eSTRIP v301

and

ADASTRIP User Exits

USER MANUAL

Trademark Acknowledgments
ADABAS and NATURAL are trademarks of SOFTWARE AG
of Germany and North America
MVS, MVS/XA, MVS/ESA and JES are products of IBM Corporation, USA.
MSP, MSP/AE and MSP/EX are products of Fujitsu, Japan.

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.au
Web: www.ccasoftware.com

Table of Contents

Copyright

Disclaimer

All information in this document is subject to periodic change and revision.

While every effort has been made to ensure that this document is accurate, Bateleur Software excludes its liability for errors or inaccuracies (if any) contained herein.

This document may contain references to optional products or to facilities, which require a separate licence. You should consult your Bateleur consultant for further information.

If you wish to ensure that this is the latest version (the version number is indicated within the document number on the title page) in circulation, please contact Bateleur who will inform you of the latest version number and provide updates on request.

Introduction

Extended STRIP package of consists of the three interdependent utilities, as follows:

- **ESTRIP** The generalized ADASTRIP extender, enabling the user to strip data from a file in which the records are dependent on a previous strip or some external data source. Essentially implementing parent child relationships in the data extraction process.

- Four **ADASTRIP** exits:

  1. ZONEDEC Formats the negative zoned decimal fields which cause a problem when the data is transferred to a different (usually ASCII) platform.

  2. STRIPZIP Combines the features of both **Zonedec** and **Zipditto** to convert the **Adastrip** extracts to compressed ASCII data, ready for transfer to the server platform.

  3. STRIPCSV Converts **Adastrip** output to a comma-separated-value (CSV) format ready for transfer to the server platform.

  4. .SRIPFMT Converts **Adastrip** output into formatted fixed column text.

- **ZIPDITTO** - A raw data compression utility for the mainframe, this works best with textual data.

ZIPDITTO

Synopsis

Nowadays, there is often a requirement to move data from one platform to another. In particular, when creating a data warehouse, large amounts of data need to be moved from the mainframe to the data warehouse server. It is one problem to ready the data on the mainframe, a second problem to transfer it and a third to load it into the data warehouse tables.

To solve the first problem, programs can be written to extract the data from the database. In the case of ADABAS, ADASTRIP can be used. The second part is simply a case of using FTP, or the like, to transfer the data to the data warehouse server. The third part is in the realm of the data warehouse software.

Typically the lion's share of the time is spent transmitting the data. Before transmitting the data it can be compressed and then decompressed on arrival at the data warehouse server. The caveat is that it takes time to compress the data, so the compression rate must be played off against elapsed time.

Below is a comparison showing the times required to transmit the raw data compared to that required to compress it up front using InfoZip's zip utility for S/390 and **ZIPDITTO**:

Product	Action	Elapsed Mins.	Total Mins.	Compression Rate
FTP	FTP uncompressed data	35:42	35.42	0%
Infozip and FTP	Zip FTP compressed data	28:47 1:40	 30.27	96%
Zipditto and FTP	Zip FTP compressed data Unzip	4:18 1:40 5:56	 11:54	96%

As can be seen, the rate of compression is the same as that achieved by InfoZip's zip utility but in 14% of the time.

ZIPDITTO was specifically written to handle columnar data and this is how the rate of compression is achieved in the short elapsed time. The data is also translated into ASCII.

Sample Job

```
//ZIPDITTO   JOB REGION=0M,NOTIFY=&SYSUID
//*
//ONE      EXEC PGM=ZIPDITTO
//STEPLIB  DD DSN=ZIPDITTO.LOAD,DISP=SHR,BLKSIZE=32760
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSUT1   DD DSN=ZIPDITTO.SOURCE(ZIPDITTO),DISP=SHR
//SYSUT2   DD DSN=DONALD.TEMP,SPACE=(TRK,(15,15)RLSE),UNIT3390,
//         DISP=NEW,CATLG),RECFM=VB,LRECL=27994,BLKSIZE=27998
//
```

