
SZ02-0301-12

SLiKZip

Industrial Grade Data Compression for z/OS platforms

USER'S GUIDE

Version 1 Release 11

October 2011

LICENSED MATERIAL

COPYRIGHT Australian Systems Engineering Pty. Ltd. 2002,2003,2004,2007,2008,2010,2011

This material always remains the property of Australian Systems Engineering Pty. Ltd. and must be used only in accordance with the terms of the agreement under which it is supplied.

Manual and Product identifiers.

This manual is identified by a publication number, **SZ02-0301**, combined with a revision level number (sometimes called a dash level) **-12**.

The licensed software product, of which this manual is a part, is identified by a **Product Name** or **Product Acronym** and an **Edition Number**. Edition numbers are made up of a **Version number** and a **Release number** within that version.

The material in this manual is intended to be used with **SLiKZiP Version 1 Release 11**. The edition number may appear in licensed material in abbreviated forms like the following: **V1R11**, and **V1.11**.

From time to time ASE may issue replacements for, or additions to, the pages in this manual.

Each time the entire manual is replaced by a later version then the revision level (the dash level) part of the publication identifier will be increased by 1.

Publication history:

Dash	product ver.rel	GA date	Highlights/added features
-0	SLiKZiP V1.1	Mar 2003	ZIP,UNZIP,AUDIT
-1	SLiKZiP V1.2	Oct 2003	GZIP
-2	SLiKZiP V1.4	Mar 2004	ZIP64, DEF64
-3	SLiKZiP V1.5	Nov 2004	AES 128/192/256 bit encryption
-4	SLiKZiP V1.5	Mar 2007	MASIS keyword
-5	SLiKZiP V1.5	Mar 2007	New messages
-7	SLiKZiP V1.9	May 2008	GZIP tape support,MTRANS,PLATFORM keywords
-10	SLiKZiP V1.10	Jun 2010	ISPF interface, Large Format Datasets, ZIP tape support
-11	SLiKZiP V1.11	Sep 2011	Storage of ZIP files as members of PDS/PDSE datasets
-12	SLiKZiP V1.11	Oct 2011	Document new keywords in the User's Guide

Related Publications

None

Published by

Australian Systems Engineering Pty. Ltd.

20 Andrew Street.

NORTHCOTE, Victoria 3070, Australia

telephone +61 3 9419 4200

email: help@slikzip.com

Trademarks

OS/390 and z/OS are registered trademarks of IBM Corp.

PKZIP and PKUNZIP are registered trademarks of PKWARE, Inc.

UNIX is a registered trademark of The Open Group

Windows is a registered trademark of Microsoft Corporation

WinZip is a registered trademark of WinZip Computing, Inc.

Contents

INTRODUCTION	11
WHAT'S NEW IN SLiKziP 1.11 (SEP 2011)	12
• <i>Support for ZIP'ing to and UNZIP'ing from members of a PDS or PDSE dataset.</i>	12
WHAT'S WAS NEW IN SLiKziP 1.10 (JUN 2010).....	12
• <i>ISPF (SLIKISPF) dialog interface to SLIKziP</i>	12
• <i>Support for ZIP'ing to and UNZIP'ing from tape datasets</i>	12
• <i>Support for Large Format sequential datasets</i>	12
• <i>New PARM= keywords, process options and built-in functions</i>	12
WHAT WAS NEW IN SLiKziP 1.9 (MAY 2008).....	13
• <i>Support for GZIP format files on tape</i>	13
• <i>Support MTRANS(..) keyword for custom member name translation</i>	13
• <i>Support PLATFORM(..) Keyword</i>	13
• <i>Support for AES encryption PASSWORD value in hex</i>	13
• <i>ZIP0213W warning messages when license key expires within 30 days</i>	13
• <i>Support for ampersand (&) in member names</i>	13
WHAT WAS NEW IN SLiKziP 1.5 (NOVEMBER 2004).....	13
• <i>AES 128/192/256 bit encryption support has been added</i>	13
WHAT WAS NEW IN SLiKziP 1.4 (MARCH 2004)	14
• <i>ZIP 64 support has been added</i>	14
• <i>DEFLATE 64 support has been added</i>	14
WHAT WAS NEW IN SLiKziP 1.2 (OCTOBER 2003).....	14
• <i>GZIP support was added</i>	14
WHAT WAS NEW IN SLiKziP 1.1 (MARCH 2003)	14
• <i>Original version of SLiKziP supporting "standard" ZIP files</i>	14
PRODUCT ACTIVATION	14
• <i>But what if I license the software</i>	15
SLiKziP AND ZIP TERMINOLOGY	15
• <i>ZIP datasets or Archives</i>	15
• <i>GZIP datasets or Archives</i>	15
• <i>ZIP member names, path, file-id, filename, extension</i>	15
ZIP DATASET CHARACTERISTICS AND LIMITATIONS	15
GZIP DATASET CHARACTERISTICS AND LIMITATIONS	16
GETTING STARTED WITH SLIKziP.....	17
REQUIRED JCL.....	17
BATCH EXAMPLES.....	18
• <i>Compress all members of a PDS to create an archive</i>	18
• <i>List the contents of an archive</i>	19
• <i>Uncompress from an archive into a sequential dataset</i>	19
• <i>Uncompress files to different output datasets</i>	19
• <i>Zipping different data types together</i>	20
• <i>Compress Multiple Sequential Datasets and Use Long File Names</i>	21
• <i>Create an archive stored in a pds member</i>	21
• <i>Uncompress from an archive stored in a pds member</i>	21
MORE EXAMPLES.....	22
• <i>Compress Text Files Destined for UNIX or LINUX Machines</i>	22
• <i>Compress One Text File to create a GZIP dataset</i>	22
• <i>Create Output Dataset Names Based on File Names in the Archive</i>	22
• <i>Translating € Characters Correctly</i>	23
PARAMETER AND CONTROL STATEMENT REFERENCE.....	25
PARAMETERS	25
• <i>Defining the process to be performed</i>	25

• <i>Overriding default DD names SYSPRINT and SYSIN</i>	25
• <i>CTLDD(ctlddname)</i>	26
• <i>PRTDD(prtdname)</i>	26
• <i>Generating additional information for problem diagnosis</i>	26
• <i>DEBUG</i>	26
CONTROL STATEMENT GENERAL SYNTAX.....	26
• <i>General Statement format</i>	26
• <i>Adding comments to statements</i>	27
• <i>Continuing a statement over multiple lines</i>	27
• <i>Case insensitivity</i>	27
CONTROL STATEMENT SUMMARY BY VERB	27
CONTROL STATEMENT REFERENCE	29
• <i>AUDIT – List the Contents of a ZIP dataset</i>	29
• <i>EXT – Define a set of Process Options for an “Extension”</i>	29
• <i>FROMDD – Specify an Input DD Name</i>	30
• <i>FROMDS – Specify Input Datasets</i>	30
• <i>GZIP – Compress Data to create a GZIP dataset</i>	31
• <i>SELECT – Select Data for Processing</i>	32
• <i>TODD – Specify an Output DD Name</i>	33
• <i>TODS – Specify an Output Dataset</i>	34
• <i>TRANSTAB – Define a translate table</i>	34
• <i>UNZIP – Uncompress Data</i>	35
• <i>ZIP – Compress Data into a ZIP dataset</i>	35
PROCESS OPTIONS REFERENCE	36
• <i>ADD</i>	37
• <i>AES</i>	37
• <i>AFUERR/NOAFUERR</i>	37
• <i>ALIASES/NOALIAS</i>	37
• <i>ASCII / NOASCII</i>	37
• <i>BINARY</i>	38
• <i>CR / NOCR</i>	38
• <i>CRLF / NOCRLF</i>	38
• <i>DEF64</i>	38
• <i>DETAILS</i>	38
• <i>EFFORT(n)</i>	38
• <i>EOF(...)</i>	38
• <i>EOR(...)</i>	38
• <i>EXT(...)</i>	39
• <i>FRESHEN</i>	39
• <i>FROMDD(...)</i>	39
• <i>FROMDS(...)</i>	39
• <i>LF / NOLF</i>	40
• <i>MASIS</i>	40
• <i>MEMTS(...)</i>	40
• <i>MNAME(...)</i>	40
• <i>MPATH/NOMPATH</i>	40
• <i>MTRANS(tablename)</i>	40
• <i>NODDESC</i>	40
• <i>OVERWRITE</i>	41
• <i>PAD(...)/NOPAD</i>	41
• <i>PASSWORD(...)</i>	41
• <i>PLATFORM(name)</i>	42
• <i>RDW</i>	42
• <i>REPLACE</i>	42
• <i>SELECT(...)</i>	42
• <i>SHOWHDR</i>	42
• <i>STRIP / NOSTRIP</i>	42

•	TEXT	43
•	TODD(...)	43
•	TODS(...)	43
•	TRANSTAB(tablename)	43
•	TRMOD(modname)	43
•	TRUNC/NOTRUNC	43
•	TZONE	43
•	UPDATE	44
•	USELOCALHEADERVERVALUES	44
•	WRAP/NOWRAP	44
•	ZDW	44
•	ZIP64	45
	BUILT-IN FUNCTIONS	45
•	Creating output dataset names for UNZIP operations	45
•	Creating archive member names for ZIP operations	45
•	Using Built-in Functions for generating names	45
•	Using Substrings and Segment numbers	45
•	Built-in Functions supported by SLiKZiP	46
•	Error message text specific to Built-in Function processing	47
	ALLOCATION OPTIONS REFERENCE	47
•	BLKSIZE(...)	47
•	BLOCK(nnnn)	48
•	CATALOG	48
•	CYLINDERS	48
•	DATACLAS(...)	48
•	DELETE	48
•	DIR(...)	48
•	DUMMY	48
•	DSNTYPE(LIBRARY/PDS/BASIC/LARGE)	48
•	EXPDT(yyyy/ddd)	48
•	KEEP	48
•	LABEL(...)	48
•	LIKE(...)	48
•	LRECL(...)	48
•	MGMTCLAS(...)	49
•	MOD	49
•	NEW	49
•	OLD	49
•	POSITION(...)	49
•	RECFM(...)	49
•	RELEASE	49
•	RETPD(nnnn)	49
•	SHR	49
•	SPACE(pri[,sec])	49
•	STORCLAS(...)	49
•	SUBSYS(name,parm1,parm2,...)	49
•	TRACKS	49
•	UNCATALOG	49
•	UNIT(...)	50
•	VOLUME(...)	50
	FAQ – FREQUENTLY ASKED QUESTIONS	51
	WHAT’S AN EXTENSION?	51
	WHAT RECORD FORMAT SHOULD I USE FOR MY ZIP DATASET?	51
	HOW DO I CREATE A GZIP DATASET WITH SLiKZiP?	51
	HOW DO I TELL SLiKZiP I WANT TO UNZIP A GZIP DATASET?	51
	FROMDD(..) PROCESS OPTION VERSUS THE FROMDD CONTROL STATEMENT	52

WHY DO I SEE OPTION ASCII EVEN THOUGH I DIDN'T SPECIFY ASCII OR TEXT?	53
HOW CAN I AUTOMATICALLY TRANSFER FILES USING FTP?.....	53
HOW DO I SPLIT A PARM STRING INTO MULTIPLE LINES AND NOT GET JCL ERRORS?	54
IS SLIKZIIP DIFFERENT FROM ISPZIP?	55
APPENDIX A – SLIKISPF - THE ISPF DIALOG APPLICATION	56
THE SLIKISPF DIALOG APPLICATION ALLOWS USERS TO:	56
SLIKISPF TUTORIAL	56
MISCELLANEOUS SLIKISPF OPERATIONAL NOTES:	56
APPENDIX B – TRANSLATE TABLES IN LOAD MODULES	57
TRANSLATE TABLE FORMAT.....	57
HOW TRANSLATION WORKS FOR EACH CHARACTER OR BYTE.....	58
CREATING YOUR OWN TRANSLATE TABLE.....	58
APPENDIX C – TRANSLATE TABLES IN CONTROL STATEMENTS	61
TRANSTAB CONTROL STATEMENT SYNTAX.....	61
TABLE MODIFICATION RECORDS.....	62
APPENDIX D – SLIKZIP MESSAGES	63
• ZIP0007I.....	63
• ZIP0008E.....	63
• ZIP0014E.....	63
• ZIP0015E.....	63
• ZIP0016E.....	64
• ZIP0018W.....	64
• ZIP0019W.....	64
• ZIP0024E.....	64
• ZIP0025E.....	64
• ZIP0026E.....	64
• ZIP0027E.....	64
• ZIP0028E.....	65
• ZIP0029E.....	65
• ZIP0030E.....	65
• ZIP0031E.....	65
• ZIP0032E.....	65
• ZIP0033E.....	65
• ZIP0034E.....	65
• ZIP0035E.....	65
• ZIP0036E.....	66
• ZIP0037E.....	66
• ZIP0038E.....	66
• ZIP0039E.....	66
• ZIP0040E.....	66
• ZIP0041W.....	66
• ZIP0042E.....	67
• ZIP0043E.....	67
• ZIP0044E.....	67
• ZIP0057I.....	67
• ZIP0058E.....	67
• ZIP0059E.....	67
• ZIP0070E.....	67
• ZIP0071E.....	68
• ZIP0072I.....	68
• ZIP0073E.....	68
• ZIP0074E.....	68
• ZIP0075E.....	68
• ZIP0076I.....	68

• ZIP0077I.....	69
• ZIP0078I.....	69
• ZIP0079I.....	69
• ZIP0080E.....	69
• ZIP0081E.....	69
• ZIP0082E.....	69
• ZIP0083I.....	69
• ZIP0084E.....	69
• ZIP0085E.....	70
• ZIP0086E.....	70
• ZIP0087I.....	70
• ZIP0088I.....	70
• ZIP0089E.....	70
• ZIP0090E.....	70
• ZIP0091I.....	70
• ZIP0092I.....	71
• ZIP0093I.....	71
• ZIP0094E.....	71
• ZIP0099E.....	71
• ZIP0100E.....	71
• ZIP0101E.....	72
• ZIP0102E.....	72
• ZIP0103I.....	72
• ZIP0104E.....	72
• ZIP0105E.....	72
• ZIP0106E.....	72
• ZIP0107E.....	72
• ZIP0108E.....	72
• ZIP0112E.....	73
• ZIP0113E.....	73
• ZIP0114E.....	73
• ZIP0115E.....	73
• ZIP0116E.....	73
• ZIP0117I.....	73
• ZIP0118I.....	73
• ZIP0119E.....	73
• ZIP0120E.....	73
• ZIP0121W.....	74
• ZIP0122E.....	74
• ZIP0123E.....	74
• ZIP0124E.....	74
• ZIP0125E.....	74
• ZIP0126I.....	74
• ZIP0127E.....	74
• ZIP0128E.....	75
• ZIP0129E.....	75
• ZIP0130E.....	75
• ZIP0131E.....	75
• ZIP0132E.....	75
• ZIP0133E.....	75
• ZIP0134E.....	75
• ZIP0135E.....	76
• ZIP0136E.....	76
• ZIP0137E.....	76
• ZIP0138E.....	76
• ZIP0139E.....	76

• ZIP0140E.....	76
• ZIP0141E.....	76
• ZIP0142E.....	76
• ZIP0143E.....	77
• ZIP0145I.....	77
• ZIP0146I.....	77
• ZIP0147I.....	77
• ZIP0148E.....	77
• ZIP0149I.....	77
• ZIP0150I.....	77
• ZIP0151I.....	77
• ZIP0152E.....	78
• ZIP0153E.....	79
• ZIP0154E.....	79
• ZIP0155E.....	80
• ZIP0156E.....	80
• ZIP0157E.....	80
• ZIP0158E.....	80
• ZIP0159E.....	80
• ZIP0160I.....	80
• ZIP0161E.....	80
• ZIP0162I.....	81
• ZIP0163I.....	81
• ZIP0164I.....	81
• ZIP0165W.....	81
• ZIP0166I.....	81
• ZIP0167W.....	81
• ZIP0168I.....	81
• ZIP0169E.....	81
• ZIP0170E.....	82
• ZIP0171E.....	82
• ZIP0172E.....	82
• ZIP0173I.....	82
• ZIP0174I.....	82
• ZIP0175E.....	82
• ZIP0176E.....	83
• ZIP0177E.....	83
• ZIP0178I.....	83
• ZIP0179E.....	83
• ZIP0180E.....	83
• ZIP0181E.....	83
• ZIP0182E.....	83
• ZIP0183E.....	83
• ZIP0184E.....	84
• ZIP0185W.....	84
• ZIP0186W.....	84
• ZIP0187E.....	84
• ZIP0188I.....	84
• ZIP0189I.....	84
• ZIP0190I.....	84
• ZIP0191E.....	84
• ZIP0192I.....	85
• ZIP0193E.....	85
• ZIP0194E.....	85
• ZIP0195I.....	85
• ZIP0196I.....	85

• ZIP0197E.....	85
• ZIP0198E.....	86
• ZIP0199E.....	86
• ZIP0200E.....	86
• ZIP0201E.....	86
• ZIP0202E.....	86
• ZIP0203E.....	86
• ZIP0204E.....	86
• ZIP0205E.....	87
• ZIP0206I.....	87
• ZIP0207I.....	87
• ZIP0208I.....	87
• ZIP0209I.....	87
• ZIP0210I.....	87
• ZIP0211I.....	88
• ZIP0212E.....	88
• ZIP0213W.....	88
• ZIP0214I.....	88
• ZIP0215W.....	88
• ZIP0216I.....	88
• ZIP0217I.....	89
• ZIP0218E.....	89
• ZIP0219I.....	89
• ZIP0220E.....	89
• ZIP0221E.....	89
• ZIP0222E.....	89
• ZIP0223E.....	89
• ZIP0224E.....	90
• ZIP0225E.....	90
• ZIP0226E.....	90

dataset already exists, so its existing RECFM, LRECL and BLKSIZE attributes will be used. The **REPLACE** keyword causes existing members of **MY.DATA.CNTL** to be replaced .

More Examples” on page 21 shows further examples of SLiKZIP usage.

“Parameter and Control Statement” on page 25 provide the reference descriptions for all control statements and for the PARM= operand of the JCL EXEC statement.

“FAQ – Frequently Asked Questions” on page 51 answers some common questions.

“Appendix A – SLIKISPF - the ISPF dialog application” on page 56 introduces the ISPF interface.

”Appendix B – Translate Tables in Load Modules” on page 56 shows you how to create and use your own translate tables.

“Appendix C – Translate Tables in Control Statements” on page 61 shows another way to manage translate tables.

“Appendix D – SLiKZIP Messages” on page 63 describes each message issued by SLiKZIP.

What's new in SLiKZIP 1.11 (Sep 2011)

- Support for ZIP'ing to and UNZIP'ing from members of a PDS or PDSE dataset.

Zip files can now be stored as members of a PDS or PDSE dataset. The TODS statement of a ZIP operation can now specify dataset(member) as the target location for the output zip file. A TODD statement can refer to a ddname allocated to a dataset(member). Similarly when UNZIP'ing a zip file stored in a PDS/PDSE, the FROMDS statement can specify dataset(member) as the location of the input zip file and a TODD statement can refer to a ddname allocated to a dataset(member). Note that zip files in a PDS/PDSE can not be updated. If they are changed they must be overwritten.

What's was new in SLiKZIP 1.10 (Jun 2010)

- ISPF (SLIKISPF) dialog interface to SLIKZIP

A comprehensive ISPF dialog application provides inspection of, extraction from, creation of ZIP datasets online. The dialog also allows automatic submission of batch jobs to complete hefty compression or uncompression tasks.

- Support for ZIP'ing to and UNZIP'ing from tape datasets

The ZIP function can now create a completely new tape-resident ZIP dataset. The UNZIP function can extract members from a multi-volume tape-resident ZIP dataset..

- Support for Large Format sequential datasets

Large Format sequential datasets (larger than 65536 tracks per volume) are supported as input to UNZIP and output from ZIP operations.

