS AN INVALID FIELD NAME XX FIELD IS NOT IN FILE
RULE card errors	RULE CARD IS EMPTY TOO MANY RULES FOR A DDNAME INVALID SYNTAX ("+" NOT FOUND) RULE SPECIFIES UNDEFINED TEST ILLEGAL BLANK IN RULE TESTS
LENGTH card errors	NO LENGTH VALUE NO FIELD NAMES ON LENGTH CARD XX IS AN INVALID FIELD NAME XX FIELD IS NOT IN FILE INVALID LENGTH VALUE XX FIELD IS FIXED XX FIELD LENGTH TOO BIG FOR TYPE
TYPE card errors	TYPE NOT FOUND TYPE IS INVALID (NOT P OR U) EXPECTED FIELD NAME(S) NOT FOUND INVALID FIELD NAME XX IS AN IMPROPER FIELD NAME XX FIELD IS INVALID TYPE XX FIELD IS NOT ON FILE

TEST or NTEST card errors	TEST CARD IS EMPTY INVALID TEST ID DUPLICATE TEST ID FOR DDNAME MAX NO OF TESTS EXCEEDED INVALID FIELD IN TEST CARD INVALID OFFSET IN TEST CARD INVALID LENGTH IN TEST CARD RIGHT BRACKET NOT FOUND TEST RANGE EXCEEDS FIELD LENGTH OPERATOR IMPROPERLY DELIMITED OPERATOR NOT GT,LT,NE,EQ,GE,LE QUOTE NOT FOUND NO ROOM LEFT IN POOL FOR TEST DATA TYPE NOT C,X,P PACKED NUMBER GT 16 DIGITS LONG ZERO LENGTH CONSTANT INVALID CONSTANT HEX LEN MISMATCH WITH 1ST OPERAND INVALID HEX CONSTANT INVALID TYPE FOR PACKED COMPARISON INVALID BYTE POSITION SPECIFIED INVALID OCCURRENCE SPECIFICATION NTEST MUST APPLY TO OUTPUT FIELD '/*' INVALID FOR NTEST EXPECTED CONTINUATION NOT FOUND INVALID CONTINUATION (NOT HEX)
INDEX card errors	NO INDEX VALUE NO FIELD NAMES ON INDEX CARD XX IS AN INVALID FIELD NAME XX FIELD IS NOT IN FILE INVALID INDEX VALUE
FORMAT card errors	INVALID FORMAT NAME
LIMIT card errors	INVALID LIMIT
EXIT card errors	XXXXXXXX CANNOT BE LOADED INVALID USER EXIT

Miscellaneous Warning Messages

Message	Meaning/Action
WARNING: DD xxxxxxxx FIELD TRUNCATION (LAST AT yy) where xxxxxxxx is the DDname and yy is the field name.	Indicates that the field specified by the LENGTH card had a length greater than specified. Refer to the LENGTH card in the parameter section of this manual (see page 21).
WARNING: DD xxxxxxxx NORMALISATION INDEX MISMATCH where xxxxxxxx is the DDname.	Warns that due to the NORMALISE card, multi-valued field occurrences have been dropped. Refer to the NORMALISE card in the parameter section of this manual (see page 16).

ABEND Diagnostic Output

The DD name STMESS is associated with a special ADASTRIP formatted "dump". This dump is much shorter than an OS dump and will not appear if STMESS is omitted. The mere appearance of this dump does not automatically mean that a support call to the ADASTRIP distributor is warranted. The cause is probably a JCL parameter, disk space or memory space problem. The dump contains registers and snapshots at register addresses for each active task plus dumps of certain global and thread areas. When a diagnostic dump is output, always check the normal job log and STPRINT output for error messages.

System Abend/User Codes

System abend codes 322, 222, 813, 913, 878, B37 (and others) represent conditions which are probably JCL related. User abend codes will be associated with a job log message bearing the same number.

Chapter 6

OPERATING HINTS

Hints on Normalizing

ADASTRIP is frequently used to "Normalize" ADABAS data. You should usually split an ADABAS file into several output files, depending on the number of PEs and MUs. Note, that in many cases you will come across MUs (such as address lines) which have a small number of occurrences and can be replaced with a fixed number of fields (e.g. address-line-1, address-line-2, adddress-line-3). This can be done without using the NORMALISE parameter, but instead use the INDEX parameter to force a fixed number of output occurrences. When you do use the NORMALISE parameter, remember that it does not strictly expand into a rigorous normal form, but simply "flattens" records (i.e. eliminates PE/MU occurrences by building one output record per occurrence).

You must decide which essential fields must be output along with each occurrence, keeping in mind subsequent processing which may be done. In many cases, you will want to eliminate certain null valued fields by the use of the NTEST and RULE parameters. Unwanted nulls can easily happen when, for example, an MU field does not have ADABAS null suppression set on. They also result when there are "really" zero occurrences of an MU/PE field which ADABAS converts to an empty field count and ADASTRIP then converts to one occurrence of a null field. If the ADABAS key fields are not unique, ADASTRIP can add a physical key if needed (see ## field). If your ADABAS input does not contain PE and/or MU fields which are to be output, then the NORMALISE parameter is redundant and will add a small processing overhead. You may wish to use it in order to influence the order of appearance of the output fields. Certain selection tests applying to PE/MU data are much more easily done using NTEST on normalized fields instead of TEST on the "raw" data.