ADASTRIP EXITS

Synopsis

ZONEDEC will sort out the negative zoned decimal fields, which cause a problem when the data is transferred to a different platform. For each record, ZONEDEC will examine each zoned decimal field (TYPE U). The sign of the value, which is the zone of the last digit, is checked. Positive numbers are x'F' which is quite acceptable because the digit is represented correctly and is ready for translation. Negative numbers, unfortunately, are indicated by x'D' in the zone portion on the last digit. This means that the last digit is represented by one of the characters "}JKLMNOPQR" (x'D0' to x'D9'). On recognising a zoned decimal field with a negative sign, ZONEDEC will correct the zone and insert a minus sign in the first character location. If the first character was not a zero, ZONEDEC will abend to prevent the loss of any data. To circumvent this use **Adastrip**'s LENGTH card to increase the width of the output field. The output files can be either variable or fixed record format.

STRIPZIP combines the features of both **Zonedec** and **Zipditto** to convert the Adastrip output into compressed ASCII data. This means that the output dataset is relatively small and that the transmission time is relatively short. The output file must be variable record format.

STRIPCSV converts **Adastrip** output to a comma-separated-value format. The first row in the output file is the heading record listing the names of the columns. Negative zoned decimal numbers are catered for and decimal points can also be inserted. Date and Time fields can also be converted to one of a variety of formats. A dataset containing the additional information must be supplied. PE and MU fields must be handled with care. An **Adastrip** INDEX card must be supplied to force a fixed number of occurrences, and headings must be supplied for each occurrence. The output file must be variable record format.

STRIPFMT converts **Adastrip** output to a fixed-column text format. This exit is similar to STRIPCSV in the way the data can be formatted except that the output fields are always fixed length.

ZONEDEC

Zoned Decimal Values

ADASTRIP is often used to extract data from an ADABAS file, which is destined for a PC. The ASCII environment and the nature of the PC operating system and applications place some restrictions of the type of data, which is sent. For example binary numbers will certainly cause a headache. Luckily, they are few and far between except for the ISN, which hopefully is not needed in the new environment. More common are packed or zoned decimal numbers. ADASTRIP can be asked to convert packed decimals to unpacked or zoned decimals. Positive zoned decimals are fine because ADABAS uses a sign of x'F' which, fortuitously, is the required zone for a displayable digit. But what about negative numbers?

The sign of a negative number is x'D'. This means that the last digit of a negative zoned decimal number will be one of the values x'D0' to x'D9', or one of }, J, K, L, M, N, O, P, Q or R. This value is acceptable in the IBM mainframe environment but not on the PC. Each field will have to be handled individually, the last character taken off, translated, put back on and the entire value negated.

A simpler solution is to use ZONEDEC (or ESTRIPZD for ESTRIP). This can be achieved by simply coding the ADASTRIP card:

```
ddname EXIT ZONEDEC
```

Each unpacked (or zoned) decimal field in the record will be examined. If the sign is negative, the zone of the last digit will be changed to x'F' to make the last digit displayable and a minus sign will be placed in the position of the first byte. This will only be done if the first byte is a zero. If not, ZONEDEC will abend with the following message:

Sign overwrite field XX ISN ???????????

The ISN field will only be inserted in the message if it was present in the output. To circumvent this abend, ADASTRIP can be requested to make the length of the field one byte longer.

STRIPZIP

Compressing the output data

The exit STRIPZIP (or ESTRIPZP for ESTRIP), will compress the data output by ADASTRIP. The compression method used is the same as performed by ZIPDITTO. (For more information on ZIPDITTO, read the help notes for that utility.)

Before the data is compressed all zoned decimal numbers are examined. Any negative zoned decimals are handled as described in the ZONEDEC help notes.

Use this exit by coding the following ADASTRIP card:

```
ddname EXIT STRIPZIP
```

Note: The data definition referred to by ddname must have variable record format and a logical record length equal to the length of the original record plus 2. All multiple value fields and periodic group must have ADASTRIP INDEX overrides so that the length of the original ADASTRIP record is fixed.

Customised Translate Tables

The translation from EBCDIC to ASCII is performed before the compression takes place. Although a comprehensive translate table is used to convert input file there is always a requirement for some special translation to occur. An attempt has been made to cater for all printable characters while all other characters are translated to an ASCII question mark. It is prudent to allow the user to change this strict rule and allow customisation of this translate table.