- New PARM= keywords, process options and built-in functions

&FRDS builtin function now represents the input dataset name with periods separating segments.

&FRPA builtin function now represents the input dataset name with periods replaced by “/” so that it looks like a path/fileid.

EXEC PGM=SLIKZIP,PARM='HELP=*msgid*' will print a description of the message for which *msgid* is the prefix. Eg: PARM='HELP=ZIP0213'.

Control statement “HELP *msgid*” will print a description of the message for which *msgid* is the prefix. Eg: “HELP ZIP0213”.

DEMOKEY= may now be specified in the PARM= field when SLIKZIP is run as a batch program. This is an alternative to providing a DEMOKEY statement in the control statement file and may be useful when the only reason for a control statement file was to contain the DEMOKEY statement.

The FROMDS statement now supports VOL and UNIT allocation parameters.

What was new in SLiKZIP 1.9 (May 2008)

- Support for GZIP format files on tape

Tape-resident GZIP format datasets can now be uncompressed directly from tape. A GZIP operation can also create a GZIP format dataset directly to tape..

- Support MTRANS(..) keyword for custom member name translation

The MTRANS(..) keyword has been added to allow the default member name EBCDIC/ASCII translation to be overridden. Users can now specify their own translate table to be used when translating member names. This allows member names in ZIP and GZIP datasets to contain special or national characters not normally found in the default ASCII and EBCDIC character sets.

- Support PLATFORM(..) Keyword

The PLATFORM(..) keyword allows the SLIKZIP user to override the default setting (MVS) and choose a value that is correctly supported by a third party program processing the ZIP file.

- Support for AES encryption PASSWORD value in hex

When AES encryption or decryption is used, the password may optionally be expressed as a string of hexadecimal digits. See the description of the PASSWORD keyword for more detail.

- ZIP0213W warning messages when license key expires within 30 days

Warning message ZIP0213W will be issued if the license key expires within the next 30 days.

- Support for ampersand (&) in member names

Warning message ZIP0213W will be issued if the license key expires within the next 30 days.

What was new in SLiKZIP 1.5 (November 2004)

- AES 128/192/256 bit encryption support has been added

SLiKZIP's AES encryption support complies with the AE-2 standard introduced by Winzip in January 2004.

ZIP0207I DEMOKEY value missing or invalid for today's date, terminating

- But what if I license the software...

When, if, you take out a license for SLiKZIP from ASE you will not have to supply DEMOKEY statements to use the software. Instead ASE will supply you with a 12-month key. During the duration of your license term ASE will automatically supply a replacement 12-month key 45 days prior to the expiry of the current key that you are using. SlikZip will automatically start warning users, via message ZIP0213W, when the current key is less than 31 days from expiring.

SLiKZIP and ZIP Terminology

- ZIP datasets or Archives

The compressed datasets read and written by SLiKZIP are in the ZIP format (or the GZIP format, see below). While on platforms other than OS/390 and z/OS it may be normal to refer to ZIP files, within the context of OS/390 and z/OS we prefer to use the term ZIP dataset and to say that ZIP datasets contain members. If you use the term "ZIP file" in communication with ASE we will expect it to be a synonym for ZIP dataset.

The term ARCHIVE used within this manual may be taken as meaning ZIP dataset.

- GZIP datasets or Archives

GZIP file format is similar to, but not the same as, the ZIP format. SLiKZIP (1.2 onward) does support the GZIP file format (SLiKZIP 1.1 did not). Note that SLiKZIP (1.2 onward) will only uncompress the first member of a multi-member GZIP file.

- ZIP member names, path, file-id, filename, extension

Each ZIP dataset contains one or more members, each with a member name that typically must conform to the file naming rules of any other platforms the ZIP dataset is to be processed on. Usually this means that the member name can consist of up to 255 ASCII characters. When the member name contains "/" or "\" characters we say that a path is present, the path being all characters prior to the rightmost slash, and that the characters following each slash are path segments. The rightmost path segment is also called the file-id. A file-id may itself be segmented by one or more periods (.). When this is the case, the rightmost of these segments may be referred to as the file-extension, or simply the extension, while all of the file-id prior to that final period is called the file-name.

```
INVOICING/JULY/INVOICE.LST
```

This member name consists of three segments **INVOICING**, **JULY** and **INVOICE.LST**. **INVOICE.LST** is the file-id, **INVOICE** is the file-name and **LST** is the extension. The path is **INVOICING/JULY**

```
TOTAL.COSTS.DAT
```

This member name contains a single segment, **TOTAL.COSTS.DAT**. The rightmost (and only) segment, **TOTAL.COSTS.DAT**, is the file-id, **TOTAL.COSTS** is the file-name and **DAT** is the extension. There is no path.

ZIP dataset characteristics and limitations

SLiKZIP supports ZIP datasets on disk or tape. Speaking in z/OS file system terms, the DSORG (dataset organisation) of a ZIP dataset must be PS (Physical Sequential) or PO (Partitioned). Record

format can be F, FB, FBA, FBM, V, VB, VBA, VBM or U with any record length or block size that is valid for the chosen record format. Note that when SLiKZIP creates a ZIP dataset with RECFM=FB, the last record is padded with binary zeroes. Some UNZIP programs on other platforms may reject ZIP files with this padding. ASE recommend that you choose RECFM=U and a large BLKSIZE value for any ZIP datasets that you create. Remember that the RECFM of the ZIP dataset has no relationship to the RECFM of any member contained within the ZIP dataset. For DSORG PO datasets a member name must be specified when the dataset is allocated.

SLiKZIP can **ADD** to or **UPDATE** an existing disk-resident ZIP dataset if it stored in a PS dataset.

SLiKZIP can **OVERWRITE** an existing disk dataset, including an existing ZIP dataset, or it can create a completely new ZIP dataset.

The so-called "standard" ZIP dataset architecture imposes limitations that SLiKZIP must observe: A maximum ZIP dataset size of 4GB, maximum uncompressed byte count of any ZIP member of 4GB and a maximum of 65,535 members within a single ZIP dataset. These limitations are dictated by the use of 32-bit binary fields for byte counts and file offsets within the "standard" ZIP file architecture.

The addition of ZIP64 support in SLiKZIP 1.4 removes the "4GB limits" imposed by the "standard" ZIP file architecture. See the ZIP64 Process Option. You must request ZIP64 format, it won't happen just because one of the 4GB limits mentioned above is exceeded during a ZIP operation!

GZIP dataset characteristics and limitations

SLiKZIP supports GZIP datasets on disk or tape. The DSORG (dataset organisation) of a GZIP dataset must be PS (Physical Sequential) or PO (Partitioned). Record format can be F, FB, FBA, FBM, V, VB, VBA, VBM or U with any record length or block size that is valid for the chosen record format. Note that when SLiKZIP creates a GZIP dataset with RECFM=FB, the last record is padded with binary zeroes. Some UNZIP programs on other platforms may reject GZIP files with this padding. ASE recommend that you choose RECFM=U and a large BLKSIZE value for any GZIP datasets that you create.

SLiKZIP cannot add to or update an existing GZIP dataset. GZIP operations always create a new dataset or overwrite any existing dataset. If the target GZIP dataset already exists you must use the **OVERWRITE** or **REPLACE** process options.

The GZIP dataset architecture does allow for the 4GB file size limits that apply to "standard" ZIP datasets to be exceeded. The GZIP Record Descriptor holds only a 32-bit CRC and a 32-bit uncompressed byte count for the member. The GZIP RFC allows for this byte count to overflow, or wrap, when it exceeds 4G. This feature may cause GZIP format to be chosen in preference to ZIP format when very large datasets are being processed. Alternatively, with the addition of ZIP64 support in SLiKZIP 1.4, the ZIP file format may be used instead of GZIP.

A "multi-member" GZIP dataset just consists of two or more GZIP datasets concatenated together. There is no equivalent of the Central Directory found at the end of ZIP datasets. Furthermore, the only way to find out where the compressed member data ends is to effectively uncompress it, a potentially expensive way to locate the second or subsequent "member" of a "multi-member" GZIP dataset, keeping in mind that the format tends to be preferred for very large datasets. Consequently ASE has chosen to limit SLiKZIP at this time to being able to UNZIP or AUDIT only the first member of an existing GZIP dataset.

SLiKZIP will create multi-member GZIP datasets if multiple input datasets or members are selected for a GZIP operation.

- The SELECT statement specifies which ZIP members will be uncompressed. The TRANSTAB operand indicates that the dynamic translate table named EURO_UNZIP will be used when translating from ASCII to EBCDIC.
- The TODS statement names the output dataset(s).

Translate tables are described in greater detail in “” on page 63 and in “Appendix C – Translate Tables in Control Statements” on page 61.

FROMDS Used to define an input source (or sources) by dataset name for the current process (AUDIT, UNZIP or ZIP). For ZIP processes any SELECT statements must follow this statement. It may contain a variety of process options for the input data.

When the process is ZIP, multiple sets of FROMDS and SELECT statements may be used. When the process is ZIP the dataset name may contain wildcards and represent a mask or filter to identify cataloged dataset(s) that are to be compressed.

GZIP Specifies that data is to be compressed to create or replace a GZIP dataset. This statement may contain all the process options required to fully specify the compress operation. Alternatively it may consist of nothing more than the verb GZIP and be followed by a TODD or TODS statement to define the output GZIP dataset and one or more FROMDD or FROMDS statements with associated SELECT statements, defining input data sources.

SELECT Used to identify PDS members to be compressed. May be used regardless of the input data source to specify how to build the ZIP or GZIP member name and other process options for the selected data.

TODD Used to define an output target by dd name for the current process (ZIP, GZIP or UNZIP). It may contain a variety of process options for the output data.

When the process is ZIP or GZIP there can only be a single output target specification.

When the process is UNZIP the TODD statement must follow any SELECT statements selecting ZIP members to be written to the specified dd name.

TODS Used to define an output target by dataset name for the current process (ZIP or UNZIP). It may contain a variety of process options for the output data.

When the process is ZIP or GZIP there can only be a single output target specification.

When the process is UNZIP the TODS statement must follow any SELECT statements selecting ZIP members to be written to the specified dataset. The output dataset name may be a pattern that includes symbolic references that will be replaced at execution time to create the actual dataset name to be used.

TRANSTAB Used to define a translate table that can be referenced by the TRANSTAB process option on various control statements. Placed typically at the start of the statement file, the TRANSTAB statement must precede any control statement that references the table being defined. See "Appendix C – Translate Tables in Control Statements" on page 61.

UNZIP Specifies that members of a ZIP or GZIP dataset are to be uncompressed. This statement may contain all the process options required to fully specify the uncompress operation. Alternatively it may consist of consist of nothing more than the verb UNZIP and be followed by a FROMDD or FROMDS statement to define the input ZIP or GZIP dataset and one or more SELECT and TODD or TODS statements defining output data targets. Note that the program automatically detects which type of input dataset is being processed, ZIP or GZIP. Note also that SLiKZiP can only process the first member of a GZIP dataset during an UNZIP operation.

ZIP Specifies that data is to be compressed to create, update or replace a ZIP dataset. This statement may contain all the process options required to fully specify the compress operation. Alternatively it may consist of consist of nothing more than the verb ZIP and be followed by a TODD or TODS statement to define the output ZIP dataset and one or more FROMDD or FROMDS statements with associated SELECT statements, defining input data sources.

DEMOKEY Required by the demo version of SLiKZiP to provide the program with an enabling key for the current date. This control statement is not required in each step if the LICENSE

process is followed instead. This would normally be the case when either a trial agreement or a valid end user license agreement has been entered into between the licensee and ASE. Refer to the LICENSE member of the SLiKziP SAMPLIB dataset.

Control Statement Reference

In the following control statement descriptions, optional operands are enclosed in square brackets [...]. A vertical bar | separates alternatives.

- **AUDIT – List the Contents of a ZIP dataset**

Use the AUDIT control statement to list the contents of an archive. It has this format:

```
AUDIT [FROMDS(...) | FROMDD(...)] [SELECT(...)] [DETAIL] [SHOWHDR]
```

Placement: An AUDIT statement defines the start of a new process. There are no specific placement requirements.

Either **FROMDS** or **FROMDD** must be specified to define the source. Use FROMDS to specify a dataset name or FROMDD to specify the name of a DD statement in your JCL. If neither of these operands is provided on the AUDIT statement then you must provide a following FROMDD or FROMDS control statement to define the input ZIP dataset or archive.

FROMDS can be abbreviated to FRDS and FROMDD to FRDD.

Use the **SELECT** option to limit the list to files matching the selection argument. See on page 31 for a complete description of the selection argument.

Use the **DETAIL** option to show extra details on the audit output.

When the SHOWHDR option is specified, headers from the ZIP dataset's central directory will be displayed.

- **EXT – Define a set of Process Options for an “Extension”**

Use the EXT control statement to group together various process options. When the extension is referenced in later control statements, the process option set defined on the EXT statement will be inherited as defaults. Specific options may be overridden by specifying them on the relevant SELECT statement.

```
EXT extname [process options]
```

Where *extname* is the extension name associated with the process options.

Placement: The EXT statement must occur earlier in the control statement file than any FROMDS, TODS or SELECT statement or other operand that may include a reference to extname. The safest thing is to place all EXT statements prior to any AUDIT, UNZIP or ZIP statement.

These process options may be specified on an EXT control statement: **AFUERR/NOAFUERR**, **ALIASES**, **ASCII/NOASCII**, **BINARY**, **CR/NOCR**, **CRLF/NOCRLF**, **DEF64**, **EFFORT(n)**, **EOF(...)**, **EOR(...)**, **LF/NOLF**, **MEMTS(...)**, **MNAME(...)**, **MPATH/NOMPATH**, **MASIS**, **NODDESC**, **PAD(...)/NOPAD**, **PLATFORM(...)**, **RDW**, **REPLACE**, **STRIP/NOSTRIP**, **TEXT**, **TODD(...)**, **TODS(...)**, **TRANSTAB(...)**, **TRMOD(...)**, **TRUNC/NOTRUNC**, **WRAP/NOWRAP**, **ZDW** and **ZIP64**. **MTRANS(...)** can be specified on an EXT control statement used for a ZIP/GZIP operation. Refer to “Process Options” on page 36 for descriptions of individual process options. Process options set here will override those set on **FROMDD**, **FROMDS**, **TODD** or **TODS** control statements preceding the SELECT statement that explicitly or implicitly names this EXT statement.

When the **TODS(...)** keyword option is present on the statement, allocation options may be specified on the EXT statement. Refer to “Allocation Options” on page 47 for descriptions of allocation options.

When the **TODS(..)** or **TODD(..)** keyword option is present on the statement and the output dataset already exists but is going to be overwritten, then structural allocation options such as **RECFM(..)**, **LRECL(..)** and **BLKSIZE(..)** may be specified.

- **FROMDD – Specify an Input DD Name**

Use the FROMDD control statement to specify an input data source.

FROMDD *ddname* [process options]

Where *ddname* is the DD name.

Placement: The FROMDD statement defines an input data source. A process (ZIP, AUDIT or UNZIP) must have been identified by a prior statement or an invocation parameter or have been implied by the program name.

FROMDD can be abbreviated FRDD.

These process options may be associated with this DD name: **ADD, AFUERR/NOAFUERR, ALIASES, ASCII/NOASCII, BINARY, CR/NOCR, CRLF/NOCRLF, DEF64, DETAILS, EFFORT(n), EOF(...), EOR(...), FRESHEN, LF/NOLF, MEMTS(...), MNAME(...), MTRANS(...), MASIS, MPATH/NOMPATH, NODDESC, PAD(...)/NOPAD, PLATFORM(...), RDW, SHOWHDR, STRIP/NOSTRIP, TEXT, TRANSTAB(...), TRMOD(...), TRUNC/NOTRUNC, UPDATE, WRAP/NOWRAP, ZDW** and **ZIP64**. Refer to “Process Options” on page 36 for descriptions of individual process options.

See also process option FROMDD(..).

- **FROMDS – Specify Input Datasets**

Use the FROMDS control statement to specify an input data source.

FROMDS *datasetname* [process options]

Where *datasetname* is the dataset name. For ZIP operations a list of names may be specified by separating the names with commas and enclosing the list in parentheses.

For ZIP operations only: Wildcards (?, * and **) can be used to make *datasetname* a filter or mask that is to be used at the commencement of the operation to search the catalog for input datasets to be compressed.

Placement: Placement rules are the same as for FROMDD.

FROMDS can be abbreviated FRDS.

These process options may be associated with this dataset: **ADD, AFUERR/NOAFUERR, ALIASES, ASCII/NOASCII, BINARY, CR/NOCR, CRLF/NOCRLF, DEF64, DETAILS, EFFORT(n), EOF(...), EOR(...), FRESHEN, LF/NOLF, MEMTS(...), MNAME(...), MTRANS(...), MASIS, MPATH/NOMPATH, NODDESC, PAD(...)/NOPAD, PLATFORM(...), RDW, SHOWHDR, STRIP/NOSTRIP, TEXT, TRANSTAB(...), TRMOD(...), TRUNC/NOTRUNC, UPDATE, WRAP/NOWRAP, ZDW** and **ZIP64**. Refer to “Process Options” on page 36 for descriptions of individual process options.

See also process option FROMDS(..).

- **GZIP – Compress Data to create a GZIP dataset**

Use the GZIP control statement to specify a new process to be performed.

GZIP [TODS(...)|TODD(...)] [FROMDS(...)|FROMDD(...)] [process options]

Placement: A GZIP statement defines the start of a new process. There are no specific placement requirements. A GZIP statement can also be used in the PARM= string in batch when PGM=SLIKZIP.

Either **TODS(..)** or **TODD(..)** process options may be specified on the GZIP statement to define the target GZIP dataset. If neither TODS nor TODD is present then the target GZIP dataset will be set according to the next TODS or TODD control statement for the process.

Either **FROMDS(..)** or **FROMDD(..)** process options may be specified on the GZIP statement to define the source dataset(s). If neither FROMDS nor FROMDD is present then the source dataset(s) will be set according to following FROMDS or FROMDD control statement(s). Preceding EXT control statements can also set the source dataset(s).

These process options may be specified on a GZIP control statement: **ADD**, **ALIASES**, **ASCII/NOASCII**, **BINARY**, **CR/NOCR**, **CRLF/NOCLRF**, **DETAILS**, **EFFORT(n)**, **EOF(...)**, **EOR(...)**, **EXT(...)**, **FRESHEN**, **LF/NOLF**, **MNAME**, **MTRANS(...)**, **MASIS**, **MPATH/NOMPATH**, **NODDESC**, **OVERWRITE**, **PASSWORD(...)**, **PLATFORM(...)**, **RDW**, **REPLACE**, **SELECT(...)**, **SHOWHDR**, **STRIP/NOSTRIP**, **TEXT**, **TRANSTAB(...)**, **TRMOD(...)**, **UPDATE** and **ZDW**. Refer to “Process Options” on page 36 for descriptions of individual process options.

These process options, if present, are ignored when creating a GZIP dataset: **PASSWORD(..)**, **ADD**, **FRESHEN** and **UPDATE**.

When **TODS(...)** is specified, allocation options may also be specified if that dataset is to be created. Refer to “Allocation Options” on page 47 for descriptions of allocation options.