Creating TEST Data

ADASTRIP can be used to create logically correct TEST data from a subset of records in the database. This will help in the development and testing phases by enabling a true but small representative sample of the live production data to be used for TEST purposes. It is recommended that data scrambling be undertaken prior to release to TESTERS etc

To further simplify this process use of the **eSTRIP User Exit** available from CCA can be used to select logically related data based on keys from different files. This provides a complete subset of data which is ready to be compressed and loaded into a TEST database.

Recovery of NATURAL Programs

Another possible use of ADASTRIP is to recover accidentally lost NATURAL programs. The suggested parameters for this are:

```
NATSYS FILE 8
NATSYS FIELDS LJ LK
NATSYS LENGTH 90 LK
NATSYS TEST A LJ(8).EQ.C'LOSTLIB'
NATSYS TEST B LJ+8(8).EQ.C'PROG1'
NATSYS TEST C LJ+8(8).EQ.C'PROG2'
NATSYS RULE A+B
NATSYS RULE A+C
```

The extract file for NATSYS should be VB with LRECL=9996. This can be read by a batch NATURAL program, which can store a record with the LJ value, then loop to update it with the LK values. Alternatively, you can use more efficient methods, such as direct calls via CMADA; the extract record can be the record buffer.

The extract file could also be used as input to ADACMP to load a small temporary system file from which programs could be moved by SYSMAIN. It would be prudent to change LJ so that a special library for reclaimed programs is used for the stored source. Remember that the extract records for a program need not be contiguous, so a "crash" in mid-transaction may leave incomplete programs in the library.

Chapter 7

INSTALLATION

As a general rule the release notes contain important information relating to the release, including the installation process – these should be read in conjunction with the manual.

All ADASTRIP Software from CCA Software is now supplied as an e-mail attachment, bundled in a compressed zip file.

Note: PC files on 3½" diskette or CDR and 3490 cartridge format is no longer available.

The use of email is a widely accepted form of software delivery and it also facilitates fast transmission of new fix levels, upgrades and full releases. In future all fixes will be provided as upgraded object code in this format.

The more traditional way of releasing Mainframe software, using 3480 cartridges, is no longer supported due to the virtually universal Internet access or PC to Mainframe file transfer facilities.

Installation from Email

The release consists of one compressed zip file:

- **ASvxxxy-release.zip**

Where: AS – internal code for ADASTRIP, xxx is the version and y= is a single character defining the fix level, so ASv502c is version 502 fix level c.

Note: For a given version of ADASTRIP [vXXX] the JCL/Install library and also the User's Guide will generally not change between specific fix levels.

Installation Procedure Overview

1. Save the email attachment to disk.
2. Unzip the supplied release libraries to a directory on a PC. There should be six files in total:
   - readme.txt; a text file containing last minute information on the release;
   - AS509-UsersGuide.PDF – The ADASTRIP Users Guide;
   - AS509-Release-Notes.PDF – The ADASTRIP Release Notes;
   - AS509SN.CMP – ADASTRIP JCL, samples and Macro Library – EBCDIC TERSED compressed;
   - AS509IN.zip – install library in ASCII – MSP/EX only -- not needed in z/OS.
   - AS509SL.TRS – ADASTRIP Load library – EBCDIC TERSED compressed. z/OS only – will not appear in MSP/EX release.
   - AS509ob - ADASTRIP object library – EBCDIC object code – MSP/EX only – will not appear in z/OS release.
3. Allocate two temporary files on the mainframe for the upload files, details in following section.
4. Load the two mainframe files (AS509SN, AS509SL) to the mainframe-using FTP or whatever file transfer mechanism is used onsite – binary mode only, without any translation as the files are already in EBCDIC format.
5. Decompress the TERSED datasets using the information in the following sections, Note this procedure will be different for MSP/EX and OS390 [z/OS] users.

Procedure to Decompress the ADASTRIP Install JCL and Load Libraries – z/OS only

1. On the mainframe pre-allocate two datasets as follows: '&SYSUID..AS509SIN.TRS' 2 cyls, '&SYSUID..AS509SL.TRS' 2 cyls, (both INSTALL and LOAD datasets have a DCB=(LRECL=1024, RECFM=FB, DSORG=PS). The BLKSIZE should be some sensible multiple of 1024, such that the blocks fits on a track. Where &SYSUID. Is just your userid on the mainframe.
2. Transfer the AS509SIN.TRS & AS509SL.TRS files to the mainframe. Use a binary FTP or file transfer, the files must be loaded without using ASCII to EBCDIC translation or CR/LF and with the LRECL and BLKSIZE as specified. If you pre-allocate the datasets as specified here above, then the FTP server on the mainframe should just overwrite them, keeping the attributes as specified. This considerably simplifies the FTP transfer, as the only thing you need specify in the FTP client is "bin" to ensure a binary transfer. This is also the reason why the mainframe files should be named as specified.