The default table is supplied as member E2A in the source library. This member can be copied to another member, which may then be changed to suit local requirements. To run STRIPZIP with a customised translate table simply add a data definition statement with ddname E2A to your job.

An example follows:

```
//STRIPZIP JOB MSGCLASS=X,NOTIFY=&SYSUID
//*
// EXEC PGM=STRIP,PARM=ccccccccccccccccccc,REGION=0M
//STEPLIB DD DSN=ADASTRIP.V304.LOAD,DISP=SHR,BLKSIZE=32760
// DD DSN=ESTRIP.LOAD,DISP=SHR
//STDUMP1 DD DSN=ADASAV6.F135,DISP=SHR,BUFNO=60
//STPRINT DD SYSOUT=*
//STRIPONE DD UNIT=SYSDA,
// SPACE=(TRK,(1500,150),RLSE),RECFM=VB,
// BLKSIZE=27998,LRECL=27994,DSN=TEMP.ONE,DISP=(,CATLG)
//E2A    DD DSN=ESTRIP.SOURCE(custom),DISP=SHR
//STMESS DD SYSOUT=*
//STPARM DD *
 MODE DUMP
 STRIPONE FILE 135
 STRIPONE LIMIT 1000
 STRIPONE FIELD AA AB AC AD AE AF AP BF BG AL AT
 STRIPONE TYPE U AB AL AT AP BG
 STRIPONE INDEX 10 BE AP
 STRIPONE NORMALISE
 STRIPONE EXIT STRIPZIP
/*
//SYSUDUMP DD SYSOUT=*
```

where *custom* is the member name of the customised translate table.

STRIPCSV

Comma Separated Values

Using the exit STRIPCSV (or ESTRIPCS for ESTRIP), the output data can be formatted into a comma-separated-value file. This file can be transferred to a PC for use as a .csv input file for many different applications. Actually a tab delimiter separates the values by default. The entire file can be translated from EBCDIC to ASCII, rendering the data readily usable. The first line of the output file contains the column headings, which must be supplied by the user.

Simply code the ADASTRIP EXIT card as follows:

```
ddname EXIT STRIPCSV
```

or use the name ESTRIPCS when using ESTRIP.

The output file must have variable record format with a logical record length large enough to contain the longest record.

Another data definition must by supplied. The ddname must be CSVDATA and it must contain cards in the following format:

```
|----+----1----+----2----+----3----+----4----+---
dddddddd  sn p ccccccccccccccccccccccccccccccc
```

where	dddddddd	is the ddname matching the ADASTRIP output file.
	sn	is the ADABAS short name of the field.
	p	is the formatting option. See below for more detail.
	cccc...	is the column heading. Upper and lower case are accepted but no blanks between words.

The formatting option p must be one of the following:

blank	for no special formatting required.
digit	for number of decimal places for numeric or binary fields. Trailing zeroes will be removed.
D	for NATURAL date field. The value will be converted to the format CCYY-MM-DD
I	for a NATURAL date field. Note that it is a lowercase 'i'. The value will be converted to CCYYMMDD
T	for a NATURAL time field. The value will be converted to CCYY-MM-DD HH-II-MM.T
c	for a NATURAL time field. Note that it is a lower case 'c'. The value will be converted to CCYYMMDDHHIIMMT
I	to format a N8 or P5 field as CCYY-MM-DD or a N6 or P4 field as YY-MM-DD
o	N6 field in the format YYMMDD will be converted to CCYYMMDD.

The DATEWINDOW option must be supplied to use this format.

u N6 field in the format MMDDYY will be converted to CCYYMMDD.
The DATEWINDOW option must be supplied to use this format.

e N6 field in the format DDMMYY will be converted to CCYYMMDD.
The DATEWINDOW option must be supplied to use this format.

U N8 field in the format MMDDCCYY will be converted to CCYYMMDD.

E N8 field in the format DDMMCCYY will be converted to CCYYMMDD.