Unlike ZIP dataset processing, SLiKZIP does not support adding to, or updating, an existing GZIP dataset. If the target GZIP dataset already exists then you must use the **OVERWRITE** (or **REPLACE**) process option to confirm to SLiKZIP that it may overwrite it. Here are three examples showing correct placement of the OVERWRITE keyword or its abbreviation, OVER:

```
//STEP1 EXEC PGM=SLIKZIP,
// PARM='GZIP TODS(MY.GZ) OVER FRDS(MY.HUGE.TEXT.DATASET) TEXT LF'
//SYSPRINT DD SYSOUT=*
//

//STEP1 EXEC PGM=SLIKZIP
//OUT DD DISP=SHR,DSN=MY.GZ
//SYSPRINT DD SYSOUT=*
GZIP TODD(OUT) OVERWRITE +
FRDS(MY.HUGE.TEXT.DATASET) TEXT LF
//

//STEP1 EXEC PGM=SLIKZIP
//OUT DD DISP=SHR,DSN=MY.GZ
//SYSPRINT DD SYSOUT=*
GZIP
TODD OUT OVERWRITE /* allow replacement of existing dataset
FRDS MY.HUGE.TEXT.DATASET TEXT LF
//
```

- **SELECT – Select Data for Processing**

Use the SELECT control statement to specify library or archive members to be processed and, optionally, to provide a member name construction template.

SELECT *selectiontext* [process options]

Where *selectiontext* is used to match input library member names (ZIP) or input archive member names (UNZIP or AUDIT). *Selectiontext* can also be a comma-separated list of terms enclosed in parentheses.

Placement: There must be a prior FROMDD or FROMDS statement or operand to identify the data source for SELECT. There may be multiple SELECT statements following a FROMDD or FROMDS statement.

ZIP dataset member names are regarded as segmented by "/" (or "\") characters. The final (right-most) segment is called the "file-id" and it may be further segmented by "." characters into "filename" and "extension". Often ZIP member names may consist of just a file-id. To match the member names present within a ZIP dataset the selection arguments can be an exact match, or can contain wildcards "?", "*" and "**".

- ? to match a single character actually present
- * to match omitted leading characters in the name segment, or match omitted trailing characters in the name segment.
- ** to match all intervening characters, including segment separators.

Matching takes place segment by segment. The ? wildcard may be used within a single segment to match a single character actually present in the member name. The * wildcard may be used within a single segment to match one or more unspecified leading, or trailing, characters. The ** wildcard may be used to match a string of unspecified characters. The string may include the segment separator character "/".

Example: assuming a ZIP member name DEV/DATA/01JAN2002.DAT

```
select DEV/DATA/??JAN*.*      will select this member
select DEV/*/*JAN*.*         will select this member
select **/*.*                 will select this member
```

In the example below members of the input partitioned dataset (PDS) whose names begin "RACF" will be selected and placed in the archive with member names consisting of their PDS member name plus ".MAC".

Because MNAME(...) was omitted, the program tries to obtain filename and extension components, with which to build a default MNAME, from any "selection text" that was specified. The filename part of this selection text is invalid for this purpose because it contains wildcard characters (the trailing * in RACF*) and instead the program uses its own internal default, "*". The extension MAC is taken from the extension part of the selection text. The rules for converting MNAME(...) strings into archive member names treat the first * encountered as a symbol for the current PDS member name. "*.MAC" thus becomes in each case, "*pdsmembername*.MAC"

```
//STEP1 EXEC PGM=ZIP,
// PARM='TODD(ZIPMAC) FRDS(hlq.MACLIB(RACF*.MAC)) TEXT'
//SYSPRINT DD SYSOUT=*
//ZIPMAC DD DISP=(,CAT),DSN=hlq.MACZIP,SPACE=(TRK,(2,5)),UNIT=WORK,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=0)
//
See SAMPLIB member ZIP11
```

In this second example the input sequential dataset is placed in the archive with a member name of "REPORT40.TXT".

There is of course no "selection" role when the input is a sequential dataset but the other function, providing an alternative source for the member name construction template when MNAME(...) is omitted, is still required, and for convenience the same structural syntax is supported.

```
//STEP1 EXEC PGM=ZIP,
// PARM=( 'FRDS(hlq.REPORT40(REPORT40.TXT)) TEXT' ,
// 'TODS(hlq.REPORT40.ZIP) OVERWRITE' )
//SYSPRINT DD SYSOUT=*
//
See SAMPLIB member ZIP12
```

In the third example the input sequential dataset is placed in the archive with a member name identical to the input dataset name but with periods replaced by slashes.

When the input is sequential and MNAME(...) is omitted, the program assumes MNAME() and MPATH. The archive member name is built by taking the input dataset name and translating all the periods (.) to slashes (/).

```
//STEP1 EXEC PGM=ZIP,
// PARM=( 'FRDS(hlq.REPORT40) TEXT' ,
// 'TODS(hlq.REPORT40.ZIP) OVERWRITE' )
//SYSPRINT DD SYSOUT=*
//
See SAMPLIB member ZIP13
```

SELECT may be abbreviated to SEL.

These process options may be specified on a SELECT control statement: **ADD**, **AFUERR/NOAFUERR**, **ALIASES**, **ASCII/NOASCII**, **BINARY**, **CR/NOCR**, **CRLF/NOCLRF**, **DEF64**, **DETAILS**, **EFFORT(n)**, **EOF(...)**, **EOR(...)**, **EXT(...)**, **FRESHEN**, **LF/NOLF**, **MEMTS(...)**, **MNAME(...)**, **MTRANS(...)**, **MASIS**, **MPATH/NOMPATH**, **NODDESC**, **PAD(...)/NOPAD**, **PASSWORD(...)**, **PLATFORM(...)**, **RDW**, **SHOWHDR**, **STRIP/NOSTRIP**, **TEXT**, **TODD(...)**, **TODS(...)**, **TRANSTAB(...)**, **TRMOD(...)**, **TRUNC/NOTRUNC**, **UPDATE**, **WRAP/NOWRAP**, **ZDW** and **ZIP64**. **REPLACE** can be specified for an UNZIP operation. Refer to "Process Options" on page 36 for descriptions of individual process options. Process options set here will override those options set on preceding **FROMDD**, **FROMDS**, **TODD** or **TODS** control statements.

When the **TODS(...)** keyword option is present on the statement, allocation options may also be specified if the dataset is to be created. Certain options, such as RECFM, LRECL and BLKSIZE, may be specified if the dataset already exists but is going to be overwritten. Refer to "Allocation Options" on page 47 for descriptions of allocation options.

See also process option EXT(..), The safest thing is to place all EXT statements prior to any AUDIT, UNZIP or ZIP statement.

- **TODD – Specify an Output DD Name**

Use the TODD control statement to specify a DD name as the data destination.

TODD *ddname* [process options]

Where *ddname* is the DD name.

Placement: The TODD statement defines a data destination. For ZIP operations, TODD may either precede or follow any SELECT statements or operands or implicit selection arguments. For UNZIP operations, TODD must follow any SELECT statements or operands that specify the member(s) to be extracted to the destination defined by TODD. For example:

UNZIP FROMDD(ZIP)	/* process is UNZIP and source is //ZIP DD
SELECT ABC*.TXT text	/* select certain text members
SELECT DEF*.TXT text	/* select other text members
TODD TEXT	/* write selected text members to //TEXT DD ...
SELECT ABC*.OBJ BIN	/* select certain object members
TODD OBJ	/* write selected object members to //OBJ DD ...

These process options may be associated with this DD name: **ADD, AFUERR/NOAFUERR, ALIASES, ASCII/NOASCII, BINARY, CR/NOCR, CRLF/NOCLRF, DEF64, DETAILS, EFFORT(n), EOF(...), EOR(...), FRESHEN, LF/NOLF, MNAME(...), MTRANS(...), MASIS, MPATH/NOMPATH, NODDESC, OVERWRITE, PAD(...)/NOPAD, PASSWORD(...), PLATFORM(...), RDW, REPLACE, SHOWHDR, STRIP/NOSTRIP, TEXT, TRANSTAB(...), TRMOD(...), TRUNC/NOTRUNC, UPDATE, WRAP/NOWRAP, ZDW** and **ZIP64**. Refer to “Process Options” on page 36 for descriptions of individual process options.

See also [process option TODD\(..\)](#).

- **TODS – Specify an Output Dataset**

Use the TODS control statement to specify a destination dataset.

TODS *dataset* [process options]

Where *dataset* is the dataset name.

For UNZIP operations only: Built-in functions can be used to make the dataset name a pattern from which to generate an actual dataset name at run time.

Placement: Placement rules are the same as those for TODD.

These process options may be associated with this dataset: **ADD, AFUERR/NOAFUERR, ALIASES, ASCII/NOASCII, BINARY, CR/NOCR, CRLF/NOCLRF, DEF64, DETAILS, EFFORT(n), EOF(...), EOR(...), FRESHEN, LF/NOLF, MEMTS(...), MNAME(...), MASIS, MPATH/NOMPATH, MTRANS(...), NODDESC, OVERWRITE, PAD(...)/NOPAD, PASSWORD(...), PLATFORM(...), RDW, REPLACE, SHOWHDR, STRIP/NOSTRIP, TEXT, TRANSTAB(...), TRMOD(...), TRUNC/NOTRUNC, UPDATE, WRAP/NOWRAP, ZDW** and **ZIP64**. Refer to “Process Options” on page 36 for descriptions of individual process options.

Allocation options may also be specified if the dataset is to be created. Refer to “Allocation Options” on page 47 for descriptions of allocation options. Allocation options must not precede the TODS(..) keyword operand.

See also [process option TODS\(..\)](#).

- **TRANSTAB – Define a translate table**

Use the TRANSTAB control statement to define a new translate table.

TRANSTAB *tablename* FOR(ZIP|UNZIP) [COPY(*copyname*)] [LOAD(*loadname*)] [VERIFY]

Placement: Transtab statements must occur prior to any reference to the table that they define. Table Modification Records, if present, must immediately follow the TRANSTAB statement.

Tablename is a 1 to 40 character name chosen by you to refer to this table in following control statements. To refer to this table on another control statement you would use this table name in a TRANSTAB(...) process option. The table you create here is dynamic and only exists in memory for the life of the current SLiKZiP invocation.

The **FOR(...)** operand is required. It indicates whether this table is to be used for ZIP (compression) or UNZIP (uncompression) operations.

The **COPY(..)** operand may be used to cause this table to be initialized by taking a copy of another table already defined to SLiKZIP by prior TRANSTAB statements. SLiKZIP's default translate table may be copied by specifying **COPY(ASCII)**. If neither COPY(...) nor LOAD(...) is specified the table is initialized as an empty table.

The **LOAD(..)** operand may be used to cause this table to be initialized by loading the named program and copying part of it as follows: For ZIP (compress) operations, the first 256 bytes of the loaded program is copied into the new translate table. For UNZIP (uncompress) operations the second 256 bytes is copied. If neither COPY(...) nor LOAD(...) is specified the table is initialized as x'00....00'.

The **VERIFY** option causes this translate table to be printed after any Table Modification Records have been processed.

Refer to "Appendix C – Translate Tables in Control Statements" on page 61 for details on creating your own translate tables and for a description of Table Modification Records.

- **UNZIP – Uncompress Data**

Use the UNZIP control statement to uncompress data from a ZIP dataset.

UNZIP [FROMDS(...)|FROMDD(...)] [TODS(...)|TODD(...)] [process options]

Placement: An UNZIP statement defines the start of a new process. There are no specific placement requirements.

Either **FROMDS(...)** or **FROMDD(...)** may be specified to define the source ZIP dataset. If neither FROMDS(...) nor FROMDD(...) is present then the source ZIP dataset will be set according to a following FROMDS or FROMDD control statement. EXT control statements can also set the source ZIP dataset.

Either **TODS(...)** or **TODD(...)** may be specified to define the target dataset. If neither TODS(...) nor TODD(...) is present then the target dataset will be set according to a following TODS or TODD control statement. EXT control statements can also identify the data destination. Built-in functions can be used to make the target dataset name a variable expression to be resolved at run time.

These process options may be specified on an UNZIP control statement: **ALIASES**, **ASCII/NOASCII**, **BINARY**, **CR/NOCR**, **CRLF/NOCLRF**, **DETAILS**, **EOF(...)**, **EOR(...)**, **EXT(...)**, **LF/NOLF**, **MEMTS(...)**, **MNAME(...)**, **MASIS**, **MPATH/NOMPATH**, **MTRANS(...)**, **NOSTRIP**, **PAD(...)/NOPAD**, **PASSWORD(...)**, **PLATFORM(...)**, **RDW**, **REPLACE**, **SELECT(...)**, **SHOWHDR**, **STRIP/NOSTRIP**, **TEXT**, **TRANSTAB(...)**, **TRMOD(...)**, **TRUNC/NOTRUNC**, **WRAP/NOWRAP** and **ZDW**. Refer to "Process Options" on page 36 for descriptions of process options.

When **TODS(...)** is specified, allocation options may also be specified if that dataset is to be created. Refer to "Allocation Options" on page 47 for descriptions of allocation options.

- **ZIP – Compress Data into a ZIP dataset**

Use the ZIP control statement to specify a new process to be performed.

ZIP [TODS(...)|TODD(...)] [FROMDS(...)|FROMDD(...)] [process options]

Placement: A ZIP statement defines the start of a new process. There are no specific placement requirements. A ZIP statement can also be used in the PARM= string in batch when PGM=SLIKZIP.

Either **TODS(...)** or **TODD(...)** process options may be specified on the ZIP statement to define the target ZIP dataset. If neither TODS(...) nor TODD(...) is present then the target ZIP dataset will be set according to the next TODS or TODD control statement for the process.

Either **FROMDS(...)** or **FROMDD(...)** process options may be specified on the ZIP statement to define the source dataset(s). If neither FROMDS(...) nor FROMDD(...) is present then the source dataset(s) will be set according to following FROMDS or FROMDD control statement(s). Preceding EXT control statements can also set the source dataset(s).

These process options may be specified on a ZIP control statement: **ADD**, **AFUERR/NOAFUERR**, **ALIASES**, **ASCII/NOASCII**, **BINARY**, **CR/NOCR**, **CRLF/NOCRLF**, **DEF64**, **DETAILS**, **EFFORT(n)**, **EOF(...)**, **EOR(...)**, **EXT(...)**, **FRESHEN**, **LF/NOLF**, **MNAME**, **MASIS**, **MPATH/NOMPATH**, **MTRANS(...)**, **NODDESC**, **OVERWRITE**, **PASSWORD(...)**, **PLATFORM(...)**, **RDW**, **REPLACE**, **SELECT(...)**, **SHOWHDR**, **STRIP/NOSTRIP**, **TEXT**, **TRANSTAB(...)**, **TRMOD(...)**, **UPDATE**, **ZDW** and **ZIP64**. Refer to "Process Options" on page 36 for descriptions of individual process options.

When **TODS(...)** is specified, allocation options may also be specified if that dataset is to be created. Refer to "Allocation Options" on page 47 for descriptions of allocation options.

If the target ZIP dataset already exists, and that dataset is a valid a ZIP dataset, then by default SLiKZIP will attempt to add new member(s) to it, or replace existing members with the same names if the UPDATE or FRESHEN process options were specified. Use the **OVERWRITE** (or **REPLACE**) process option to instruct SLiKZIP to overwrite an existing target dataset. Here are three examples showing correct placement of the OVERWRITE keyword or its abbreviation, OVER:

```
//STEP1 EXEC PGM=SLIKZIP,
// PARM='ZIP TODS(MY.ZIP) OVER FRDS(MY.TEXT.DATASET) TEXT'
//SYSPRINT DD SYSOUT=*
//

//STEP1 EXEC PGM=SLIKZIP
//OUT DD DISP=SHR,DSN=MY.ZIP
//SYSPRINT DD SYSOUT=*
ZIP TODD(OUT) OVERWRITE +
FRDS(MY.TEXT.DATASET) TEXT
//

//STEP1 EXEC PGM=SLIKZIP
//OUT DD DISP=SHR,DSN=MY.ZIP
//SYSPRINT DD SYSOUT=*
ZIP
TODD OUT OVERWRITE /* force replacement of existing dataset
FRDS MY.TEXT.DATASET TEXT
//
```

Process Options Reference

Process options are used to control or modify compress and uncompress operations. The options described here may appear on most control statements. When specified, they become the default for following control statements. For example, when the **TEXT** option (a short way of specifying **ASCII** and **CRLF**) is specified on a **FROMDS(...)** statement, **ASCII** and **CRLF** become the defaults for any following **SELECT** statements unless overridden by other relevant options such as **BINARY**, **NOASCII**, **NOCR**, **NOLF**, **LF** and **CR**.

These options may be abbreviated as long as they do not conflict with other options that are valid on the current control statement. For example on a **SELECT** control statement, **TRANSTAB(...)** may be abbreviated to **TRANS(...)** or **TRA(...)**, but not **TR(...)** due to ambiguity with **TRMOD(...)**. However, on a **TODS** control statement **TRA(...)** cannot be used due to ambiguity with the **TRACKS** allocation option.

Ordering of Process Options on a single statement

Process Options are processed left to right, from the start of a statement to the end. Some process options can override or modify the effect of earlier process options. For example, the CRLF process option will be overridden by a subsequent (i.e. placed to the right) LF process option on the same statement. If the order of these keywords were reversed then a different option set would result. To verify which options are in effect, see the ZIP0178I message in the SLiKZIP control listing.

- **ADD**

All selected data is to be added to the archive. If a file of the same name already exists in the archive, it will not be replaced. Note that **ADD** is assumed when none of **ADD**, **FRESHEN** or **UPDATE** is specified. This option is assumed when a GZIP dataset is being created.

- **AES**

During a ZIP operation, the current member is to be encrypted in compliance with the AE-2 (Advanced Encryption Standard – 2) specification. Encryption key length will be 128, 192 or 256 bits, depending upon the length of the password string specified via the **PASSWORD(..)** process option.

During an UNZIP operation, if the member has been encrypted according to the AE-2 (or AE-1) standards then SLiKZIP will automatically recognize this and use AES decryption.

For more information about AE-2 see: http://www.winzip.com/aes_info.htm

- **AFUERR/NOAFUERR**

During a ZIP operation this keyword is used to specify how errors are to be treated when processing an existing ZIP dataset. If the **AFUERR** keyword is specified and one of the following errors occur:

- An **ADD** operation is being processed and the member already exists;
- An error occurs during an **UPDATE** operation and the member can't be updated;

then a return code of 4 will be set and the current operation will be terminated.

The **NOAFUERR** keyword can be specified to ignore these errors and have the current ZIP process continue with the next input file/member. The default is **AFUERR**.

Note that when the operation is **FRESHEN** (as opposed to **ADD** or **UPDATE**) and the member does not already exist in the ZIP dataset, a new member will not be added. (**FRESHEN** only updates members that already exist in the ZIP file). While this event is not an error, the program will document the event by issuing a ZIP0192I message. The current ZIP operation continues.

- **ALIASES/NOALIAS**

Used when compressing a load module to tell the program to search for and include data for any **ALIAS** names for the selected member. When this member is uncompressed any **ALIAS** entries for it will also be set in the target library. **ALIASES** is the default. Specify **NOALIAS** to ignore alias entries.

- **ASCII / NOASCII**

Enables data translation. During a ZIP or GZIP operation SLiKZIP will translate from EBCDIC to ASCII prior to compression. During an UNZIP operation, SLiKZIP will translate from ASCII to EBCDIC after uncompression. Use **NOASCII** to override this option when higher level control statements imply **ASCII**. Alternative translate tables may be specified using either the **TRANSTAB(...)** or **TRMOD(...)** options. See also **BINARY**.

- **BINARY**

Disables data translation.

- **CR / NOCR**

Indicates that records are terminated with carriage return. During a ZIP or GZIP operation a carriage return character (x'0d') is added to the end of every record. During an UNZIP operation the x'0d', is recognised as an end-of-record indicator is then discarded. Use **NOCR** to cancel this option when higher level control statements imply **CR**.

- **CRLF / NOCRLF**

Indicates that records are terminated with carriage return and line feed character combination (x'0d0a'). During a ZIP or GZIP operation x'0d0a' is added to the end of every record. During an UNZIP operation, x'0d0a', is recognised as an end-of-record indicator and is then discarded. Use **NOCRLF** to cancel this option when higher level control statements imply **CRLF**.