3. On the mainframe run the following job, where the names of the INFILE files should remain the same, but the names of the OUTFILE files may be changed to meet site standards. TRSMAIN is available free from IBM. Once the TERSE job has been run, and the INSTALL & LOAD libraries exist, the INFILE files may be deleted.

```
//XXXXXTRS JOB MSGLEVEL=(1,1),CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//UNTERSE  EXEC PGM=TRSMAIN,PARM='UNPACK'
//SYSPRINT DD   SYSOUT=*
//INFILE   DD   DISP=SHR,DSN=&SYSUID..AS509SL.TERSED
//OUTFILE  DD
DISP=(,CATLG),SPACE=(CYL,(2,2,45),RLSE),UNIT=SYSDA,
//         DSN=&SYSUID..ADASTRIP.V509S.LOAD
//UNTERSE  EXEC PGM=TRSMAIN,PARM='UNPACK'
//SYSPRINT DD   SYSOUT=*
//INFILE   DD   DISP=SHR,DSN=&SYSUID..AS509SIN.TERSED
//OUTFILE  DD
DISP=(,CATLG),SPACE=(CYL,(2,2,45),RLSE),UNIT=SYSDA,
//         DSN=&SYSUID..ADASTRIP.V509S.INSTALL
```

The INSTALL dataset should contain approximately 30 members and the LOAD library approximately 70 members.
At this point both the JCL and Load libraries will have been populated and the members are ready for tailoring and testing. There is no longer a requirement to run the build step to create the load modules under z/OS [the decompress process has already created the load modules].

Once the above JCL/Source and Load Libraries are transferred to the mainframe and decompressed the installer will be in a position to undertake a series of test runs of ADASTRIP V5.09s to ensure correct installation.

xxxxxx.AS509s.INSTALL - library contains JCL, example user-exits, macros and ADASTRIP execution jobs.

xxxxxx.AS509s.LOAD – binary executable ADASTRIP application

The supplied example JCL must be modified to conform to local site standards, making appropriate changes to the dataset names in these members to match the ones locally at the site, especially the JOBLIB/STEPLIB card to point to the V50 Load Library.

Procedure to Install ADASTRIP for MSP/EX - only

On MSP/EX, prior to Ftp'ing the release files to mainframe disk, undertake the following steps:

Install the Source Library

Unzip the file ASxxxyin.zip and upload the unzipped *.txt files into a previously created mainframe dataset called [hlq]**.ASxxx.INSTALL** where [hlq] is your high-level DSN qualifier.

Dataset Attributes:

Organization:	**PO**
Record Format:	**FB**
Record Length:	**80**
Blocksize:	**Multiple of 80**

Install the Load Library

FTP the file ASxxxyob.jcl [where xxx is the supplied version and y is fix level, eg v502c] to the install library. The load library should have the following attributes:

Dataset Attributes:

Organization:	**PO**
Record Format:	**U**
Record Length:	**0**
Blocksize:	**Your choice, suggest same as ADABAS**

This member contains JCL to produce the load library members and will need to be modified to your library name. So, please modify both the SYSLIB and SYSLMOD DD statements to point to this library. You will also need to review the JOBCARD and modify it to meet your local site standards before running the job.

PLEASE NOTE THE JCL TO EXTRACT THE LOAD MODULES CONTAINS A WARNING THAT YOU SHOULD READ BEFORE EXECUTION.

NOTE: This file [ASxxxyob.jcl] is in EBCDIC form and needs to be loaded as a BINARY file NOT as a .txt file, that is WITHOUT ASCII → EBCDIC TRANSLATION. You should NOT, REPEAT NOT, use the ASCII CRLF option for the upload of this file.

Once the above Source and Load Libraries are in position and populated you should be in a position to run ADASTRIP.

The install library contains example JCL, parameters and user exits. Please ensure you modify these to conform to your local standards making appropriate changes to the dataset names in these members to match the ones at your site, especially the JOBLIB/STEPLIB card to point to the ADASTRIP Load Library.

Apply Product Protection Code

ADASTRIP requires a Product Protection Code, the codeword is at least 22-bytes in length and will need to be supplied so that ADASTRIP will run on your system.

A code provided previously, which hasn't expired, may still work with this new release **BUT** it is recommend that all customers request new product codes from their local support representative.

The code is supplied to ADASTRIP as part of the ADASTRIP EXEC card as follows:

```
//STRIPST EXEC PGM=STRIP,PARM='12345678901234567890123'
```

OR

The code may be permanently zapped into the ADASTRIP load modules, this zap must be created by CCA and takes the place of the CODE parameter.

An example only, of this zap is supplied in the install dataset. In order to run ADASTRIP, you will either need a codeword or zap supplied by CCA.

Attempting to apply the codeword zap to a previously zapped load module will fail, to get around this simply comment out the VER's OR apply the zap to a fresh copy of the load module. It is recommended that all original install libraries be backed up before applying any zaps.

INDEX

Q

R

S

T

U

V

W