It is imperative that the order of the cards follow the order of the fields in the ADABAS FDT when not using NORMALIZE and the order of the FIELD statement when using NORMALIZE.
For multiple values and periodic groups the correct number of entries must be supplied. No checking is performed.

To suppress the first line of column headings code the following option:

```
|----+---1----+---2----+---3----+---4----+--
dddddddd NOHEADER
```

where dddddddd is the ddname matching the ADASTRIP output file.

The default delimiter is the horizontal tab. To change this code the following before any field definition card:

```
|----+---1----+---2----+---3----+---4----+--
dddddddd DELIMITER s
```
or
```
dddddddd DELIMITER X'hh'
```

where dddddddd is the ddname matching the ADASTRIP output file.
s is the delimiter required. Any character can be coded here. Alternatively, code as X'hh' where hh is the hexadecimal representation of the desired delimiter.

The ISN will normally be converted to hexadecimal. With the following option it will be converted to a decimal number. When using the ADASTRIP NORMALISE option the PE and MU indices, which follow the ISN will also be rendered as decimal numbers and separated by a delimiter.

```
|----+---1----+---2----+---3----+---4----+--
dddddddd ISNBASE10
```

where dddddddd is the ddname matching the ADASTRIP output file.

An alphanumeric field, which is blank, will be rendered as a single blank in the output. Likewise a packed or unpacked field, which is zero, will be output as a single zero ("0"). Including the following option will suppress these values and two consecutive delimiters will be output.

```
|----+---1----+---2----+---3----+---4----+--
dddddddd NULLSUPPRESS
```

where dddddddd is the ddname matching the ADASTRIP output file.

To determine which century a 2 digit year belongs to when using the o, u and e formatting options it is compared to a window. If the 2 digit year is greater or equal to the window value the century is presumed to be the 20th century and the CC digits are set to '19'. If it is less then it is presumed to be in the 21st century and the CC digits are set to '20'. The default window value is 50.

To change this value, use the following directive:

```
|----+---1----+---2----+---3----+---4----+--
dddddddd DATEWINDOW yy
```

where dddddddd is the ddname matching the ADASTRIP output file.
yy is the window value required.

STRIPFMT

Formatted Output

Using the exit STRIPFMT (or ESTRIPFM for ESTRIP), the output data can be formatted into a fixed column file. The entire file can be translated from EBCDIC to ASCII, rendering the data readily usable on non-S390 Platforms.

Simply code the ADASTRIP EXIT card as follows:

```
ddname EXIT STRIPFMT
```

or use the name ESTRIPFM when using ESTRIP. The output file must have variable record format with a logical record length large enough to contain the longest record.

Another data definition must be supplied. The ddname must be FMTDATA and it must contain cards in the following format:

```
|----+----1----+----2----+----3----+----4----+---
dddddddd sn p ccccccccccccccccccccccccccccccc
```

where	dddddddd	is the ddname matching the ADASTRIP output file.
	sn	is the ADABAS short name of the field.
	p	is the formatting option. See below for more detail.
	cccc...	is treated as a comment.

The formatting option p must be one of the following:

blank for no special formatting required.
digit for number of decimal places for numeric data.
D for NATURAL date field. The value will be converted to the format CCYY-MM-DD
i for a NATURAL date field. *Note that it is a lower case 'i'.*
The value will be converted to CCYYMMDD
T for a NATURAL time field. The value will be converted to
N6 field in the format MMDDYY will be converted to CCYYMMDD.
The DATEWINDOW option must be supplied to use this format.
e N6 field in the format DDMMYY will be converted to CCYYMMDD.
The DATEWINDOW option must be supplied to use this format.
U N8 field in the format MMDDCCYY will be converted to CCYYMMDD.
E N8 field in the format DDMMCCYY will be converted to CCYYMMDD.

It is imperative that the order of the cards follow the order of the fields in the ADABAS FDT when not using NORMALIZE and the order of the FIELD statement when using NORMALIZE.

For multiple values and periodic groups the correct number of entries must be supplied. No checking is performed.