- **DEF64**

Indicates that the DEFLATE 64 compression algorithm is to be used by SLIKZIP when compressing data. DEFLATE 64 may offer a better compression ratio or a space saving over the default DEFLATE algorithm. Perform your own tests to determine if there is an advantage to using DEF64. Supported from SLIKZIP 1.4.

Note that some older UNZIP programs may not support this enhanced ZIP file compression algorithm and therefore may be unable to unzip your data.

- **DETAILS**

During an AUDIT or UNZIP operation, any additional information present in ZIP dataset directory entries will be displayed on the control listing.

- **EFFORT(n)**

Specify a number between 1 and 6 to indicate the amount of effort to be expended when performing a compress operation. The default is **EFFORT(1)**. With higher numbers, a somewhat greater compression may be achieved at the cost of greater CPU time consumption. The degree of improvement is dependent on the data.

- **EOF(...)**

Specifies the end-of-file character sequence. During a ZIP or GZIP operation the specified character sequence is added at the end of the file ie. after the final record. During an UNZIP operation the characters are simply recognised as indicating end of record and are discarded (note that end-of-member is actually determined by the end of the compressed datastream). The character sequence is specified as one, two or three pairs of characters. Valid character pairs are CR, LF, CZ, EM, NU, +A to +Z or any hex digit pair. See the **EOR(...)** process option for a detailed description of the valid character pairs.

- **EOR(...)**

Specifies the end-of-record character sequence. During a ZIP or GZIP operation the specified character sequence is added to the end of each record (except the final record). During an UNZIP operation, its presence is used to signify a record boundary and is then discarded. The character sequence is specified as one, two or three pairs of characters. Valid character pairs are CR, LF, CZ, EM, NU, +A to +Z or any hex digit pair.

- CR – carriage return: x'0d' also known as ctrl-M.

- LF – line feed: x'0a' also known as ctrl-J.
- CZ – end of file: x'1a' also known as ctrl-Z.
- EM – end of message x'19' also known as ctrl-Y.
- NU – null x'00'
- Use a plus sign followed by any letter from a to z to mimic the ctrl-key plus letter-key operations familiar in many platforms. Here are some examples:
 - +A x'01' ctrl-A
 - +B x'02' ctrl-B
 - +J line feed x'0a' ctrl-J
 - +M carriage return x'0d' ctrl-M
 - +Z end of file x'1a' ctrl-Z
- Use any pair of hex digits: 00, 01, ..., 10, 11, ..., FE, FF.

Some of the most commonly required EOR(..) combinations also have their own process option keywords:

- CRLF for EOR(CRLF);
- CR for EOR(CR);
- LF for EOR(LF).

- **EXT(...)**

May be used on a SELECT statement. The 1- to 8-character name of a user-supplied EXT statement can be specified on a SELECT statement to incorporate the processing options defined on that EXT statement. Note that any options specified on the SELECT statement itself will take precedence over the same option also present on an EXT statement.

- **FRESHEN**

Update files already in the archive that match the selected data. New files will not be added to the archive. Note that **ADD** is assumed when neither **ADD**, **FRESHEN** nor **UPDATE** is specified. This option is not supported when a GZIP dataset is being created.

- **FROMDD(...)**

Specifies a list of DD names that define input data sources. When two or more DD names are specified, commas must separate them. Each DD name may optionally have a selection argument enclosed in parentheses eg. **FROMDD(DD1(*.TXT))** where DD name DD1 has a selection argument of ***.TXT**. When used on a ZIP statement the ddnames designate datasets to be compressed. When used on an UNZIP statement a single ddname only is allowed and it will designate the input ZIP dataset. **FROMDD** may be abbreviated to **FRDD**. The DD names must be present in your JCL.

- **FROMDS(...)**

Specifies a list of datasets that define input data sources. When two or more datasets are specified, commas must separate them. Each dataset name may optionally have a selection argument enclosed in parentheses e.g. **FROMDS(SYS1.MACLIB(IHA*.MAC))** where the dataset SYS1.MACLIB has a selection argument of IHA*.MAC. When used on a ZIP or GZIP statement the names designate datasets to be compressed. When used on an UNZIP statement a single name only is allowed and it will designate the input ZIP or GZIP dataset. If the input ZIP or GZIP dataset is a member of a PDS then if the optional selection argument is specified it is enclosed in parentheses following the member name e.g. **FROMDS(ASE.STANDARD.ZIP(MEMBER1(*.TXT))**. **FROMDS** may be abbreviated to **FRDS**. The datasets must already exist.

When used on a ZIP or GZIP statement the name(s) may be dataset selection masks that SLiKziP will use to interrogate the system catalog to identify the actual dataset(s) to be compressed. The selection masks may include the "?", "*" and "***" wildcards.

- **LF / NOLF**

Indicates that records are terminated with line feed. During a ZIP or GZIP operation, a line feed is added to the end of every record. During an UNZIP operation, a line feed is used to determine where each record ends. It is then discarded. Use **NOLF** to cancel this option when higher level control statements imply **LF**.

- **MASIS**

Specifies the "member" name specified in the **MNAME(...)** should be used "asis" i.e no substitution of built in functions will be performed. For example if **MNAME(Report_&DATE)** and **MASIS** are specified then the member name will be "Report_&DATE". If **MASIS** is not specified then &DATE would be replaced by the current date and the member name would be "Report_20070301"

- **MEMTS(...)**

Specifies how the program is to determine the member timestamp during a ZIP operation.

Allowed values are:

- **CHA** specifies that the CHANGED date for the input dataset is to be used;
- **REF** specifies that the REFERENCED date and time for the input dataset is to be used;
- **NOW** specifies that the current date and time is to be used. See Also **TZONE**

If omitted, MEMTS defaults to **CHA**.

- **MNAME(...)**

Specifies the "member" name to be used during a ZIP or GZIP operation. This is the name to be given to the corresponding member within the ZIP or GZIP dataset or archive. MNAME can be used to assign a "long file name" to a member. like **MNAME(Quarterly Sales Revenue.txt)**. When **MNAME(...)** is omitted and the source is a library or PDS, the default is the PDS member name plus any extension provided on the member selection text, if present. When the source is a sequential dataset, the default is the complete dataset name with all "." replaced by "/". This is the same as specifying the **MPATH** option.

- **MPATH/NOMPATH**

During a ZIP or GZIP operation **MPATH** indicates that a path should be included in the archive member name. The path is constructed from the source dataset name by replacing all "." with "/". For example dataset PROD.DAILY.REPORTS would become archive member **prod/daily/reports**. Use **NOMPATH** to cancel this option when higher level control statements imply **MPATH**.

- **MTRANS(tablename)**

Specifies a dynamic translate table to be used when creating or reading the names of members in a ZIP dataset. This table overrides SLiKZIP's internal translate tables used to translate EBCDIC names into ASCII when creating a ZIP dataset or when translating ASCII name into EBCDIC when reading ZIP datasets. This facility allows the creation of names with characters that are not part of the standard ASCII character set. A prior **TRANSTAB** control statement must have already defined the table. Refer to "Appendix C – Translate Tables in Control Statements" on page 61 for detail on creating your own translate tables. **MTRANS** cannot be specified on an **EXT** statement that is to be used in an UNZIP or AUDIT operation.

- **NODDESC**

Indicates that when adding members to a ZIP dataset, data descriptors must not be used.

A data descriptor is a 16-byte "trailer" record following the compressed data stream for a ZIP dataset member. Using data descriptors allows the ZIP program to avoid having to go back and update the ZIP

member header with the CRC and the compressed and uncompressed byte counts determined during the compression process. Some older unzip programs on other platforms do not support data descriptors. It's best to only specify this option when a specific need has been demonstrated.

This option is not supported when a GZIP dataset is being created. The GZIP file format requires data descriptors. GZIP data descriptors have a different format from those used in ZIP files.

- **OVERWRITE**

During a ZIP operation, if SLiKZIP determines that the output dataset already exists, it will expect it to be a valid ZIP dataset and it will attempt to add new members to, or replace existing members in, this dataset. Use this option to unconditionally replace the output ZIP dataset's contents (if any). Use this option with care as the output dataset will be completely overwritten.

This option is required when a GZIP operation will write to an existing, as opposed to a new, dataset. The OVERWRITE keyword should appear following a TODD(.) or TODS(.) process option on the GZIP statement –or– as a process option on a TODD or TODS statement. REPLACE is a synonym for OVERWRITE.

- **PAD(...) / NOPAD**

During an UNZIP operation, if the output dataset has a fixed record length and the uncompressed record is shorter than the LRECL then the record will be padded to the fixed record length with the pad character specified here. The pad character may be specified as a single character or as a pair of hex digits. The default pad character is a space. Use **NOPAD** to reset to the default when an higher level control statement has set a pad character.

- **PASSWORD(...)**

Specifies an encryption password. During a ZIP operation, the supplied password is used to encrypt the data added to the archive. During an UNZIP operation, the supplied password must match the password that was used to encrypt files in the archive.

When you are choosing passwords for your own data follow these simple rules: No dates, no names, no telephone or vehicle registration numbers, no character strings that might be found in dictionaries, encyclopaedias, common texts available on the internet, film dialogs, song lyrics, popular sayings et cetera. Do include at least one decimal digit and use a password of at least 12 characters.

This option is not supported when a GZIP dataset is being created or when one is being read. The GZIP file format does not support encryption.

During a ZIP operation, when the AES Process Option is specified, the Advanced Encryption Standard (AE-2) encryption is used. Encryption key length will be 128, 192 or 256 bits, depending upon the length of the password string specified:

Password length	Encryption key size in bits:
1-31	128
32-63	192
64-99	256

During an UNZIP operation, if the member has been encrypted according to the AE-2 or AE-1 standards, then SLiKZIP will automatically recognize this and use AES decryption.

To specify a password value in hex format, use the syntax (x'*hexvalue*') or password(x''*hexvalue*''). Hex digits are characters 0,1,2,3,4,5,6,7,8,9,0,a,b,c,d,e,f. The program supports upper and lower case for a,b,c,d,e,f. There must be an even number of hex digits in *hexvalue*. To improve the readability of a long string of hex digits, the program will allow *hexvalue* to be broken up into groups of hex digit pairs by embedded spaces, like PASSWORD(x'c1c2 f9d5').

Note 1: SLIKZIP allows a password of no more than 99 characters (198 hex digits) when the operation is ZIP and a password of no more than 128 characters (256 hex digits) when the operation is UNZIP or AUDIT. The 99 character limit is to enforce compatibility with Winzip which allows AES passwords no longer than 99 characters.

For more information about AE-2 see: http://www.winzip.com/aes_info.htm

- **PLATFORM(name)**

Sets the field in the ZIP file header that defines the type of system that created the ZIP file to the specified value. One of the following values can be specified: **DOS, AMIGA, VMS, UNIX, VM, ATARI, OS2, MAC, Z, CPM, WIN, MVS, VSE, ACORN, VFAT, AMVS, BEOS, TANDEM, OS400, OSX**. The default value set by SLiKZIP is **MVS**.

Some programs that read ZIP files use this field to determine how to display non standard ASCII characters in member names. When creating a ZIP file it may be necessary to use PLATFORM(..) to specify a different value to overcome the problem of incorrect characters being displayed by the program unzipping the ZIP file.

- **RDW**

During a ZIP or GZIP operation, specifies that Record Descriptor Words (RDW) are to be inserted. During an UNZIP operation, indicates that Record Descriptor Words are present in the data. This is a way of preserving the record structure of a RECFM=V/VB/U dataset that might otherwise be lost.

During a ZIP or GZIP operation, the data portion of each mainframe record will be prefixed with a standard RDW consisting of an unsigned 16 bit length followed by two bytes of zeroes. Note that the length of the RDW itself **is** included in the 16-bit length value.

During an UNZIP operation, if the RDW option is in effect, then the program determines how many bytes of data are in each uncompressed record by using the length from the RDW prefixing each record. Note that the RDW is not regarded as part of the data and is discarded.

RDW format example:

```
x'000F0000', 'DATA LINE 1'
```

```
  |   |
  |   | third and fourth bytes of the RDW are zero
```

```
  |
  | length of the data plus the RDW is 15 bytes (most significant byte first)
```

- **REPLACE**

The **REPLACE** option may be used on a **TODD** or **TODS** statement or following a **TODD(...)** or **TODS(...)** operand on a **ZIP, GZIP** or **UNZIP** statement. It can also be used on a **SELECT** statement during an UNZIP operation.

During an UNZIP operation, **REPLACE** allows the program to overwrite existing datasets or members. For a compress operation, **REPLACE** is equivalent to specifying **OVERWRITE**. It causes the program to overwrite any existing output dataset with a new ZIP or GZIP dataset.

- **SELECT(...)**

Use the **SELECT** option to specify library or archive members to be processed and, optionally, to provide a member name construction template. For a description of the selection argument see "*SELECT – Select Data for Processing*" on page 31.

- **SHOWHDR**

Headers from the ZIP dataset's central directory will be displayed as members of the ZIP dataset are read.

- **STRIP / NOSTRIP**

Indicates that During a ZIP or GZIP operation, trailing blanks are to be removed. **NOSTRIP** is the default. Use **NOSTRIP** to cancel this option when higher level control statements imply **STRIP**.

- **TEXT**

Indicates that files to be compressed or uncompressed are “text” files. This is equivalent to specifying **ASCII** and **CRLF** together ie. data is translated and records are terminated with CRLF. The translation process may be modified by specifying either the **TRANSTAB(...)** or **TRMOD(...)** options as well. During a ZIP operation, the TEXT keyword option will cause the TEXT member attribute to be set that tells compatible UNZIP programs to use the ASCII and CRLF options when uncompressing the member.

- **TODD(...)**

As a process option on a ZIP or GZIP statement, a single TODD(*ddname*) keyword may be used to designate the DD name for the pre-allocated output ZIP dataset.

As a process option on an UNZIP statement, one or more TODD(*ddname*) keywords may be specified, each preceded by SELECT(..) keywords and member processing options. The DD name *ddname* must be represented in your JCL by a corresponding DD statement.

- **TODS(...)**

As a process option on a ZIP or GZIP statement, a single TODS(*dsname*) keyword may be used to name the output ZIP dataset.

As a process option on an UNZIP statement, one or more TODS(*dsname*) keywords may be specified, each preceded by SELECT(..) keywords and member processing options. The dataset name *dsname* may be an actual dataset name or a dataset name pattern composed of literal characters and built-in function expressions to be resolved at run time.

If the output dataset already exists it will be allocated and its existing attributes will take effect. The REPLACE or OVERWRITE keywords may need to be specified. UNZIP operations may by default EXTEND (equivalent to DISP=MOD) an existing dataset if REPLACE is not specified.

If the output dataset does not exist then the program will attempt to allocate it. See the list of Allocation Options that may be specified following TODS(..) for allocating new datasets.

- **TRANSTAB(tablename)**

Use the named dynamic translate table to override SLiKZip's internal EBCDIC to ASCII translate table. A prior **TRANSTAB** control statement must have already defined the table. Refer to “Appendix C – Translate Tables in Control Statements” on page 61 for detail on creating your own translate tables. **TRANSTAB** is only effective when translation is already implied by **ASCII** or **TEXT** options.

- **TRMOD(modname)**

Use the named load module translate table to override SLiKZip's internal EBCDIC to ASCII translate table. Refer for details on creating your own translate tables. **TRMOD** is only effective when translation is already implied by **ASCII** or **TEXT** options.

- **TRUNC/NOTRUNC**

During uncompression, if the uncompressed data stream is longer than the record length of the output dataset, the excess data will be discarded if TRUNC is in effect. Use **NOTRUNC** or **WRAP** to cancel this option.

- **TZONE**

Specifies the time zone to be used if the member timestamp option **MEMTS(NOW)** is specified.

Allowed values are:

- **GMT** specifies that the timestamp is to be GMT time;
- **LOCAL** specifies that the timestamp is to be local time;

If omitted it defaults to **LOCAL**.

- **UPDATE**

Updates members already in the archive that match the selected data and also adds new members that are not already in the archive. Note that **ADD** is assumed when none of **ADD**, **FRESHEN** or **UPDATE** is specified. This option is not supported when a GZIP dataset is being created.

- **USELOCALHEADERVALUES**

During an UNZIP operation, this keyword tells the program to use the values present in the Local Header (located at the start of each member) and to ignore values in any Data Descriptor (an optional structure located at the end of the member) that may be present.

In a properly constructed ZIP file, if a Data Descriptor is encountered at the end of a member then the CRC field and the uncompressed & compressed byte count fields in the Local Header will be zero. However, if the Local Header values are not zero then SLiKZIP will report an error unless:

1. the values in the Data Descriptor exactly match those in the Local Header –or–;
2. **USELOCALHEADERVALUES** was specified.

If you encounter a situation where this option is necessary to allow UNZIP to work then you should contact the person supplying the ZIP file and notify them that the file is not strictly correct. It may be that they can use a different program to build the ZIP file or get the erroneous program corrected.

- **WRAP/NOWRAP**

During uncompression, if the uncompressed data stream is longer than the record length of the output dataset, excess data is inserted as the one or more following records in the output dataset. **WRAP** is the default when neither **WRAP** nor **TRUNC** is specified. Use **NOWRAP** or **TRUNC** to cancel this option.

- **ZDW**

During a ZIP or GZIP operation, **ZDW** specifies that ZIP Descriptor Words are to be inserted. During an UNZIP operation **ZDW** indicates that ZIP Descriptor Words are present in the data.

During a ZIP or GZIP operation, the data portion of each mainframe record will be prefixed with a 32-bit unsigned little-endian binary field containing the length of the data. Note that the length of the 32-bit prefix itself is **not** included in the 32-bit length value.

During an UNZIP operation, if the **ZDW** option is in effect, then the program determines how many bytes of data are in each uncompressed record by using the length from the **ZDW** prefixing each record. Note that the **ZDW** is not regarded as part of the data and is discarded.

ZDW format example:

```
x'0B000000', 'DATA LINE 1'
|
length of the following data is 11 bytes (least significant byte first)
```

Note that each ZIP member has a **ZDW** flag bit that will be set on by ZIP programs when the **ZDW** option was in effect when the member was created. UNZIP programs can recognise the presence of this member-level attribute flag without the user having to specify the **ZDW** keyword on UNZIP control statements.

- **ZIP64**

Allows SLiKZIP to compress data files longer than 4GB and to create a ZIP file larger than 4GB. This option should only be used when file sizes are likely to approach or exceed 4GB. Supported from SLiKZIP 1.4.

Note that some older UNZIP programs may not support this enhanced ZIP file architecture and therefore may not be able to unzip your data.

Built-in Functions

- **Creating output dataset names for UNZIP operations**

For UNZIP operations, output dataset names can be built at run time using built-in functions. You may want to vary target dataset names depending on the contents of an archive. This feature can greatly enhance automation of UNZIP operations.

When the dataset name provided as the first operand of a TODS statement, or as the value of a TODS(..) operand, contains “&” or * characters then we refer to it as a pattern dataset name.

The actual dataset whose name is generated at run time may or may not exist. If it does exist then you must specify the REPLACE keyword to allow SLiKZIP to overwrite it, or the MOD keyword to allow SLiKZIP to extend it. If it doesn't exist you may want SLiKZIP to create it. To assist this process you may specify many allocation attributes on control statements. See “Allocation Options” on page 47 for details.

When building a dataset name, any lower case characters selected are converted to upper case. If the resulting dataset name is not a valid one then the dynamic allocation will fail and SLiKZIP will be unable to complete the operation.

- **Creating archive member names for ZIP operations**

Member names created during a ZIP or GZIP operation can be specified via the MNAME(..) process option. The MNAME string can include symbolic references to all or parts of, for example, input dataset names, PDS member names or the current date and time. If you need to create a member name containing a “&” character then you must specify “&&”. The “&&” characters will be replaced by a single “&” in the member name.

- **Using Built-in Functions for generating names**

SLiKZIP supports a number of symbolic references that can be used to create pattern dataset names for UNZIP operations and archive member names during ZIP operations.