The ISN will normally be converted to hexadecimal. With the following option it will be converted to a decimal number. When using the ADASTRIP NORMALISE option the PE and MU indices, which follow the ISN will also be rendered as decimal numbers.

```
|----+----1----+----2----+----3----+----4----+---
dddddddd ISNBASE10
```

where dddddddd is the ddname matching the ADASTRIP output file.

To compress the output in ZIPDITTO form, use the following option. The output must be treated as a binary file and needs to be decompressed using unzditto. It is imperative that each record of the output file has the same length. This can be achieved by using the INDEX option of ADASTRIP to force a constant number of occurrences on each PE or MU field.

```
|----+----1----+----2----+----3----+----4----+---
dddddddd COMPRESS
    or
dddddddd ZIPDITTO
```

where dddddddd is the ddname matching the ADASTRIP output file.

To determine which century a 2 digit year belongs to when using the o, u and e formatting options it is compared to a window. If the 2 digit year is greater or equal to the window value the century is presumed to be the 20th century and the CC digits are set to '19'. If it is less then it is presumed to be in the 21st century and the CC digits are set to '20'. The default window value is 50. To change this value, use the following directive:

```
|----+----1----+----2----+----3----+----4----+---
dddddddd DATEWINDOW yy
```

where dddddddd is the ddname matching the ADASTRIP output file.
yy is the window value required.

The lengths of the formatted fields can be calculated using the following table:

FDT Format	FDT Length	STRIPFMT Format	Output Length
A	b	blank	b
B	b	blank	2b
B	1	0	4
B	2	0	6
B	3	0	8
B	4	0	11
B	5	0	13
B	6	0	16
B	7	0	18
B	8	0	20
B	1	digit>0	5
B	2	digit>0	6
B	3	digit>0	9
B	4	digit>0	12
B	5	digit>0	14
B	6	digit>0	17
B	7	digit>0	19
B	8	digit>0	21
P	b	blank	2b
P	b	digit	2b+1
P	4	D	10
P	4	i	8
P	7	T	21
P	7	t	15
P	7	c	15
P	4	I	8
P	5	I	10
U	b	blank	b+1
U	b	digit	b+2
U	6	I	8
U	6	o,e,u	8
U	8	I	10
U	8	E,U	8

ESTRIP

Synopsis

ADASTRIP, as many of our customers will attest to, is a wonderful and most useful tool. It does have some limitations though, which need to be solved using very intuitive techniques. The developers left the door open by allowing a user exit to be called on each record. This user exit has the final say on whether a record joins the stripped set of data or not. It can also change fields in the record. But, as with every 'catch-all', there are drawbacks. The exit will necessarily need to be written in Assembler language. COBOL is really a non-starter, as the return code which signals whether the record must be written to the output dataset must be set. This cannot be done without divine intervention. Secondly, there are problems if this COBOL routine is to be called many millions of times.

So, to really use ADASTRIP, one needs the services of an Assembler programmer. This is often not possible, or if it can be done the installation is left with a maintenance thorn.

ESTRIP is an ADASTRIP extender. ESTRIP will enable the user to strip data from a file in which records are dependant on a previous strip or some external data source. This means that once a "parent" file has been stripped, the "child" files can be stripped so that related records end up in the output file.

There are two components to ESTRIP. The main component is a program called ESTRIP, which handles the input cards and reading of any external data. It also calls ADASTRIP. The second component, ESTRIPX, is an ADASTRIP user exit. A parameter card requesting this user exit must be present in the ADASTRIP input to allow ESTRIP to work.

The main routine, ESTRIP, will read the parameter cards. These are either stored in memory for later use by the exit, ESTRIPX, or acted upon, as in the case of reading, sorting and creating the in memory tables.

Alternatively, using ESTRIPZD as the exit will combine the functionality of ESTRIPX and ZONEDEC. Using ESTRIPZP combines ESTRIPX and STRIPZIP, ESTRIPCS combines ESTRIPX and STRIPCSV, while ESTRIPFM combines ESTRIPX and STRIPFMT.