- **Using Substrings and Segment numbers**

All of the functions permit optional substring specifications (up to three numbers enclosed in braces { } immediately following the function name). Substring specification supports two forms: ‘from:to’ and ‘from,for’. Use of the colon or comma differentiates between the two forms.

‘from:to’ format indicates the start and end positions of a range. Negative numbers indicate positions relative to the right-hand end e.g. 2:-2 means from position 2 to the second last position.

The ‘from,for’ format indicates the start position and a count of characters. The ‘from’ position can be a negative number here as well e.g. ‘-2,1’ means from the second last position for a count of one.

There is a third form where a single number provides a “from” position and the “for” value is assumed to be “the rest”.

Some of the built-in functions support a segment number as well. Dataset names, like SYS1.PROD.SYSLOG, and paths, like /etc/editor/settings.conf, consist of multiple parts that we refer to as segments.

The segment number may be used to select a single segment from the name. Again, a negative number may be used with, for example, '-1' meaning the last or right-most segment. Segment number 0 refers to the complete name including the segment separator characters.

When using a function reference without any substring specification, the reference must be followed immediately by a trailing "." so that SLiKZiP knows unambiguously where the function name ends. Assuming we wanted ZIP member names to consist of the PDS member name plus an appended "#0" then if we tried MNAME(&MEM#0) SLiKZiP would complain about *unknown variable or function in pattern*. Instead we'd have to code MNAME(&MEM.#0). The corollary of this is that when we actually want a period to follow the resolution of a function name we have to use two periods, the first of which will be discarded. e.g. MNAME(&time..txt)

- Built-in Functions supported by SLiKZiP

The following built-in functions are supported by SLiKZiP:

&zf[[from:to | from,for]] refers to the current file-id.

&zn[[from:to | from,for]] refers to the current file-name.

&mem is a synonym for **&zn** in UNZIP operations and for the PDS member name in ZIP operations.

&zx[[from:to | from,for]] refers to the current file extension.

&ext is a synonym for **&zx** in UNZIP operations.

&frdd[[from:to | from,for]] refers to the current source DD name.

&frds[[segment#[,segmentcount | :segment#]]] refers to the current source dataset name. Note that dataset name segments are separated by periods (.). This syntax allows one or more dataset name segments to be extracted.

&frds[[segment#[,from:to | from,for]]] refers to the current source dataset name. Note that dataset name segments are separated by periods (.). This syntax allows all or part of a single dataset name segment to be extracted.

&frpa[[segment#[,segmentcount | :segment#]]] refers to the current source dataset name with periods replaced by "/" so that the dataset name looks like a path with segments separated by the "/" character. This syntax allows one or more dataset name segments to be extracted with "/" separating them.

&frpa[[segment#[,from:to | from,for]]] refers to the current source dataset name with periods replaced by "/" so that the dataset name looks like a path with segments separated by the "/" character. This syntax allows all or part of a single dataset name segment to be extracted.

&date[[from:to | from,for]] refers to the local date in yyymmdd format.

&time[[from:to | from,for]] refers to the local time in hhmmss format.

&gmdate[[from:to | from,for]] refers to the GMT date in yyymmdd format.

&gmtime[[from:to | from,for]] refers to the GMT time in hhmmss format.

&ZP[[segment#[,from:to | from,for]]] refers to the current path. Path segments are separated by the "/" character. The last segment, &ZP{-1}, is equivalent to &ZF.

Here are some examples illustrating various substring arguments. These examples assume that **&frds** has the value "APPL.SALES.D020516" and **&mem** has the value "DAT01JUN".

&frds {2}	resolves to	"SALES"
&frpa {2:3}		"SALES/D020516"
&frds {1,2}		"APPL.SALES"

&frds{3,2:7}	"020516"
&frds{3,4,4}	"0516"
&frds{3}	"D020516"
&frds{-1}	"D020516"
&frpa{0,4,3}	"L/S"
&MEM{4:8}	"01JUN"
&MEM{-3}	"JUN"

- **Error message text specific to Built-in Function processing**

The following error text can appear as a part of, or in association with, other SLiKZiP messages:

invalid template or pattern syntax

Review the syntax rules, review the template in MNAME(..) or TODS(..) operands.

result buffer overflow

The template expansion exceeds 256 bytes, the maximum length supported

unknown variable or function in pattern

Review the supported function names. Perhaps a function name should be terminated with “.”

empty sublist after variable reference in pattern

The program found { } characters indicating a sublist is present but it was empty.

a sublist term is null

too many sublist terms

The specific built-in function does not support as many terms as the sublist contains.

segment numbers out of order

Starting segment # must not resolve higher than ending segment #

sublist contains a negative "for" value

For values must be 0,1,2,...

sublist "to" value resolves lower than "from" value

If negative “from” or “to” values were specified then there may be fewer segments or positions in the value of the built-in function or variable name than you expected.

Allocation Options Reference

Some control statements can specify output datasets. The options described here may be used to control the allocation of those datasets. SLiKZiP supports essentially the same allocation options as the TSO ALLOC command. Not all of the possible allocation options are described here. You may find additional useful information about some allocation options by using the TSO HELP command for ALLOC.

These options may be abbreviated as long as they do not conflict with other options that are valid on the current control statement. For example on a **SELECT** control statement, **TRANSTAB(...)** may be abbreviated to **TRANS(...)** or **TRA(...)**, but not **TR(...)** due to ambiguity with **TRMOD(...)**. However, on a **TODS** control statement **TRA(...)** cannot be used due to ambiguity with the **TRACKS** allocation option.

A few of these options may be effective and useful even when the output dataset has already been allocated but is going to be overwritten. These structural options include **RECFM(..)**, **LRECL(..)** and **BLKSIZE(..)**.

- **BLKSIZE(...)**

The block size for a new dataset.

- **BLOCK(nnnn)**

Space is to be allocated by blocks, where **nnnn** is the block size. The number of blocks to be allocated is specified by the **SPACE(...)** option.

- **CATALOG**

The dataset is to be cataloged when freed.

- **CYLINDERS**

Space is to be allocated by cylinders. The number of cylinders to be allocated is specified by the **SPACE(...)** option.

- **DATACLAS(...)**

Data class for SMS managed datasets.

- **DELETE**

The dataset is to be deleted when freed.

- **DIR(...)**

The number of directory blocks.

- **DUMMY**

SLiKZiP will write output to a dummy dataset.

- **DSNTYPE(LIBRARY|PDS|BASIC|LARGE)**

Specifies the type of dataset to be created.

- **EXPDT(yyyy/ddd)**

Specify an expiry date for the dataset.

- **KEEP**

The dataset is to be kept when freed.

- **LABEL(...)**

The label type for tape datasets.

- **LIKE(...)**

Specify a dataset that attributes will be copied from for use in this allocation. This includes **RECFM**, **LRECL** and **BLKSIZE**.

- **LRECL(...)**

The logical record length.

- **MGMTCLAS(...)**

Management class for SMS managed datasets.

- **MOD**

The dataset may or may not exist. If it doesn't exist it will be created. In any case, extracted member's are to be appended to the end of the dataset.

- **NEW**

The dataset doesn't exist and is to be created.

- **OLD**

The dataset exists and exclusive control is required.

- **POSITION(...)**

The file position for a tape dataset.

- **RECFM(...)**

The record format.

- **RELEASE**

Free unused space when the dataset is closed.

- **RETPD(nnnn)**

Specify a retention period date for the dataset.

- **SHR**

The dataset exists and will be shared.

- **SPACE(pri[,sec])**

The numbers of primary and secondary space units. The secondary is optional.

- **STORCLAS(...)**

Storage class for SMS managed datasets.

- **SUBSYS(name,param1,param2,...)**

Specify a subsystem to handle this allocation and any parameters to be passed to it.

- **TRACKS**

Space is to be allocated by tracks. The number of tracks is specified by the **SPACE(...)** option.

- **UNCATALOG**

The dataset is to be uncataloged when freed.

- **UNIT(...)**

The device type where the dataset is, or is to be, allocated.

- **VOLUME(...)**

The volume on which the dataset resides or is to reside.


```
//ZIP EXEC PGM=SLIKZIP,PARM=ZIP
```

And it's often possible to do without control statements altogether and instead just use process options in the PARM= field to provide all the necessary information:

```
//ZIP EXEC PGM=SLIKZIP,PARM='ZIP TODD(OUT) FROMDD(IN) TEXT'
```

As this example suggests, you can effectively place a single ZIP, GZIP, UNZIP or AUDIT statement in the PARM= field, if it will all fit in 100 characters.

Using anywhere close to that 100 characters (the limit is a restriction of batch JCL, not of SLiKziP) will still often mean that you'll need to split the PARM= field across two or more lines of JCL. There's an FAQ on how to do that a little further down in this section.

Why do I see option ASCII even though I didn't specify ASCII or TEXT?

Q Why does SLIKZIP tell me that the ASCII option is in effect, when I didn't specify ASCII or TEXT!

A The process option **ASCII** does tell SLIKZIP that you want translation of the data before compression (ZIP) or after uncompression (UNZIP). The default translation is between EBCDIC and ASCII using SLIKZIP's internal default translation tables. However, it's possible to override these default tables by using either of the **TRANSTAB** or **TRMOD** process options. Both these keywords also imply translation and SLIKZIP will consequently show ASCII in the ZIP0178I message:

ZIP0178I options are: ASCII,CRLF

How can I automatically transfer files using FTP?

Q I want to use FTP to do file transfers. It becomes tedious having to enter ftp commands each time I want transfer a ZIP dataset. Is there a quicker and easier way to do it?

A Yes, the following steps describe one technique you can use. There are probably many other similar techniques as well. It assumes that you have access to an ftp server on your OS/390 or z/OS system and are running Windows 2000 or later on your desktop. Feel free to use different file and directory names to those shown below.

- Create a directory on your *c:* drive called *xfer*.
- Create a file called *zipsend.bat* with the following contents and save it in your *c:\xfer* directory. Substitute *hostname* with the host name or IP address of your OS/390 or z/OS system.

```
c:
cd \xfer
c:\windows\system32\ftp.exe -s:zipsend.ct1 hostname
```

- Create a file called *zipsend.ct1* with the following contents and save it in your *c:\xfer* directory. Substitute *userid* with your userid and *password* with your password.

```
userid
password
binary
hash
bell
bin
put xfer.zip
bye
```

- Go to your Windows desktop and click the right mouse button. From the menu select New > Shortcut. On the next dialog box enter *ftp* into the *Command line* text box and click the *Next* button. On the next dialog box enter an appropriate title eg. *send zip file to Host* and click the *Finish* button. Right click the new icon that has appeared on your desktop and click *Properties* on the popup menu. Click the *Shortcut* tab on the next dialog box. In the *Target* text box, enter *c:\xfer\zipsend.bat*. In the *Start in* text box, enter *c:\xfer*. Finally click the *OK* button.

Create or update *c:\xfer\xfer.zip* with your favorite ZIP program and double click your new shortcut to send the file to OS/390 or z/OS. When running SLiKZiP, use the name *xfer.zip* as your compressed dataset.

Follow a similar procedure to create your receive shortcut with the following changes.

- Create *ziprecv.bat* with the following contents. Substitute *hostname* with the host name or IP address of your OS/390 or z/OS system.

```
c:
cd \xfer
c:\windows\system32\ftp.exe -s:ziprecv.ct1 hostname
```

- Create *ziprecv.ct1* with the following contents. Substitute *userid* with your userid and *password* with your password.

```
userid
password
binary
hash
bell
bin
get xfer.zip
bye
```

- Create another shortcut but call it *receive ZIP file from Host* instead. In the *Target* text box, enter *c:\xfer\ziprecv.bat*. All other details of the shortcut are the same.

After you have created or updated *xfer.zip* with SLiKZiP, switch to Windows and double click your new receive shortcut.

How do I split a PARM string into multiple lines and not get JCL errors?

Q The PARM string I need is too long for one line. When I try to split it over two lines, I either get a JCL error or SLiKZiP doesn't get the parameter string I expect. How do I split the PARM string into multiple lines and not get JCL errors?

A Here's a valid example split over three JCL lines:

```
//STEP1 EXEC PGM=ZIP,
// PARM=('FRDS(PROD.DAILY.ACCTS.REPTS(*.TXT))', <-source dataset
// 'TODS(PROD.DAILY.TRANS.REPTS.ZIP) ', <-target dataset
// 'TEXT LF') <-text + LF for UN*X
//
See SAMPLIB member PARMCONT
```

The 77 characters in this PARM= string will be presented to the program looking like this:

```
"FRDS ( PROD . DAILY . ACCTS . REPTS ( * . TXT ) ) , TODS ( PROD . DAILY . TRANS . REPTS . ZIP ) , TEXT LF"
```

Points to note:

- The part of the whole PARM= string that appears on any individual line is enclosed in single apostrophes. These apostrophes do not appear within the final string the program sees;
- Each continued line must end with a comma. These commas do appear within the final string the program sees. While SLiKZiP expects terms in the PARM= string to be separated by one or more spaces (blanks) it will treat any comma found in the PARM= string, where space is expected, as though it actually is a space.
- The maximum allowable length of the PARM= string is limited by JCL syntax rules to 100 characters. This will include any comma(s) that appear at the end of a continued line.
- The entire PARM= value must be enclosed in parentheses (it's a general rule for JCL operand values continued on a subsequent line). Thus PARM=(... on the first line is matched by ..) on the last line.
- Breaking the PARM= string up allows some commenting so it may be useful to break up a PARM= string even when it will all fit on a single line.
- While the issues arising from continuation of the operand of the PARM= keyword parameter on the EXEC statement are not specific to SLiKZiP (the rules are made by IBM, not by ASE), getting it right or wrong will affect SLiKZiP users, hence these notes!

Is SLiKZiP different from ISPZIP?

Yes. SLiKZiP is a new product designed from the ground up over the last three years. It is a functional replacement for ISPZIP and for other mainframe ZIP programs.

Q What are the main differences between SLiKZiP and ISPZIP?

A SLiKZiP uses different default DD names, has a much more extensive parameter and control statement syntax, can perform multiple ZIP, UNZIP and AUDIT operations in a single step and has better reporting of processes performed.

Q Does SLiKZiP have an ISPF dialog component like ISPZIP does?

A Yes. SLiKISPF was delivered with Version 1.10 in June 2010.

Appendix A – SLIKISPF - the ISPF dialog application

The SLIKISPF dialog application allows users to:

- **View** the content of existing ZIP datasets;
- **Extract** (UNZIP) members of existing ZIP datasets;
- **create** new ZIP datasets;
- **update** (ADD / UPDATE / FRESHEN) members of existing ZIP datasets.
- **list** datasets, the members of libraries and the members of ZIP datasets

SLIKISPF Tutorial

- See <http://www.slikzip.com/sztutorial.htm> to see a list of typical tasks for ZIP datasets. Click each link to see a corresponding set of screen shots illustrating the use of SLIKISPF.
- ASE highly recommend that all users spend the several minutes needed to view these tutorials.

Miscellaneous SLIKISPF operational notes:

- See the install guide: <http://www.slikzip.com/szinstall.htm> for ISPF-specific install questions.
- The Program Function (PF) key assignments appear at the bottom of the display screen. Use the PFSHOW primary command to toggle these on and off. Use the KEYS primary command to view and edit PF key assignments.
- Use the HELP key throughout the dialog application to get help.
- The SLIKZIP ISPF dialog uses application identifier (APPLID) "ZSE" (assigned by IBM to ASE).
- Use primary command "msg zipnnnn" to get a description of any ZIPnnnn message.
- If you are trialling the program, and it requests a valid DEMOKEY value, you can obtain a short-term valid key by going to the SLIKZIP 1.10 Download Page...

<http://www.slikzip.com/szdemo.htm>

...and clicking:

"I have read, and accept, the conditions listed above".

Appendix B – Translate Tables in Load Modules

When the ASCII option is in effect SLiKZiP translates the data. During a ZIP operation, the data is translated from EBCDIC to ASCII before compression. During an UNZIP operation the data is translated from ASCII to EBCDIC after uncompression. Note that the TEXT process option implies the ASCII process option.

When SLiKZiP's built-in translate tables don't meet your specific requirements you can tell the program to use alternative tables by using the TRMOD(...) process option. The LOAD(...) keyword on TRANSTAB statements also allows you to load a translate table packaged in load module form and to optionally modify the table before using it.

TRMOD(...) and TRANSTAB LOAD(...) both name a program, or load module, that contains a pair of 256-byte tables. The first table in the load module is used during compression to convert each EBCDIC byte to an equivalent ASCII byte. The second table is used during uncompression to convert each ASCII byte to an equivalent EBCDIC byte.

Translate Table Format

This assembler source implements the default translate table pair used by SLiKZiP.

```

*
EBCTOASC CSECT  USED DURING COMPRESS OPERATIONS
*
          0 1 2 3 4 5 6 7 8 9 A B C D E F
DC      XL1'00,01,02,03,CF,09,D3,7F,D4,D5,C3,0B,0C,0D,0E,0F' 0.
DC      XL1'10,11,12,13,C7,B4,08,C9,18,19,CC,CD,83,1D,D2,1F' 1.
DC      XL1'81,82,1C,84,86,0A,17,1B,89,91,92,95,A2,05,06,07' 2.
DC      XL1'E0,EE,16,E5,D0,1E,EA,04,8A,F6,C6,C2,14,15,C1,1A' 3.
DC      XL1'20,A6,E1,80,EB,90,9F,E2,AB,8B,9B,2E,3C,28,2B,7C' 4.
DC      XL1'26,A9,AA,9C,DB,A5,99,E3,A8,9E,21,24,2A,29,3B,5E' 5.
DC      XL1'2D,2F,DF,DC,9A,DD,DE,98,9D,AC,B3,2C,25,5F,3E,3F' 6.
DC      XL1'D7,88,94,B0,B1,B2,FC,D6,FB,60,3A,23,40,27,3D,22' 7.
DC      XL1'F8,61,62,63,64,65,66,67,68,69,96,A4,F3,AF,AE,C5' 8.
DC      XL1'8C,6A,6B,6C,6D,6E,6F,70,71,72,97,87,CE,93,F1,FE' 9.
DC      XL1'C8,7E,73,74,75,76,77,78,79,7A,EF,C0,DA,5B,F2,F9' A.
DC      XL1'B5,B6,FD,B7,B8,B9,E6,BB,BC,BD,8D,D9,BF,5D,D8,C4' B.
DC      XL1'7B,41,42,43,44,45,46,47,48,49,CB,CA,BE,E8,EC,ED' C.
DC      XL1'7D,4A,4B,4C,4D,4E,4F,50,51,52,A1,AD,F5,F4,A3,8F' D.
DC      XL1'5C,E7,53,54,55,56,57,58,59,5A,A0,85,8E,E9,E4,D1' E.
DC      XL1'30,31,32,33,34,35,36,37,38,39,B3,F7,F0,FA,A7,FF' F.
*
ASCTOEBE CSECT  USED DURING UNCOMPRESS OPERATIONS
*
          0 1 2 3 4 5 6 7 8 9 A B C D E F
DC      XL1'00,01,02,03,37,2D,2E,2F,16,05,25,0B,0C,0D,0E,0F' 0
DC      XL1'10,11,12,13,3C,3D,32,26,18,19,3F,27,22,1D,35,1F' 1
DC      XL1'40,5A,7F,7B,5B,6C,50,7D,4D,5D,5C,4E,6B,60,4B,61' 2
DC      XL1'F0,F1,F2,F3,F4,F5,F6,F7,F8,F9,7A,5E,4C,7E,6E,6F' 3
DC      XL1'7C,C1,C2,C3,C4,C5,C6,C7,C8,C9,D1,D2,D3,D4,D5,D6' 4
DC      XL1'D7,D8,D9,E2,E3,E4,E5,E6,E7,E8,E9,AD,E0,BD,5F,6D' 5
DC      XL1'79,81,82,83,84,85,86,87,88,89,91,92,93,94,95,96' 6
DC      XL1'97,98,99,A2,A3,A4,A5,A6,A7,A8,A9,C0,4F,D0,A1,07' 7
DC      XL1'43,20,21,1C,23,EB,24,9B,71,28,38,49,90,BA,EC,DF' 8
DC      XL1'45,29,2A,9D,72,2B,8A,9A,67,56,64,4A,53,68,59,46' 9
DC      XL1'EA,DA,2C,DE,8B,55,41,FE,58,51,52,48,69,DB,8E,8D' A
DC      XL1'73,74,75,6A,15,B0,B1,B3,B4,B5,6A,B7,B8,B9,CC,BC' B
DC      XL1'AB,3E,3B,0A,BF,8F,3A,14,A0,17,CB,CA,1A,1B,9C,04' C
DC      XL1'34,EF,1E,06,08,09,77,70,BE,BB,AC,54,63,65,66,62' D
DC      XL1'30,42,47,57,EE,33,B6,E1,CD,ED,36,44,CE,CF,31,AA' E
DC      XL1'FC,9E,AE,8C,DD,DC,39,FB,80,AF,FD,78,76,B2,9F,FF' F
*
          END
See SAMPLIB member TRMODSRC

```

How translation works for each character or byte

SLiKZIP translates uncompressed bytes. Each uncompressed byte is used as an offset (programmers sometimes say “index”) into the appropriate table and the byte at that offset in the table is then used to replace the original byte in the uncompressed data.