Operations

Data definitions

The data definition statements are as for ADASTRIP, but must also include:

ESTPARM	Card images supplying instructions to eSTRIP.
ESTPRINT	Message listing.
Seqfile	Input file of external data in fixed record format, or output file to contain saved keys.
SYSOUT	Message output for SORT (if the eSTRIP READ instruction is used).

ESTRIP Help Notes

ESTRIP is an ADASTRIP extender. ESTRIP will enable the user to strip data from a file in which the records are dependent on a previous strip or some external data source. This means that once a "parent" file has been stripped, the "child" files can be stripped so that related records end up in the output file.

There are two components to ESTRIP. The first (ESTRIP) handles the input cards and reading of any external data. It also calls ADASTRIP. The second component (ESTRIPX) is an ADASTRIP user exit. A parameter card requesting this user exit must be present in the ADASTRIP input to allow ESTRIP to work.

The data definition statements are as for ADASTRIP and must include:

ESTPARM card images supplying instructions for ESTRIP.
ESTPRINT message listing.
seqfile input file of external data in fixed record format.
SYSOUT message output for SORT (if the ESTRIP READ instruction is used).

The ESTRIP instructions are:

To save values from a field of stripped data:

```
ddname SAVE sn seqfile
```

where	ddname	is the ddname matching the ADASTRIP ddname of the file from whence the values must be saved.
	sn	is the ADABAS short name of the field.
	seqfile	is the ddname of the sequential file to which the values will be written.

To read values previously saved or an external data source:

```
ddname READ tn seqfile f
```

where	ddname	is the ddname matching the ADASTRIP ddname of the file. This parameter is actually ignored.
	tn	is the table name under which these values will be kept in memory.
	seqfile	is the ddname of the sequential file from which the values will be read. The dataset must have fixed record format.
	f	is the format of the data, C(haracter), B(inary), P(acked) or U(npacked) are the only values allowed.

To test a value against previously read values:

```
ddname TEST ffott
```

where	ddname	is the ddname matching the ADASTRIP ddname of the file.
	ff	is the ADABAS short name of the field.
	o	is the operator '=' to test for the presence of the value in the table and '#' to test for the absence of the value.
	tt	is the table name under which these values were loaded into memory.

Examples:

```
CHILD1 TEST AB=T1
OTHERS TEST AB#T1
```

To call a field exit: (See document #EXITF for more information)

```
ddname EXITF exitname sn
```

where	ddname	is the ddname matching the ADASTRIP ddname of the file.
	exitname	is the name of the exit. This module will be loaded at run-time.
	sn	is the ADABAS short name of the last field which must be passed to the exit.

Notes:

1. The record length of the sequential file will be taken from the length of the field when saved and will be used as the element length of the entries in the memory table.

2. Output messages from ESTRIP are written to SYSPRINT while messages from ESTRIPX are written to the job log. These messages appear when ESTRIPX determines that there is an error in the instructions given.

3. Any number of files can be stripped in a single invocation of ESTRIP. Obviously related "child" files have to be stripped in a subsequent invocation.

4. The values in the sequential file need not be sorted beforehand as this will be done when there are loaded into the memory table.

5. A field must be selected for stripping before it can be used in the instructions to ESTRIP.

Parameters

These are 80-byte, fixed record format data supplied via the ESTPARM data definition.

To save values from a field of stripped data:

ddname SAVE *sn seqfile*

Where	*ddname*	is the ddname matching the ADASTRIP ddname of the file from whence the values must be saved.
	sn	is the ADABAS short name of the field.
	seqfile	is the ddname of the sequential file to which the values will be written.

To read values previously saved or an external data source:

ddname READ *tn seqfile f*

Where	*ddname*	is the ddname matching the ADASTRIP ddname of the file. This parameter is actually ignored.
	tn	is the table name under which these values will be kept in memory. It must be two characters long.
	seqfile	is the ddname of the sequential file from which the values will be read. The dataset must have fixed record format.
	f	is the format of the data, C(haracter), B(inary), P(acked) or U(npacked) are the only values allowed.

To test a value against previously read values:

ddname TEST *sn=tn*

Where	*ddname*	Is the ddname matching the ADASTRIP ddname of the file.
	sn	Is the ADABAS short name of the field.
	tn	Is the table name under which these values were loaded into memory.

To perform an operation against a field:

ddname EXITF *module*

Where	*Ddname*	Is the ddname matching the ADASTRIP ddname of the file.
	module	Is the of the ADASTRIP exit.

Notes:

a. The record length of the sequential file will be taken from the length of the field when saved and will be used as the element length of the entries in the memory table.

b. Output messages from ESTRIP are written to SYSPRINT while messages from ESTRIPX are written to the job log. These messages appear when ESTRIPX determines that there is an error in the instructions given.

c. Any number of files can be stripped in a single invocation of ESTRIP. Obviously related "child" files have to be stripped in a subsequent invocation.

d. The values in the sequential file need not be sorted beforehand, as this will be done when they are loaded into the memory table.

e. A field must be selected for stripping before it can be used in the instructions to ESTRIP.

f. The ISN (##) can also be used as values to be saved or tested.

Sample Job

In this job, the parent file is stripped first and the values of the keys, which were stripped, are saved as foreign keys for the subsequent strip of the child file.

```
//ESTRIP   JOB REGION=0M,NOTIFY=&SYSUID
//*
//ONE      EXEC PGM=ESTRIP,PARM=XXXXXXXXXXXXXXXXXXXX
//STEPLIB  DD DSN=CCA.ADASTRIP.V302E.LOAD,DISP=SHR,BLKSIZE=32760
//         DD DSN=ESTRIP.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//STDUMP1  DD DSN=ADABAS.ADASAV,DISP=SHR,BUFNO=60
//STPRINT  DD SYSOUT=*
//STMESS   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//STRIPONE DD UNIT=SYSDA,SPACE=(CYL,(100.10),RLSE),RECFM=VB,
// BLKSIZE=32760,LRECL=700,DSN=DONALD.TEMP.ONE,DISP=NEW
//MAINKEY  DD UNIT=SYSDA,SPACE=(TRK,(100.10),RLSE),RECFM=FB,
// BLKSIZE=32760,LRECL=10,DISP=(NEW,PASS)
//STPARM   DD *
MODE DUMP
STRIPONE FILE 123
STRIPONE FIELD ## **
STRIPONE INDEX 1 HA
STRIPONE NORMALISE
STRIPONE TEST A SC.NE.C'OPS'
STRIPONE RULE A
STRIPONE EXIT ESTRIPX
/*
//ESTPRINT DD SYSOUT=*
//ESTPARM  DD *
STRIPONE SAVE AA MAINKEY
/*
//*
// IF RC=O THEN
//TWO      EXEC PGM=ESTRIP,PARM=XXXXXXXXXXXXXXXXXXXX
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DSN=CCA.ADASTRIP.V302E.LOAD,DISP=SHR,BLKSIZE=32760
//         DD DSN=ESTRIP.LOAD,DISP=SHR
//STDUMP1  DD DSN=ADABAS.ADASAV,DISP=SHR,BUFNO=60
//STPRINT  DD SYSOUT=*
//STMESS   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//STRIPTWO DD UNIT=SYSDA,SPACE=(CYL,(100.10),RLSE),RECFM=VB,
// BLKSIZE=32760,LRECL=700,DSN=DONALD.TEMP.TWO,DISP=NEW
//MAINKEY  DD DSN=*.ONE.MAINKEY,UNIT=SYSDA,DISP=(OLD,DELETE)
//STPARM   DD *
STRIPTWO FILE 124
STRIPTWO FIELD ## **
STRIPTWO EXIT ESTRIPX
/*
//ESTPRINT DD SYSOUT=*
//ESTPARM  DD *
STRIPTWO READ K1 MAINKEY C
STRIPTWO TEST AA=K1
/*
// ENDIF
```

Notes:

a) The authorisation code for ADASTRIP must be supplied as a parameter to ESTRIP. This parameter is passed on as-is to ADASTRIP.

b) All the testing features of ADASTRIP can still be used.

c) Step 1 can be simply an ADASTRIP job.

ESTRIPX

Field Exits

The ESTRIP field exit will be called by ESTRIPX, or one of the ESTRIPX flavours, on each field in the record until the field specified on the EXITF card has been reached.