Let's assume we are compressing character data. The EBCDIC character set assigns the bit pattern b'11000001', or x'C1', to the upper case letter “A”. So if we go to offset x'C1' in the first table shown above, the byte there contains x'41'. The ASCII character set assigns the bit pattern b'01000001' or x'41', to the upper case letter “A” so what we found looks symmetrical.

If we were uncompressing data then the ASCII “A” will be x'41' and if we go to that offset in the second table we find x'C1', the hex equivalent of the bit pattern that the EBCDIC character set assigns to “A”.

Creating Your Own Translate Table

No single pair of translation tables will meet everybody's requirements. Individual countries may have national currency symbols mapped to bit patterns that have no specific assignments in EBCDIC or ASCII. Requirements may be based on the need to interface to particular machinery or to correct for mistakes made elsewhere in the data processing continuum.

Let's look at an example of creating your own translate table pair in load module form. The Euro € is usually assigned to x'9F' in EBCDIC and x'80' in ASCII (be careful to check your local requirements before assuming that these assignments are appropriate for you).

Follow this procedure to produce your own translate table in load module form:

*Note: See **Appendix B** for how to do all this without using Assembler and creating load modules. However, you'll probably find it helpful to read the rest of Appendix A anyway at some point, even if you do decide to use the method described in Appendix B.*

Start by creating a new assembler source file by taking a copy of the source shown on the previous page.

1. We have to make complementary changes to each of the two tables:

- Look through the EBCTOASC part of the table and locate the byte at the offset that EBCDIC uses for the character, in this case the offset is x'9F'. Make a note of the value already there (x'FE') and replace it with the ASCII value for the character, in this case x'80'.
- Look through the ASCTOEBE part of the table and locate the byte at the offset that ASCII uses for the character, in this case x'80'. Make a note of the value already there (x'43') and Note the value there (x'43') and replace it with the EBCDIC value for the character, in this case x'9F'.
- Before we modified it, the EBCTOASC table translated x'9F' to x'FE' and, maintaining the two-way nature of these translation tables, the ASCTOEBE table translated x'FE' to x'9F'. Similarly, the original ASCTOEBE table translated x'80' to x'43' and the original EBCTOASC table translated x'43' to x'80'.
- So, tying up loose ends, and assuming that x'43' in EBCDIC and x'FE' in ASCII don't have any character assignments, we recommend that you change the EBCTOASC table to translate x'43' to x'FE' and the ASCTOEBE table to translate x'FE' to x'43'. What this does is to simply arrange that if you used SLiKZIP to first compress data containing x'43' in any bytes and to then uncompress that ZIP dataset, then the original x'43' bytes will be

reconstituted correctly. If the ZIP dataset was shipped to another platform and uncompressed without the corresponding translation table then results will be unpredictable.

- Your translate table should now look like this (the changes are marked in bold):

```

*
* SLIKZIP translate table modified to include euro
*
EBCTOASC CSECT  USED DURING COMPRESS OPERATIONS
*
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
DC      XL1'00,01,02,03,CF,09,D3,7F,D4,D5,C3,0B,0C,0D,0E,0F' 0.
DC      XL1'10,11,12,13,C7,B4,08,C9,18,19,CC,CD,83,1D,D2,1F' 1.
DC      XL1'81,82,1C,84,86,0A,17,1B,89,91,92,95,A2,05,06,07' 2.
DC      XL1'E0,EE,16,E5,D0,1E,EA,04,8A,F6,C6,C2,14,15,C1,1A' 3.
DC      XL1'20,A6,E1,FE,EB,90,9F,E2,AB,8B,9B,2E,3C,28,2B,7C' 4.
DC      XL1'26,A9,AA,9C,DB,A5,99,E3,A8,9E,21,24,2A,29,3B,5E' 5.
DC      XL1'2D,2F,DF,DC,9A,DD,DE,98,9D,AC,B3,2C,25,5F,3E,3F' 6.
DC      XL1'D7,88,94,B0,B1,B2,FC,D6,FB,60,3A,23,40,27,3D,22' 7.
DC      XL1'F8,61,62,63,64,65,66,67,68,69,96,A4,F3,AF,AE,C5' 8.
DC      XL1'8C,6A,6B,6C,6D,6E,6F,70,71,72,97,87,CE,93,F1,80' 9.
DC      XL1'C8,7E,73,74,75,76,77,78,79,7A,EF,C0,DA,5B,F2,F9' A.
DC      XL1'B5,B6,FD,B7,B8,B9,E6,BB,BC,BD,8D,D9,BF,5D,D8,C4' B.
DC      XL1'7B,41,42,43,44,45,46,47,48,49,CB,CA,BE,E8,EC,ED' C.
DC      XL1'7D,4A,4B,4C,4D,4E,4F,50,51,52,A1,AD,F5,F4,A3,8F' D.
DC      XL1'5C,E7,53,54,55,56,57,58,59,5A,A0,85,8E,E9,E4,D1' E.
DC      XL1'30,31,32,33,34,35,36,37,38,39,B3,F7,F0,FA,A7,FF' F.

*
ASCTOEBC CSECT  USED DURING UNCOMPRESS OPERATIONS
*
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
DC      XL1'00,01,02,03,37,2D,2E,2F,16,05,25,0B,0C,0D,0E,0F' 0
DC      XL1'10,11,12,13,3C,3D,32,26,18,19,3F,27,22,1D,35,1F' 1
DC      XL1'40,5A,7F,7B,5B,6C,50,7D,4D,5D,5C,4E,6B,60,4B,61' 2
DC      XL1'F0,F1,F2,F3,F4,F5,F6,F7,F8,F9,7A,5E,4C,7E,6E,6F' 3
DC      XL1'7C,C1,C2,C3,C4,C5,C6,C7,C8,C9,D1,D2,D3,D4,D5,D6' 4
DC      XL1'D7,D8,D9,E2,E3,E4,E5,E6,E7,E8,E9,AD,E0,BD,5F,6D' 5
DC      XL1'79,81,82,83,84,85,86,87,88,89,91,92,93,94,95,96' 6
DC      XL1'97,98,99,A2,A3,A4,A5,A6,A7,A8,A9,C0,4F,D0,A1,07' 7
DC      XL1'9F,20,21,1C,23,EB,24,9B,71,28,38,49,90,BA,EC,DF' 8
DC      XL1'45,29,2A,9D,72,2B,8A,9A,67,56,64,4A,53,68,59,46' 9
DC      XL1'EA,DA,2C,DE,8B,55,41,FE,58,51,52,48,69,DB,8E,8D' A
DC      XL1'73,74,75,6A,15,B0,B1,B3,B4,B5,6A,B7,B8,B9,CC,BC' B
DC      XL1'AB,3E,3B,0A,BF,8F,3A,14,A0,17,CB,CA,1A,1B,9C,04' C
DC      XL1'34,EF,1E,06,08,09,77,70,BE,BB,AC,54,63,65,66,62' D
DC      XL1'30,42,47,57,EE,33,B6,E1,CD,ED,36,44,CE,CF,31,AA' E
DC      XL1'FC,9E,AE,8C,DD,DC,39,FB,80,AF,FD,78,76,B2,43,FF' F

*
END

```

- Assemble and link your new translate table. Here is some sample JCL to do that. Substitute “**translate-table-source**” and “**your-load-library**” with the dataset names for your translate table source and your target load library respectively. Substitute “**trtbname**” with the name of your new translate table.

```

//ASM      EXEC PGM=ASMA90,
//          PARM='DECK,LIST,OBJECT,TERM'
//SYSPRINT DD SYSOUT=*
//SYSLIB  DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD DUMMY
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2  DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3  DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSTEM  DD SYSOUT=*
//SYSLIN  DD DSN=&&SYSLIN,UNIT=SYSDA,DISP=(,PASS),
//          SPACE=(TRK,(1,1)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//SYSIN   DD DSN=translate-table-source,DISP=SHR
//*
//LKED    EXEC PGM=IEWL,COND=(4,LT),
//          PARM='XREF,NCAL'
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD DD DSN=your-load-library,DISP=SHR
//SYSLIN  DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//          DD *
//          NAME trtbname(R)
//
See SAMPLIB member TRMODASM

```

- Use your new translate table by coding the **TRMOD(trtbname)** keyword on the appropriate EXT statement. Ensure that the load library containing the module is available to SLiKZiP via //STEPLIB DD or //JOBLIB DD or via the linklist.

Appendix C – Translate Tables in Control Statements

Translate tables can be created dynamically using TRANSTAB control statements.

- Variations on the default tables can be created “in flight”;
- PKZIP-compliant translate tables can be loaded and modified “in flight”;
- Complete translate tables can be built “in flight”;
- Any translate table can be printed.

TRANSTAB control statement syntax

TRANSTAB tablename FOR(ZIP|UNZIP) [COPY(tablename)] [LOAD(programname)] [VERIFY]

tablename This is a 1 to 40 character name chosen by you to identify the table in the following control statements. To refer to this table on another control statement you would use this table name in a TRANSTAB(...) option. The table you create here is dynamic and only exists for the life of the current SLiKziP run.

FOR(...) Indicates whether this table is to be used for ZIP (compression) or UNZIP (uncompression) operations. This operand is required.

COPY(name) This table is initialized by taking a copy of another table defined by an earlier TRANSTAB statement. If neither COPY(...) nor LOAD(...) is specified, the table is initialized as an empty table.

LOAD(name) This table is initialized by loading the named program. For ZIP (compress) operations, the first 256 bytes of the program is copied into the new translate table. For UNZIP (uncompress) operations the second 256 bytes is copied. If neither COPY(...) nor LOAD(...) is specified, the table is initialized as an empty table.

VERIFY The translate table is printed after any changes have been applied. The display is 16 lines. Each line shows 16 bytes as hex digit pairs, like this:

```
ISPZ151I  VERIFY requested - table dump follows:
+..  _0 _1 _2 _3 _4 _5 _6 _7 _8 _9 _A _B _C _D _E _F
+00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+20  6D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+30  F9 F8 F7 F6 F5 F4 F3 F2 F1 F0 00 00 00 00 00 00
+40  00 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96
+50  97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 00 00 00 00 00
+60  00 C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6
+70  D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 00 00 00 00 00
+80  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+90  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
+F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Table Modification Records

Changing or defining translate tables “in flight” is achieved using Table Modification Records. You can place any number of these immediately following the TRANSTAB control statement. Each such record **must** be blank in column one and have the following general format.

```
+offset data [more of each] /* comments
```

The offset looks like `+xx` where `xx` is any valid pair of hex digits. This provides access to every byte of a 256-byte table. The *data* consists of pairs of hex digits or character strings enclosed in apostrophes. This data is converted to one or more bytes and placed in the translate table beginning at the specified offset.

In the following example, SLiKziP's default translate tables are modified to handle the Euro symbol. The Euro € is usually assigned to `x'9F'` in EBCDIC and `x'80'` in ASCII. Take care to check your local requirements before assuming that these assignments are appropriate for you.

Before making any changes you can run SLiKziP with the following control statements. You can then see the complete translate tables before determining what changes you need to make.

```
/* list the standard tables

/* EBCDIC->ASCII
TRANSTAB TEST1 COPY(ASCII) FOR(ZIP) VERIFY

/* ASCII->EBCDIC
TRANSTAB TEST2 COPY(ASCII) FOR(UNZIP) VERIFY

See SAMPLIB member TRANSTA1
```

The following control statements will create two dynamic translate tables. `EURO_ZIP` to be used when translating from EBCDIC to ASCII in ZIP operations, and `EURO_UNZIP` when translating from ASCII to EBCDIC in unzip operations. Note that another byte, as well as the euro, is also translated. This is done to ensure symmetrical translation. See the discussion in “Creating Your Own Translate Table” on page 58 for the rationale behind this.

```
/* modify the standard tables to handle the euro

/* EBCDIC->ASCII with euro
TRANSTAB EURO_ZIP COPY(ASCII) VERIFY FOR(ZIP)
/* gotta leave col 1 blank for this to work
+43 FE /* translate x'43' to x'fe'
+9F 80 /* translate x'9f' to x'80'

/* ASCII->EBCDIC with euro
TRANSTAB EURO_UNZIP COPY(ASCII) VERIFY FOR(UNZIP)
/* gotta leave col 1 blank for this to work
+80 9F /* translate x'80' to x'9f'
+FE 43 /* translate x'fe' to x'43'

See SAMPLIB member TRANSTA2
```

Appendix D – SLiKZIP Messages

This section lists SLiKZIP messages in message-id sequence.

All SLiKZIP messages are prefixed ZIPnnnnc. Values of c may be E/e (error), W/w (warning) and I/i (information). Note that ASE has been assigned the MVS component ids ZIP and UNZ by IBM.

- ZIP0007I

ZIP0007I levels:

This message summarises the code levels in effect for major components of slikzip. Several entries "Pnnnn yymmdd-hhmm" are printed in the message. By referring to the "Product Support" section at <http://www.slikzip.com> you can see if the copy of slikzip that you are running is at the latest maintenance level. When reporting any slikzip problem to ASE you must include the image of this message in the material emailed to help@slikzip.com.

- ZIP0008E

ZIP0008E unexpected end of file encountered on compressed dataset

The program encountered apparent eof while reading a compressed dataset when further data was expected or required. Processing terminates. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer.

- ZIP0014E

ZIP0014E Unsupported ZIP program version number in member header

The program does not recognize the "version needed to extract" contained in the initial member header in the input ZIP dataset. The program supports version 1 (000Ah), version 2 (0014h and 0015h) and version 4.5 (002dh) only. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the ZIP dataset was created by a supported version of PKZIP or a compatible program then rerun the operation with the SHOWHDR option, preserve the output and contact ASE for support.

- ZIP0015E

ZIP0015E Unsupported compression method encountered:

The program has encountered a member in the input dataset with a compression method not supported by this program. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. The program will display contents of the file header for diagnosis. Preserve the output and contact ASE for support.

- ZIP0016E

ZIP0016E GZIP header FLG field (offset +3) bits 5 to 7 not all zero

The flag byte at offset +3 in the GZIP header is invalid. Bits 5 to 7 are not all zero.

- ZIP0018W

ZIP0018W Output LRECL less than input data length - truncating data

The record length of the output dataset is less than the apparent record length of the uncompressed data. The uncompressed data will be truncated. To preserve truncated data, remove the TRUNC keyword or override it with the WRAP keyword.

- ZIP0019W

ZIP0019W Output LRECL less than input data length - wrapping data

The record length of the output dataset is shorter than the record length of the uncompressed data. The uncompressed record will be wrapped by creating additional output records as required to hold the extra data.

- ZIP0024E

ZIP0024E Bypassing encrypted member, PASSWORD not supplied.

The file cannot be uncompressed because it is encrypted. Use the PASSWORD(..) keyword option to provide the password for this member.

- ZIP0025E

ZIP0025E data descriptor / local file header values inconsistent

While examining the local file header in preparation for uncompress or audit processing, the program found that either the crc and length fields in the the header are zero and the data descriptor flag is not set, or, the data descriptor flag is set and the CRC, Uncompressed and compressed size fields are not all zero. The program will display contents of the file header for diagnosis.

- ZIP0026E

ZIP0026E I/O error on dataset:

The program has encountered an i/o error condition while attempting to read from or write to the specified dataset.

- ZIP0027E

ZIP0027E diagnostics:

Additional diagnostic information provided by the system.

jobname,stepname,unit,dt,ddname,operation,errdesc,position,BSAM
message shows diagnostic information provided by the system:

```
* unit      device number / unit number
* dt:       device type, D for disk, T for tape
* errdesc:  a short error description
* position: for disk: bbbbccchhhrrr, for tape: relativeblk#
The program signals an unrecoverable i/o error for this file.
```

- ZIP0028E

ZIP0028E invalid o/p dataset name:

The program has encountered an invalid dataset name. The name may have been built from a dataset name pattern. If this is the case then there may be previous messages that are relevant to the error. Correct the pattern and rerun the job.

- ZIP0029E

ZIP0029E Too many ALIAS entries:

An attempt was made to uncompress a load module with more than 50 alias entries. Subsequent alias entries are ignored

- ZIP0030E

ZIP0030E Bad Alias entry

A bad alias entry was found while attempting to uncompress a load module.

- ZIP0031E

ZIP0031E Unsupported method:

The program cannot decompress the member because the compression method identified in the member header is not one the program supports.

- ZIP0032E

ZIP0032E Data Descriptor expected, not found

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. Preserve the output and contact ASE for support.

- ZIP0033E

ZIP0033E Skipping member:

The program has encountered an error that prevents it from completing the extraction of this member. The member is skipped.

- ZIP0034E

ZIP0034E Unknown code block type

The program has encountered an unrecognised code block type. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- ZIP0035E

ZIP0035E Invalid length in stored block

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another

platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- **ZIP0036E**

ZIP0036E No match found for HUFFMAN code

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- **ZIP0037E**

ZIP0037E Two values have the same HUFFMAN code

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- **ZIP0038E**

ZIP0038E Code size exceeds maximum

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- **ZIP0039E**

ZIP0039E Invalid bit length value

The program has encountered an apparent structural error within the compressed data. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the problem persists rerun the program with SHOWHDR specified, preserve the output and contact ASE for support.

- **ZIP0040E**

ZIP0040E missing DSPE/DSNA/SDSE for i/p dataset

Rerun the job with a //DIAGNOSE DD statement present in the JCL. If this error re-occurs then save the output and contact HELP@ASE.COM.AU

- **ZIP0041W**

ZIP0041W process bypassed - required FROM./TO.. specification omitted

The program cannot perform the process because a required source or target data specification has been omitted. The process ends with condition code 8.

- ZIP0042E

ZIP0042E existing pattern-based dsname without MOD or REPLACE:

The program has detected that an output dataset whose name was generated from a pattern already exists. Neither MOD nor REPLACE were specified. The program will not write to the dataset unless either MOD or REPLACE are specified.

- ZIP0043E

ZIP0043E Invalid RDW or BDW found – negative length

The program has encountered an RDW or BDW field in the uncompressed data which has a negative value. Either the compressed file is corrupt or the file does not have RDW or BDW fields before each record. Try rerunning the unzip without RDW or BDW specified.

- ZIP0044E

ZIP0044E bad syntax in dataset name:

The dataset name shown does not comply with dataset name syntax rules. Any dataset name segment that includes an ampersand may contain more than 8 characters. The program will validate dataset names built by substitution of symbols at a later stage.

- ZIP0057I

ZIP0057I term associated with error:

An allocation request has failed. If this message appears it will display the ALLOCATE operand associated with the error. Review the control statements for allocation information that may contain syntax or other errors.

- ZIP0058E

ZIP0058E allocate failed for xxx dataset:

The program attempted to dynamically allocate the dataset for the usage shown by xxx (I/P or O/P) but the allocation failed. Please refer to any immediately prior ZIP0057W message.

- ZIP0059E

ZIP0059E catalogued xxx dataset not on volume vvvvvv:

The program attempted to read the label of the catalogued dataset. The label (DSCB) was not found on the volume indicated in the catalog entry.

- ZIP0070E

ZIP0070E Not a ZIP dataset. Possible structural error.

The program is reading what it expects to be a valid ZIP dataset. This situation may occur during compression or uncompression. The input or existing ZIP dataset does not begin with a valid signature string -or- there is insufficient data to hold a valid signature string. The dataset may have been damaged. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the dataset

is not a valid existing ZIP dataset then you may code the OVERWRITE keyword option to force a compression operation to overwrite an existing dataset without inspection.

- ZIP0071E

ZIP0071E member is encrypted, the password supplied is incorrect

The password specified is not correct. The member cannot be uncompressed.

- ZIP0072I

ZIP0072I Cent Dir File Header #nnnnn #bytes c:cccccccccc u:uuuuuuuuuu

The SHOWHDR option is in effect. The program is reading the central directory of the ZIP dataset. This message shows the member number and the number of compressed and uncompressed bytes recorded in the Central Directory entry.

- ZIP0073E

ZIP0073E Central Directory contains non-CDFH/ECDR data. Structural error .

While reading the Central Directory of a ZIP dataset, the program either encountered data that was neither a Central Directory File Header (CDFH) or End of Central Directory Record (ECDR), -or- the program encountered end of file without finding the ECDR. The dataset may have been damaged. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the dataset is not a valid existing ZIP dataset then you may code the OVERWRITE keyword option to force a compression operation to overwrite an existing dataset without inspection.

- ZIP0074E

ZIP0074E member is encrypted, password must be specified

A password must be specified on the select statement to decrypt the file.

- ZIP0075E

ZIP0075E Unable to find to Central Directory. Possible structural error .

The program cannot find a Central Directory in the ZIP dataset. The dataset may have been damaged. If the dataset was transferred from another platform then verify that it was not translated from ASCII to EBCDIC during the transfer. If the dataset is not a valid existing ZIP dataset then you may code the OVERWRITE keyword option to force a compression operation to overwrite an existing dataset without inspection.

- ZIP0076I

ZIP0076I Central Directory File Headers will be shown

The SHOWHDR option is in effect. The program will display Central Directory File Headers as they are read to determine what members are present in an existing ZIP dataset.

- ZIP0077I

ZIP0077I Cent Dir totals: #mem:nnnnnn #bytes c:ssssssssss u:ssssssssss

The program has finished reading the Central Directory of the ZIP dataset and is displaying its accumulated control totals for number of members and total bytes compressed and uncompressed because the SHOWHDR option is in effect or because a discrepancy exists with the control totals in the ECDR. An ECDR dump follows.

- ZIP0078I

ZIP0078I Local File Headers for copied members:

The SHOWHDR option is in effect. The program will display Local File Headers for existing members copied to the new ZIP dataset.

- ZIP0079I

ZIP0079I unable to rename new dataset to:

The attempt to rename the new dataset to take the place of the old has failed. Please check catalog information.

- ZIP0080E

ZIP0080E unable to delete dataset:

The attempt to delete a ZIP dataset has failed. Please check catalog information.

- ZIP0081E

ZIP0081E ADD, UPDATE and FRESHEN are mutually exclusive options

Mutually exclusive options have been chosen Choose the correct option and remove the conflicting keyword(s).

- ZIP0082E

ZIP0082E TEXT and BINARY are mutually exclusive options

Mutually exclusive options have been chosen Choose the correct option and remove the conflicting keyword(s). TEXT implies ASCII and CRLF. BINARY implies NOASCII and NOCRLF.

- ZIP0083I

ZIP0083I day hh:mm:ss end pppppppp returned rc nnnn, cpu:sssss.s sec

the program is displaying the highest return code encountered while performing the operation. The message also contains information about the day of week, time of day and cpu times used in performing the process.

- ZIP0084E

ZIP0084E open failed for print file dd xxxxxxxx,

The program cannot open the print file. Please check the allocation of the ddname shown in the message. The PRTDD= keyword parameter may be used in the PARM= string to specify an alternative ddname.

- ZIP0085E

ZIP0085E program invocation error, unsupported environment or interface.

The program cannot identify the operating system, operating mode (batch or command line), locate the program name or locate the parameter string or command image passed to the program. Check that the program is being invoked by one of its supported entry point names: SLiKZIP, ZIP, UNZIP or one of the optional aliases that may be defined at install time, ISPZIP or ASAP. Check that the program is being invoked in the either the batch EXEC PGM= or TSO command interface style, or from another program via the ATTACH or LINK macros.

- ZIP0086E

ZIP0086E parameter string or command image exceeds 4091 bytes

The program supports a maximum of 4091 bytes of parameter string or command operands.

- ZIP0087I

ZIP0087I PARM=

The text of this message is the PARM= or command operand string passed to the program. If the line has a plus sign (+) in position 80 then the string is continued starting in position 2 of the next line.

- ZIP0088I

ZIP0088I //DEBUG DD statement detected, diagnostic displays enabled

The program will perform additional diagnostic displays such as displaying the source of processing options and dumping member headers and other control structures found in or built for ZIP or GZIP files.

- ZIP0089E

ZIP0089E errors were detected while parsing the PARM= or command string

The program detected an error while parsing the PARM= parameter or command operands string.

- ZIP0090E

ZIP0090E ZIP/GZIP/UNZIP/COMP/DECOMP/UNCOMP/AUDIT process not specified

The program cannot determine what process to perform. The process name has not been specified in the PARM= or command operand strings nor as a statement in the control file. Either execute the program as PGM=ZIP or PGM=UNZIP -or- insert the correct process name in the PARM= string, like PARM='PRTDD=MYPRT ZIP FROMDD(..) ..' or insert an appropriate statement in the control file.

- ZIP0091I

ZIP0091I day hh:mm:ss start process to/fr/dd/ds(name)

The program is about to start a new ZIP, UNZIP or AUDIT operation. The message shows the process number and the compressed dataset involved.

This message helps to segment the listing when multiple processes have been requested.

- **ZIP0092I**

ZIP0092I dd aaaaaaaa not realloc :

The target ZIP dataset was pre-allocated to ddname aa and identified to the program using TODD. The program may have had to unallocate the original ZIP dataset to perform a DELETE + RENAME sequence. In this situation, to make the ddname available to subsequent operations that may refer to it, the program attempts to re-allocate the correct ZIP dataset using the same ddname. This re-allocation has failed. Please check any prior messages for information that may help to explain the allocation failure.

- **ZIP0093I**

ZIP0093I dd aaaaaaaa reallocated :

The target ZIP dataset was pre-allocated to ddname aa and identified to the program using TODD. The program may have had to unallocate the original ZIP dataset to perform a DELETE + RENAME sequence. In this situation, to make the ddname available to subsequent operations that may refer to it, the program attempts to re-allocate the correct ZIP dataset using the same ddname. This re-allocation has been successful.

- **ZIP0094E**

ZIP0094E ZDW/RDW conflict with scan for end-of-record or end-of-file

Mutually exclusive options have been chosen. ZDW/RDW specifies that a binary length prefix precedes the data for each unzipped record. CRLF, CR, LF, EOR(..) or EOF(..) process options indicate that end-of-record byte(s) are used to delimit each record. Note that the TEXT process option also implies CRLF. Choose the correct option and remove the conflicting keyword(s).

- **ZIP0099E**

ZIP0099E

The program encountered the error described by the message while processing a member name template. Check the syntax of the explicit or implied MNAME template string. Descriptions include:~ invalid template or pattern syntax;~ result buffer overflow (template expands to > 256 bytes);~ unknown variable or function in pattern;~ empty sublist after variable reference in pattern;~ a sublist term is null;~ too many sublist terms;~ segment numbers out of order;~ sublist contains a negative "for" value;~ sublist "to" value resolves lower than "from" value.

- **ZIP0100E**

ZIP0100E invalid PDS member name:

The program encountered the error while attempting to build a PDS member name from an MNAME(..) pattern or template.

- ZIP0101E

ZIP0101E OPEN failed for PDS directory

The directory of the FROM dataset could not be opened. Check that the FROMDS\FROMDD parameter is specified correctly.

- ZIP0102E

ZIP0102E OPEN failed for member:

The specified member could not be opened. Check that the member exists in the dataset.

- ZIP0103I

ZIP0103I action. oo fff dataset :

The program has opened the dataset for reading or writing. oo is DSORG (PS,PO), fff is RECFM (F,FB,FBS,V,VB,VBS) This message audits the use of the dataset by the program.

- ZIP0104E

ZIP0104E ZDW must be specified to zip load modules.

The dataset being zipped contains load modules and the ZDW keyword was not specified. ZDW must be specified for load modules to enable the module to be correctly unzipped.

- ZIP0105E

ZIP0105E Zipping of program library objects not supported.

The dataset being zipped is a program library. The current version of Slikzip does not support zipping of program library objects.

- ZIP0106E

ZIP0106E Unzipping to a program library not supported

The output dataset for the current unzip operation is a program library. Unzipping to a program library is currently not supported by Slikzip.

- ZIP0107E

ZIP0107E UNZIP is not supported for GZIP in this build of SLIKZIP

An attempt was made to use a feature not supported in the current level or build of the program. Please contact ASE for information about availability of this feature.

- ZIP0108E

ZIP0108E OPEN failed for control statement file, DD=ddddddd

The program could not open the control statement file. Please verify that correct DD statement is present and that the CTLDD(..) parameter specifies the correct ddname if it is other than SYSIN.

- ZIP0112E

ZIP0112E OPEN failed for o/p dataset :

The output dataset could not be opened. Check that the dataset name was correctly spelled.

- ZIP0113E

ZIP0113E OPEN failed for control statement file, DD=ddddddd

The program could not open the control statement file. Please verify that correct DD statement is present.

- ZIP0114E

ZIP0114E Existing member not replaced:

The directory of the TODS/TODD library is full.

- ZIP0115E

ZIP0115E Directory update failed, no space in directory for member:

The directory of the TODS/TODD library is full.

- ZIP0116E

ZIP0116E Directory update failed, STOW retc=nn reas=nn for member:

An update of a PDS directory has failed. The STOW macro return code and reason code appear in the message. Check the target dataset for i/o errors.

- ZIP0117I

ZIP0117I alias:

The program has created the ALIAS entries shown in the list

- ZIP0118I

ZIP0118I Input file data in block at TTRZ:

The program is able to determine that the data being displayed is located at the offset shown within a block located at the TTR shown.

- ZIP0119E

ZIP0119E CRC check failed, calculated=xxxxxxx, control=xxxxxxx

The CRC calculated by the program during uncompression is different from the CRC passed within the ZIP file header or data descriptor.

- ZIP0120E

ZIP0120E The specified table is not designed for the current process

The translate table specified by the TRANS() operand is for another process. If the current process is ZIP or COMP then the TRANSTAB statement specified a FOR value of UNZIP, UNCOMP or AUDIT.

- ZIP0121W

ZIP0121W Partial table specified without prior COPY or LOAD, VERIFY set.

Some bytes of the translate table area were not specified by TRANSTAB control statements and the table had not been initialised using the COPY or LOAD options. The VERIFY option will be set on to cause the table area to be displayed in the control listing. The program is taking these actions to alert the user in case an incomplete translate table has been used. The user can suppress this message by specifying all 256 positions of the table area -or- by initialising the table area by means of the COPY or LOAD options.

- ZIP0122E

ZIP0122E COPY and LOAD are mutually exclusive options

Mutually exclusive options have been chosen. Choose the correct option and remove the conflicting keyword(s).

- ZIP0123E

ZIP0123E OPEN failed for input dataset:

The dataset could not be opened for input. Verify that the dataset exists, that FROMDD/FROMDS has been correctly specified and that the user has access to the dataset.

- ZIP0124E

ZIP0124E COPY specifies a source table with conflicting FOR(..) value

The program requires that the source table to be used to initialise the table being defined have a compatible FOR(..) value. Specifically, if the new table is FOR(ZIP) or FOR(COMP) then the source table must not be FOR UNZIP, DECOMP or AUDIT. The reverse is also true.

- ZIP0125E

ZIP0125E Specify table usage as FOR=ZIP/UNZIP/COMP/DECOMP/AUDIT

The TRANSTAB statement must specify whether the table is intended for use during compression or uncompression processing.

- ZIP0126I

ZIP0126I OPEN failed for update of TO dataset

The output dataset could not be opened for update. Check that the dataset is correctly allocated and that you are authorized to update the dataset.

- ZIP0127E

ZIP0127E OPEN failed for input dataset, unopened:

The dataset could not be opened for input because the dsllstar value was zero indicating that the dataset had never been written to. This action is taken to prevent unpredictable data being read. This action also prevents inadvertent access to residual data from prior usage of the allocated disk extent.

- ZIP0128E

ZIP0128E ZIP64 Extended Information Extra Field missing in CDFH

The ZIP64 Extended Information Extra Field was missing in the CDFH. There may be a structural problem with the ZIP file.

- ZIP0129E

ZIP0129E ZIP64 extended information extra field not found

The ZIP64 extended information extra field was not found. The ZIP dataset may be corrupted.

- ZIP0130E

ZIP0130E Missing/Bad ZIP64 extended information field.

A Local File Header (LOCH), or a Central Directory File Header (CDFH), was missing a ZIP64 extended information field. The dataset may have been damaged.

- ZIP0131E

ZIP0131E too much selection data

The program could not process all the selection arguments provided for a single uncompress operation. Use wildcards to shorten some parts of the selection arguments or split the arguments across multiple uncompress operations. The total amount of selection text is limited to approximately 6000 characters.

- ZIP0132E

ZIP0132E bad syntax in a SELECT argument:

The program found invalid syntax in a selection argument string. The leading part of the string in error is displayed in the message following the colon (:). Processing terminates.

- ZIP0133E

ZIP0133E RDJFCB failed for input

The JFCB could not be read for an input dataset. The dataset may be one selected for input to a compress operation or it may be an existing ZIP dataset that the program is attempting to determine the contents of. Verify that the dataset name has been correctly specified.

- ZIP0134E

ZIP0134E SWAREQ macro non zero return code rc=xx:

An attempt to obtain a JFCB extension block for a multivolume dataset failed. Contact ASE for support.

- ZIP0135E

ZIP0135E bad syntax in a generic selection argument:

The generic selection argument contains a misplaced '*' character. The '*' character must be the last character.

- ZIP0136E

ZIP0136E FROM dataset is already compressed.

The input dataset is already compressed. Check that you have specified the correct input dataset.

- ZIP0137E

ZIP0137E MTRANS is not valid on EXT statement for UNZIP or AUDIT

A member name translate table cannot be specified on an EXT statement that is used for an UNZIP or AUDIT operation. MTRANS can only be specified on a FROMDD/FROMDS, TODD/TODS or SELECT statement for UNZIP or AUDIT operations.

- ZIP0138E

ZIP0138E no selection arguments specified

The program could not select any members to be uncompressed because no selection arguments were specified. Please specify a selection (SEL) statement prior to the TODS/TODD statement.

- ZIP0139E

ZIP0139E Unable to find translate table specified by TRANSTAB option

The translate table specified by the TRANSTAB(..) operand was not defined. Use the TRANSTAB control statement to define the table.

- ZIP0140E

ZIP0140E Unable to find translate table specified by TRMOD option:

The translate table specified by the TRMOD(..) operand was not found because LOAD failed for the named load module.

- ZIP0141E

ZIP0141E more than one DSPE/DSNA/SDSE for i/p dataset

Rerun the job with a //DIAGNOSE DD statement present in the JCL. If this error re-occurs then save the output and contact help@ase.com.au

- ZIP0142E

ZIP0142E input dataset name cannot be a pattern name

The input dataset name must be specified in full. There is no support for the use of symbolic or pattern characters within the input ZIP or GZIP dataset name.

- ZIP0143E

ZIP0143E open failed for input dataset

The program could not open the specified dataset. Please check that the dataset name has been correctly spelled, that the dataset exists and that you have the necessary access to the dataset name.

- ZIP0145I

ZIP0145I Pattern:

The current pattern dsname string being used to create an input or output dataset name is shown after the colon (:)

- ZIP0146I

ZIP0146I

This message shows an internal variable and its substitution value.

- ZIP0147I

ZIP0147I Digital Signature Record:

The program dumps the Digital Signature Record.

- ZIP0148E

ZIP0148E pattern dataset name build failed:

Output dataset name creation has failed. The error is identified by the rest of this message. The preceding messages show the pattern and the current variable values.

- ZIP0149I

ZIP0149I Control statement listing...

The program lists the control statements following this message along with any messages about syntax or other errors.

- ZIP0150I

ZIP0150I Zip64 End Central Directory Locator Record:

The program dumps the ZIP64 End of Central Directory Locator Record

- ZIP0151I

ZIP0151I VERIFY requested - table dump follows:'

The VERIFY keyword option was coded on the TRANSTAB statement. The program dumps the translate table after any changes due to overlay control cards following the TRANSTAB statement.

- **ZIP0152E**

ZIP0152E <error description>:

While parsing the control statement shown immediately above in the control listing the program has encountered a term that is not recognised or supported at the point where it was found. It is also possible for this message to be generated for miscellaneous errors. Check the spelling and syntax rules for the control statement or parameter string shown and resubmit the job.

Possible error descriptions for this message include:

Unknown term: <term>

The keyword term is unknown or not supported where used.

error, ddname or dsname list is empty

No ddname or dsname was specified where one was expected.

only one TODD/TODS/TODSN operand is allowed

The operation supports only a single output file.

logic error, last block built not DSPE/EXTD

Contact ASE for support.

SELECT misplaced, no preceeding FROMDD/FROMDS

SELECT statement(s) or the SELECT(..) operand cannot precede the FROMDD/FROMDS statement or operand that defines the input.

ZIPINDEX misplaced, no preceeding TODD/TODSN

A ZIPINDEX statement may only follow the specification of the output ZIP dataset by either a TODD/TODS/TODSN process option on the ZIP statement -or- a separate TODD/TODS/TODSN statement.