The exit should be coded in assembly language. It must be AMODE 31.

The parameters passed to the exit are as follows:

A(FDTE)	pointer to the field element
A(field length)	pointer to the field length (halfword)
A(field value)	pointer to the field value
A(0)	dummy
A(TABLES)	pointer to table base

-Generating an exit to do table lookup.

A macro, GENEXITF, has been supplied to generate a user exit. Quite often, there is a requirement to check a foreign key, which comprises more than one field. The values for the table must be read into memory by using the READ command.

Code as follows:

```
name  GENEXITF field1,field2,...fieldn,TABLE=tabname
      END
```

where name is the name of the exit. It must start in column 1.
To make things simpler use the same name as the member.
fieldi defines each field in the order in which they appear
in the composite key. Each one is coded as follows: (sn,fmt,len)

where sn is the ADABAS short name of the field.
fmt is the format of the data, one of A, C, P, N, U, B.
len is the length in bytes of the field.

NB. Each triad must be enclosed in brackets.
tabname is the name of the table which must be search.

Subfields can be specified by coding the short name as follows:

```
sn(from,to)
```

where from is the first byte (counting left to right from 1).
to is the last byte of the subfield.

The length value, len, will be ignored and need not be coded if the second format is used.

The order in which the fields are specified must match the order of the values in the table, or vice-versa. The table will be searched when the last field specified in the list is presented to the exit. The fields are presented in the order of the ADABAS FDT. This means that the last field in the list must be the lowest field in the FDT.

Example:

```
TSTEXITF GENEXITF (BM,N,9),(CS(2,10),N,12),TABLE=T1
         END
```

To compile the exit see the sample job JCLASMXF.
Obviously, it would need to be tailored to the environment.

Installation Instructions

The extended strip package is distributed as a single zip file containing the following members:

1. eSTRIP Manual – PDF Documentation.
2. ESTRIP.<<ver>>.SOURCE.XMIT – XMIT format JCL, Macros and source library.
3. ESTRIP.<<ver>>.LOAD.XMIT – XMIT format load library.
4. ESTRIP.<<ver>>.ADAV7.SOURCE.XMIT – XMIT format JCL, Macros and source library for ADABAS V7 support.
5. ESTRIP.<<ver>>.ADAV7.LOAD.XMIT – XMIT format load library for ADABAS V7 support.
6. Release notes – PDF describing the release.

On the mainframe create two datasets as follows:

<<hlq>>.ESTRIP.VXYY.SOURCE.XMIT
<<hlq>>.EXTRIP.VXYY.LOAD.XMIT

DCB details for all files:

If support for ADABAS V7 is required then create the additional datasets as well:

<<hlq>>.EXTRIP.VXYY.ADAV7.SOURCE.XMIT
<<hlq>>.EXTRIP.VXYY.ADAV7.LOAD.XMIT

LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PS

Space requirements, allow 1 cylinder with expansion for 1 cylinder on a 3390 device type

The process to load the files is as follows:

- Transfer the two (or three) XMIT format files to the mainframe by a binary FTP of binary file transfer.(fixed record format, 80-byte records).

- In TSO issue the following command for the transferred source file:
- receive inds("*source dataset"*)

- Enter the following when asked to enter more options:
- da('*hlq.product.version*.SOURCE')

- Enter the following when asked to enter more options:
- da('*hlq.product.versionV7*.SOURCE')

- In TSO issue the following command for the transferred load file:

- receive inds(“sourceV7 *dataset”*)

- In TSO issue the following command for the transferred load file:
- receive inds(“*loadV7 dataset”*)

- Enter the following when asked to enter more options:
- da(‘*hlq.product.version*.LOAD’)

At this point there will be the JCL and the load library (and the V7 libraries) will be ready to use.

The Source library contains various example JCL members and parameters examples. There is also a job to apply the product code zap, JCLZAP. The product code is available from CCA Software.

eSTRIP and the Exits will not work without the product code zap being applied. The product code requires a CPUID. This is supplied from the output of the command D M=CPU. Note the whole output is required to generate the zap.

****