<keyword> misplaced, ZIP, UNZIP or AUDIT not yet specified

The program must first know what operation is to be performed change the name of the program being executed to ZIP or UNZIP or specify ZIP, UNZIP or AUDIT in the PARM= field or begin the control statement file with a ZIP, UNZIP or AUDIT statement.

<keyword> misplaced, source DD or DSN already known for <oper>

FROMDD/FROMDS can only be specified once for UNZIP or AUDIT.

-or-

TODD/TODS can only be specified once for ZIP.

not allowed for operation AUDIT

TODD/TODS is inappropriate for the AUDIT operation

<keyword> misplaced, source DD or DSN not yet known for <oper>

UNZIP and AUDIT operations require that the input dataset be defined before any TODD/TODS statements or operands are encountered.

PASSWORD must be from 1 to 128 characters only

Check the password syntax. If embedded spaces, commas or parentheses are used then enclose the value within apostrophes ('). If embedded apostrophes are used then enclose the value within doublequotes (") or vice versa. You may not use both apostrophes and doublequotes within the same password value string. See also: Specifying password value in hex format.

PASSWORD in hex format has syntax errors

The program has detected that the password value is specified as a hex digit string (it commences x'... or x"...).and one of the following errors has been detected:

- The value is not terminated with a matching apostrophe (') or doublequote (");
- At least one character within the string is not a valid hex digit (ie. not 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f);
- The value contains more than 256 hex digits;
- there is an odd number of hex digits present.

DDNAME not valid for FRDD/TODD

The specified DDNAME is not a valid DDNAME for the FRDD or TODD keyword. Select another ddname and re-run the job.

seq field 73-80 not all decimal digits

Cols 73-80 of the first record of the statement file contained all decimal digits. This causes the program to assume that the statement area ends in col 72 and to require that every record of the statement file contains decimal digits in col 73-80. A record in the current statement breaks this rule.

SELECT misplaced, no following TODS/TODD statement

Select statements for **UNZIP** operations must be followed by a **TODS/TODD** statement that specifies the target dataset for the operation.

- **ZIP0153E**

ZIP0153E translate table definition already exists:

The TRANSTAB control statement contains the name of a table already defined by a prior TRANSTAB statement. A default translate table called "ASCII" is internally defined and cannot be redefined. See TRANSTAB .. COPY(..) and LOAD(..) for options. Remove the duplicate definition or change the name on the TRANSTAB statement.

- **ZIP0154E**

ZIP0154E translate table not previously defined:

The table named in the COPY(..) operand has not been previously defined nor is it the internally defined default table "ASCII".

- ZIP0155E

ZIP0155E translate table module not found:

The load module named in the LOAD(..) operand cannot be located. Check that the library containing the module is in the default search path for load modules (eg: steplib, joblib, linklist, lpalib).

- ZIP0156E

ZIP0156E too many selection arguments

A single SELECT statement or SELECT(..) operand is limited to a maximum of about 1,000 characters of selection data. Use wildcards to shorten some parts of the selection arguments or split the arguments between multiple SELECT statements.

- ZIP0157E

ZIP0157E TODD/FROMDD ddname not allocated:

The ddname specified in the statement is not currently allocated. Correct the spelling of the ddname in the statement or ensure that the JCL contains the required DD statement.

- ZIP0158E

ZIP0158E TODD used to specify target ZIP dataset without OVERWRITE :

TODD was used to define the target ZIP dataset. If the target ZIP dataset is a new dataset then the program cannot process it. Either use TODS to designate the target ZIP dataset and remove the DD statement allocating it from the JCL -or- code the OVERWRITE keyword option immediately after the TODD(..) operand or after the ddname on the TODD statement to cause the program to create a new ZIP dataset in the space occupied by the output dataset -or- change the output DD statement to use a different dataset name with an initial disposition of NEW.

- ZIP0159E

ZIP0159E error parsing path/fileid:

The member path/fileid string shown caused errors when parsing into segments separated by "/" or while parsing the final segment into parts separated by "." to develop the filename and extension.

- ZIP0160I

ZIP0160I Zip64 End Central Directory Record:

The program dumps the ZIP64 End of Central Directory Record

- ZIP0161E

ZIP0161E Unknown data structure encountered in a ZIP or GZIP dataset

The program has encountered data within a ZIP or GZIP dataset that is not a recognised control structure at a point where one should be. Up the 256 bytes of data from that point in the dataset will be dumped in hex and ASCII characters.

- ZIP0162I

ZIP0162I ZIP file member header dump follows:

The program dumps the contents of the Local File Header (LOCH) for the current member of the ZIP dataset.

- ZIP0163I

ZIP0163I Data Descriptor dump follows:

The following lines are a dump of the data descriptor.

- ZIP0164I

ZIP0164I Central directory file header dump follows:

The following lines are a dump of the central directory.

- ZIP0165W

ZIP0165W Input file length exceeds 4GB, recommend use of ZIP64 or GZIP

The input file is longer than the maximum length (4GB) supported by the "basic" ZIP file format. SLiKZIP will be able to uncompress this file but other programs may not. If the file is to be uncompressed on another platform then your choices are to create a GZIP file or to create a ZIP file using the ZIP64 option. Always check that the target platform supports whichever file format and option you choose.

- ZIP0166I

ZIP0166I End Central Directory Record (ECDR):

The program dumps the ECDR that should be the final data structure in the ZIP dataset.

- ZIP0167W

ZIP0167W Output file length exceeds 4GB - GZIP file format recommended

The output file is longer than the maximum length (4GB) supported by the ZIP file format. SLiKZIP will decompress this file but other programs may not. It is recommended that the GZIP file format is used to process this file.

- ZIP0168I

ZIP0168I

The message text may help diagnose a problem with the print file.

- ZIP0169E

ZIP0169E ZIP/GZIP option conflict

The program has been told to create both a ZIP file and a GZIP file. These options are mutually exclusive. Examine the control listing for an indication of the source of these options.

- ZIP0170E

ZIP0170E pad not single character or hex digit pair

The PAD(..) keyword operand supports a single character or a hex digit pair (00,01,...,FE,FF). If two characters were specified then they must be valid hex digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- ZIP0171E

ZIP0171E WRAP and TRUNC are mutually exclusive

Mutually exclusive options have been chosen Choose the correct option and remove the conflicting keyword(s). Specify WRAP (this is the default) when the uncompressed datastream does not have end-of-record indicators, like Carriage Return + Line Feed or just Line Feed. The program has to be told to look for end of record indicators. This is done with CRLF, LF or EOR(..) keywords. Options like CRLF may be implied by other options. For example the ASCII option implies CRLF. The TEXT option implies ASCII and CRLF and the CRLF option implies LF during decompression operations. TRUNC causes excess data beyond that which will fill one record in the target dataset to be discarded. Unusual choice. The default WRAP still results in an internal return code of 4 from the decompression operation. However, if the WRAP keyword was specified then this tells the program that the user expects wrapping and the internal return code 4 will be suppressed.

- ZIP0172E

ZIP0172E ZDW and RDW are mutually exclusive

Mutually exclusive options have been chosen Choose the correct option and remove the conflicting keyword(s). ZDW specifies that a 4-byte little-endian binary length field prefixes each unzipped record. This option is often used when compressing non-text data where the embedded end-of-record signals like CRLF cannot be reliably used. RDW is an alternative that specifies a standard Record Descriptor Word is to prefix each unzipped record.

- ZIP0173I

ZIP0173I CR/LF or EOR(..) specified without translation

The program is informing the user that an unusual combination of options has been chosen.

- ZIP0174I

ZIP0174I translation requested without CR/LF or EOR(..) specified

The program is informing the user that an unusual combination of options has been chosen.

- ZIP0175E

ZIP0175E extension already defined: xxxxxxxx

An EXT statement specifies an extension name that has been defined by a prior EXT statement. Remove the duplicate definition or change the name on the EXT statement.

- ZIP0176E

ZIP0176E translate table already defined: tablename

The TRANSTAB statement contains a table name that is already in use. Remove the duplicate definition or change the name on the TRANSTAB statement.

- ZIP0177E

ZIP0177E invalid record or file terminator value:

Record and File terminator values can be specified by combining one, two or three two-character keywords. These can be any of CR, LF, CZ, EM or NU -or- the character + followed by any letter A-Z -or- any pair of hex digit characters. (Valid hex digits are: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F). Examples: CRLF, 0D0A, +J+M, LFCR, 0A0D, +M+J+Z

- ZIP0178I

ZIP0178I options are:

The processing options currently in effect are displayed in the format: keyword=value

- ZIP0179E

ZIP0179E input dataset has unsupported DSORG

The program does not support the dataset organisation of the input dataset. Supported DSORG values are PS and PO.

- ZIP0180E

ZIP0180E missing DSPE/DSNA/SDSE for o/p dataset

Rerun the job with a //DIAGNOSE DD statement present in the JCL. If this error re-occurs then save the output and contact HELP@ASE.COM.AU

- ZIP0181E

ZIP0181E more than one DSPE/DSNA/SDSE for o/p dataset

Rerun the job with a //DIAGNOSE DD statement present in the JCL. If this error re-occurs then save the output and contact HELP@ASE.COM.AU

- ZIP0182E

ZIP0182E output dataset name cannot be a pattern name

The output dataset name must be specified in full. There is no support for the use of symbolic or pattern characters within the output ZIP or GZIP dataset name.

- ZIP0183E

ZIP0183E open failed for output dataset

The program could not open the specified dataset. Please check that the dataset name has been correctly spelled, that the dataset exists if UPDATE or FRESHEN have been specified, that space exists on the target volume(s) if the dataset is NEW and that you have the necessary access to the dataset name.

- ZIP0184E

ZIP0184E bad syntax in ext(..) operand or ext name

The program detected an invalid pattern or name value while attempting to match an EXT value to the names of EXT definitions provided via EXT control statements. Please check the rules for wildcard (*,?) usage within the name field of EXT statements

- ZIP0185W

ZIP0185W PASSWORD specified, NODDESC ignored

PASSWORD or encryption support in ZIP files requires that Data Descriptors be created within the ZIP member. The program will ignore the specified NODDESC option.

- ZIP0186W

ZIP0186W GZIP specified, NODDESC ignored

GZIP files require Data Descriptors. The program will ignore the specified NODDESC option.

- ZIP0187E

ZIP0187E RDJFCB failed for output ddname:

The JFCB could not be read for the output dataset. Verify that the dataset name is correctly specified.

- ZIP0188I

ZIP0188I action--- type dataset :

This message audits the type (ZIP/GZIP) and name of the output dataset. If action is "create" or "overwrite" then a new ZIP dataset will be created. If action is "modify" then an existing ZIP dataset will be processed. If action is "overwrite" then if the output dataset will be overwritten completely regardless of whether it appeared to be an existing ZIP dataset.

- ZIP0189I

ZIP0189I miscellaneous information about the input dataset

If the program made assumptions about record maximum record length or block size then this message reports the choices made.

- ZIP0190I

ZIP0190I text

The program found it necessary to choose or adjust record format (RECFM), logical record length (LRECL) or block size (BLKSIZE) when opening the output dataset.

- ZIP0191E

ZIP0191E Central Directory control totals discrepancy exists

The program found that the control totals it accumulated while reading Central Directory File Headers (CDFH) differ from the total fields in the End of Central Directory Record (ECDR). The dataset

may have been damaged. Rerun the program with the SHOWHDR option coded to get a dump of the individual CDFH.

- **ZIP0192I**

ZIP0192I aaaaaaaaa member not bbbbbbbbb:

If an existing ZIP dataset is being processed: A new member cannot be ADDED if a member of the same name already exists in the ZIP dataset. A new member cannot be FRESHENed if a member of the same name did not already exist in the ZIP dataset. If a new ZIP dataset is being created: A member cannot be ADDED if a member of the same name has already been added to this ZIP dataset. Possible actions to consider before rerunning: Change the member naming via the mname(..) parameter; Change the operation from ADD or FRESHEN to UPDATE. (Note that if unspecified, action defaults to ADD.) For ADD and UPDATE operations the current process will be terminated. Use the NOAFUERR keyword to ignore this error and continue processing. For FRESHEN the current member will be skipped and processing continued.

- **ZIP0193E**

ZIP0193E OBTAIN failed for o/p dataset:

The program was unable to read the label for the output dataset. Verify that the output dataset is actually on the volume specified in the dataset catalog or via UNIT= VOL= keywords in the JCL.

- **ZIP0194E**

ZIP0194E previous errors prevent delete/rename sequence for o/p ZIP file

The program is updating an existing ZIP dataset. Refer to message ZIP0188I for the current name of the output ZIP dataset. The program will not proceed to delete the old ZIP dataset and rename the transient dataset to the old dataset name due to an unexpected condition. See previous messages for an indication of the problem.

- **ZIP0195I**

ZIP0195I deleted dataset :

The program has deleted either an old version of a now-updated ZIP dataset, or an empty new ZIP dataset.

- **ZIP0196I**

ZIP0196I renamed new dataset to :

The program has renamed the new ZIP dataset to take the place of the old dataset.

- **ZIP0197E**

ZIP0197E the program has encountered too many SELECT arguments.

Redesign the selection arguments. Make use of generic or wildcard selection arguments. Move some of the SELECT statements after an additional FROMDD/FROMDS for the same input dataset(s). Approximately 4000 bytes of storage is available to build the current selection argument list in. Calculate storage required as total length of all arguments plus 10 bytes per argument.

- ZIP0198E

ZIP0198E Catalog cleanup required for dsname group:

Use ISPF option 3.1 or 3.4 to list datasets with names conforming to the group name shown and delete datasets no longer required.

- ZIP0199E

ZIP0199E OBTAIN failed for input dataset:

The program was unable to read the label for the input dataset. Verify that the input dataset is actually on the volume specified in the dataset catalog or via UNIT= VOL= keywords in the JCL.

- ZIP0200E

ZIP0200E unable to create new ZIP dataset:

The program generates a new o/p dataset name that includes a random numeric component of 2 digits. 10 attempts have been made and all names generated are already catalogued datasets. The user should use ISPF 3.4 to examine the existing dataset names of the form shown and delete those not required to be kept.

- ZIP0201E

ZIP0201E cannot update ZIP, dsname too long:

The existing ZIP dataset has a dsname longer than 41 characters. The program needs to create an output dataset with a name derived by appending .Zn to the end of the old ZIP dataset name. Rename the existing ZIP dataset to have a shorter name.

- ZIP0202E

ZIP0202E not a ZIP dataset:

The output dataset is neither empty nor is it a valid ZIP dataset. If the output dataset is not empty then it must be a valid ZIP file that the program can read and process per UPDATE or FRESHEN rules. The OVERWRIT keyword may be used to force the program to overwrite the output dataset without regard for its state or content.

- ZIP0203E

ZIP0203E I/P ZIP dataset format error:

The program is reading an existing ZIP dataset member by member. The program did not find a valid or complete member header when one was expected. The accompanying diagnostic output shows the file offset, dump of last header successfully processed and dump of up to 256 bytes starting at that offset.

- ZIP0204E

ZIP0204E Unexpected EOF encountered while reading a ZIP dataset

While reading the ZIP dataset the program encountered an unexpected end of file condition. The accompanying diagnostic output shows the file offset, dump of last header successfully processed and dump of up to 256 bytes starting at that offset.

- ZIP0205E

ZIP0205E TODD without OVERWRITE for existing ZIP dataset, use TODS

The program can only process an existing ZIP dataset correctly if it is specified by dataset name, using either the TODS(..) operand or the TODS control statement. Alternatively, an existing dataset can be overwritten by specifying the OVERWRITE option following the TODD(..) operand or on the TODD control statement. When OVERWRITE is specified the program simply writes over the dataset.

- ZIP0206I

ZIP0206I terminating with COND CODE nnnn

The program is terminating with the COND CODE value shown. If this value is non-zero then the control listing should be examined for indications of problems encountered. The message also contains information about the day of week, time of day and cpu utilisation

- ZIP0207I

ZIP0207I DEMOKEY value missing or invalid for today's date, terminating.

The program requires a key, specific to today's date, to be provided via the DEMOKEY control statement. A different key is required for each day. Today's and Tomorrow's keys are always available on our web site. This compliance mechanism makes it possible for us to let you perform short-term ad hoc trials of our valuable product without any reference to us. If you would like to conduct an official trial of the software without the bother of getting the new key from the web site each day then please go to <http://www.slikzip.com> and follow the FREE TRIAL link. When ASE has received a suitably completed Trial Agreement then we will supply a single DEMOKEY value that will be effective for the full period of the trial.

- ZIP0208I

ZIP0208E PASSWORD is not supported by the GZIP file format.

An unsupported option has been chosen. Encryption is not supported by the GZIP file format. Remove the PASSWORD keyword and rerun the job. Alternatively you may wish to use the ZIP file format.

- ZIP0209I

ZIP0209E UPDATE and FRESHEN are not supported for the GZIP file format

An unsupported option has been chosen. Remove the UPDATE or FRESHEN keyword and rerun the job. Alternatively you may wish to use the ZIP file format.

- ZIP0210I

ZIP0210E OVERWRITE/REPLACE not specified when GZIP output dataset exists

The program determined that the output dataset exists. The program does not support UPDATE or FRESHEN process options for existing GZIP datasets. Use the OVERWRITE or REPLACE keywords to cause the program to write the new GZIP dataset over the existing dataset. Alternatively, change the GZIP target designation to a NEW dataset.

- ZIP0211I

ZIP0211E NODDESC is not supported for the GZIP file format

An unsupported option has been chosen. Remove the NODDESC keyword and rerun the job. Alternatively you may wish to use the ZIP file format.

- ZIP0212E

ZIP0212E syntax error in table name: tablename

The TRANSTAB statement contains a table name that is invalid. From 1 to 40 Characters, A-Z, 0-9 and @, #, \$ and _ only. Change the table name to correct the error and rerun.

- ZIP0213W

ZIP0213W WARNING: License key expires in nn days

ZIP0213W WARNING: License key expires today

Your license to use Slikzip expires in the specified number of days. After the specified number of days you will not be able to use this program. Please contact ASE to renew the license and obtain a new license key.

- ZIP0214I

ZIP0214I KEY value in licence csect expired or invalid , terminating.

During installation of the program key values were specified in the license csect which was linked with the SLIKZIP program. The key values are invalid or have expired. Check that the key values have been correctly specified. If so contact ASE to obtain a valid key.

- ZIP0215W

ZIP0215W Alias xxxxxxxx ignored, member yyyyyyy not selected

Alias xxxxxxxx matched the selection criteria but was not included in the output zip file because the member that it references was not included or noalias was specified .

- ZIP0216I

ZIP0216I xxxxxxxx is an alias of yyyyyyy

An alias entry for xxxxxxxx has been created in the header information for member yyyyyyy of the zip file.

- ZIP0217I

ZIP0217I ignoring data descriptor fields, USELOCALHEADERVALUES specified

The keyword USELOCALHEADERVALUES has been specified and the zip data is followed by a data descriptor. Values in the descriptor for crc , compressed size and uncompressed size will be ignored and the values will be obtained from the local header.

- ZIP0218E

ZIP0218E USELOCALHEADERVALUES ignored, local header values unsuitable

The USELOCALHEADERVALUES keyword was specified but one or more of the crc and length fields in the the header are zero. Therefore the values in the data descriptor will be used to extract the file.

- ZIP0219I

ZIP0219I LOCHCRC: xxxxxxxxh LOCHCSIZ: xxxxxxxxh LOCHUSIZ: xxxxxxxxh

The values of the CRC, compressed size and uncompressed size from the header are shown.

- ZIP0220E

ZIP0220E O/P ZIP dataset is a PDS and member name not specified

The program is attempting to create a ZIP dataset in a PDS and a member name has not been specified in the JCL on the TODS statement.

- ZIP0221E

ZIP0221E O/P ZIP PDS member already exists and OVERWRITE not specified

The program is attempting to create a ZIP dataset in a PDS and the member already exists. ZIP files in PDS members cannot be updated. The OVERWRITE keyword must be specified for existing members.

- ZIP0222E

ZIP0222E Error during BLDL, ZIP PDS member not found

The program is attempting to read or write a member in a PDS. A BLDL macro for the member has failed. The member does not exist.

- ZIP0223E

ZIP0223I waiting for access to file

A file that the program is trying to update is in use by another job/user. SLiKziP will wait for the file to be released before continuing. Message ZIP0224I contains the name of the file.

- ZIP0224E

ZIP0224I dsn member

The program is waiting for access to the specified file.

- ZIP0225E

ZIP0225I Access Acquired

The program has acquired access to the file that it was waiting for and will continue processing.

- ZIP0226E

ZIP0226I Invalid member name: membername

The member name specified is invalid. Perhaps you have specified a selection argument instead of a member name. The syntax for specifying a pds in a FROMDS statement is:
pdsname(member(selection)).

Publication number: SZ02-0301-12

This manual serves as a reference source for users of the SLiKZiP product. Please use this form to comment on the publication in terms of its organization or content with the understanding that ASE may use or distribute the information that you supply in any way that it believes appropriate without incurring an obligation to you.

If you require further copies of this publication please order them through your ASE representative. If you require assistance in the installation, configuration or use of the SLiKZiP product please contact ASE for support.

Suggested aspects that might be addressed by your comments include:

Clarity Accuracy Completeness Organization Examples Legibility

Please give your name and address if you require a reply:

name:

email:

Address:

Occupation:

Comments: