English

FUJITSU Software BS2000

BS2ZIP V1.2G

Zip Archiving in BS2000

User Guide

Edition November 2014

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2008

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2008.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Contents

1 Introduction

BS2ZIP is a BS2000 TU application with a simple, easy-to-use SDF-statements interface: version V1.2 can be used with BS2000/OSD V7.0 and higher.

1.1 Objectives and target groups of this manual

This manual is intended both for privileged and nonprivileged BS2000 users.

1.2 Summary of contents

BS2ZIP provides the following functions:

- The creation of a BS2ZIP container compliant with WinZip® (PKZIP 4.5)
- The addition of all kinds of BS2000 files and organization: PAM, ISAM, SAM and PLAM (separate library elements also being possible) using the compliant compression method “Zlib”: RFC 1950 Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler: jloup@gzip.org / madler@alumni.caltech.edu
- The extraction and restore of the BS2000 files with their original DMS attributes
- The display of the available files included in the BS2ZIP container
- The support of multiple files in the Bs2ZIP container
- Support of standard Zip 2.0 encryption when files are added or extracted
- The interoperability with Windows environment:
  - Under BS2000:
    Opening of WinZip made transferred archive files in BS2000 as BS2ZIP container (transfer via ftp)
    Extracting files as text files (SAM files) with possible ISO8859F/WINANSI conversion into EDF04F on request
    Extracting files as raw files (binary) on request (PAM files)
  - Under Windows:
    Opening of transferred BS2ZIP container using WinZip (transfer via ftp)
    Conversion EBCDIC text files (such as SAM/ISAM) ensured by the application (conversion EDF0x into ISO8859F/WINANSI).
    All other elements are considered as “binary” file
  - Support of transfer using openFT < V11.0 by converting the file format of BS2ZIP containers (from SAM with RECORD-FORMAT=U to PAM and vice versa).
- Interoperability with Linux/Unix system environment.
  The possibility of reading and extracting file from GZIP archives.

A program interface in C++ is provided that offers all the functionalities of the TU application.

Readme file

The functional changes to the current product version and revisions to this manual are described in the product-specific Readme file.

Readme files are available to you online in addition to the product manuals under the various products at *http://manuals.ts.fujitsu.com*. You will also find the Readme files on the Softbook DVD.

Information under BS2000

When a Readme file exists for a product version, you will find the following file on the BS2000 system:

```
SYSRME.<product>.<version>.<lang>
```

This file contains brief information on the Readme file in English or German (<lang>=E/D). You can view this information on screen using the `/SHOW-FILE` command or an editor. The `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` command shows the user ID under which the product's files are stored.

Additional product information

Current information, version and hardware dependencies, and instructions for installing and using a product version are contained in the associated Release Notice. These Release Notices are available online at *http://manuals.ts.fujitsu.com*.

1.3 Changes since the last edition of the manual

The following changes have been implemented for BS2ZIP V1.2G:

- ADD-FILE statement:
  In the operands FROM-FILE and EXCEPT-FILE-NAME a file containing a list of the files to be added to the ZIP archive or to be excluded from the addition can be specified.
- ADD-FILE and EXTRACT-FILE statements:
  The operands LOGGING controls the amount of the message output.

1.4 Notational conventions

The following typographical elements are used in this manual:

ℹ	For notes on particularly important information
⚠	This symbol designates special information that points out the possibility that data can be lost or that other serious damage may occur.
[ ]	References to other publications within the text are given in abbreviated form followed by numbers; the full titles are listed in the “References” section at the back of this manual.
`input`	Inputs and system outputs in examples are shown in typewriter font

2 BS2ZIP application overview

2.1 ZIP containers

The BS2ZIP application enables ZIP containers to be created on BS2000 systems, and files to be added to and extracted from these containers. The ZIP compression used is the Zlib method.

This application is also able to read ZIP containers built on foreign platforms (Windows, Unix systems, Linux) provided they use the Zlib compression method and provided they are compliant with PKZIP 4.5 or GZIP.

The containers created in BS2000 can also be reopened on foreign platforms if they were created in WinZip-compatible format.

The command to start the BS2ZIP application is /START-ZIP-MANAGER or /START-ZIP.

After starting the application, the user has to open an existing container or to create a new one with the OPEN-ZIP-CONTAINER statement. In the case of the creation of a new container the user has to select the appropriate container format (operand FORMAT of the statement) according to the usage planned for the container to be created.

Please respect the following rules to select the appropriate format:

1. **Container creation**:
   - if you plan to use the container on BS2000 platforms only, you can create it in BS2000 format However, if you zip SAM or ISAM file including special print control characters, use the BS2000 format.
   - if you plan to export the container on foreign platforms, create your container in *WINZIP compatible format (default one) only.
2. **Container open**:
   - you don’ t need to specify the container format at open of an existing container. The program finds itself the container format. However, if you specify the format, it must be the real format of the container otherwise the OPEN-ZIP-CONTAINER statement is rejected. Once, the container is opened, you can add or extract files.

The following table summarizes the default behavior of the BS2ZIP application for the different file types according to the container open format:

ZIP Format	**File Access**					
	SAM		**ISAM**		**PAM/PLAM**	
WINZIP compatible	A:	Data only are saved WIN-ANSI conversion if file is EBCDIC encoded; CRLF (0D0A) are inserted as record delimiters	A:	Data and keys are saved WIN-ANSI conversion if file is EBCDIC encoded; CRLF (0D0A) are inserted as record delimiters	A:	Binary save without any conversion
	X:	Files are extracted with same organi-zation than the original file. EBCDIC conversion if necessary Record rebuild according to 0D0A delimiter *Default processing if no organization info is associated to the file*	X:	Files are extracted with same organi-zation than the original file. EBCDIC conversion if necessary Record rebuild according to 0D0A delimiter	X:	Files are extracted with same organi-zation than the original file. Binary rebuild
BS2000	A:	Record length saved with data No conversion	A:	Record length saved with data No conversion	A:	Binary save
	X:	Files are extracted with same organi-zation than the original file. No conversion	X:	Files are extracted with same organi-zation than the original file. No conversion	X:	Files are extracted with same organi-zation than the original file. Binary rebuild

A: Add
X: Extract

Using the ADD-FILE and EXTRACT-FILE statements you can modify this default behavior. See those statements for details.

Note

In the case of a WinZip-compatible container it is assumed that a CRLF (0D0A) indicates the end of a data record. The original file may not contain any such binary values, otherwise the file is “spoilt” when it is extracted. A WinZip-compatible container should not contain any non-printable SAM or ISAM files.

ADD-FILE

CHARACTER CONVERSION	**Meaning**
*BY-CONTAINER-FORMAT	Default value. Behavior determined by the ZIP container open format
*NO	No conversion is forced
*TO-EBCDIC	EBCDIC conversion is forced (only for SAM/ISAM files in ASCII encoding)
*TO-WIN-ANSI	WIN-ANSI conversion is forced (only for SAM/ISAM files in EBCDIC encoding)

EXTRACT-FILE

The open format determined by ZIP container format can be modified at EXTRACT-FILE time:

DATA-TYPE	**Container format**	
	WINZIP-COMPATIBLE	**BS2000**
*NOT-SPECIFIED	If no file info found in ZIP container Then *CHARACTER is assumed Else file info is used - EBCDIC conversion done according to file type	If no file info found Then error Else file info is used - no conversion
*CHARACTER	File is extracted as a SAM file Records are assumed to be delimited by 0D0A EBCDIC conversion performed	Statement rejected
*BINARY	File is extracted as a PAM file No conversion performed	Statement rejected
*SAM-BINARY	File is extracted as a SAM file, REC-FORM=U No conversion performed	Statement rejected

CHARACTER-CONVERSION	**Meaning**
*BY-CONTAINER-FORMAT	Default value. Behavior determined by the ZIP container open format
*NO	No conversion is forced
TO-WIN-ANSI	WIN-ANSI conversion is forced (only for extraction to SAM/ISAM file)
*TO-EBCDIC	EBCDIC conversion is forced (only for extraction to SAM/ISAM file)

Block format when extracting to various disk formats

When extracting a file, BS2ZIP takes into account the disk format in which the file is stored, the block format (BLOCK-CONTROL-INFO) of the original file which is stored in the archive, and the specification in the BLOCK-CONTROL-INFO operand which controls whether the block format must be retained. As a result of this, in some cases the file cannot be extracted, or a block format which differs from that of the original file must be selected.

Extracting to K disks

The extracted files contain the block format of the original file.

Extracting to NK2 disks

Files with the PAMKEY block format cannot be extracted using the BLOCK-CONTROL= *KEEP option.

Depending on the archive format, the BLOCK-CONTROL=*IGNORE option is used to set the block format as follows:

Properties of the original file		**Specification in the BLOCK-CONTROL operand**			
		(BS2000 format)		(WINZIP-COMPATIBLE format)	
FCB type	**BLK-CTRL**	***KEEP**	***IGNORE**	***KEEP**	***IGNORE**
PAM	PAMKEY	Not extracted	NO	Not extracted	NO
PAM	DATA	Like original	Like original	Like original	Like original
PAM	NO	Like original	Like original	Like original	Like original
SAM	PAMKEY	Not extracted	DATA	Not extracted	DATA
SAM	DATA	Like original	Like original	Like original	Like original
ISAM	PAMKEY	Not extracted	DATA2K	Not extracted	DATA2K
ISAM	DATA	Like original	Like original	Like original	Like original

Extracting to NK4 disks, file with even blocking factor

Files with the PAMKEY block format cannot be extracted using the BLOCK-CONTROL=*KEEP option.

Depending on the archive format, the BLOCK-CONTROL=*IGNORE option is used to set the block format as follows:

Properties of the original file		**Specification in the BLOCK-CONTROL operand**			
		(BS2000 format)		(WINZIP-COMPATIBLE format)	
FCB type	**BLK-CTRL**	***KEEP**	***IGNORE**	***KEEP**	***IGNORE**
PAM	PAMKEY	Not extracted	NO	Not extracted	NO
PAM	DATA	Like original	Like original	Like original	Like original
PAM	NO	Like original	Like original	Like original	Like original
SAM	PAMKEY	Not extracted	NO	Not extracted	NO
SAM	DATA	Like original	Like original	Like original	Like original
ISAM	PAMKEY	Not extracted	DATA4K	Not extracted	DATA4K
ISAM	DATA	DATA4K[1]	DATA4K	Like original	Like original

[1] Only ISAM files with BLOCK-CONTROL-INFO=DATA4K and BUF-LEN=*STD(n) with n=4, 8, 12 or 16 can be extracted.

Extracting to NK4 disks, file with odd blocking factor

Files with an odd blocking factor cannot be extracted to NK4 disks.

2.2 Transferring files

In BS2000, ZIP containers are always PAM files. They can be transferred to other BS2000 systems or to systems with other operating systems (MS/Windows, Unix, Linux, zOs). In non-BS2000 systems only ZIP containers in WinZip-compatible format can be used.

A ZIP container in PAM file format can be transferred to another system using ftp or openFT V11.0 and higher. Binary mode must be set for the transfer.

When a file is to be transferred to or from a non-BS2000 system using openFT, openFT < V11.0 only accepts SAM files with RECORD-FORMAT=U. A ZIP container in PAM file format can be converted to a SAM file with RECORD-FORMAT=U or vice versa using the //CONVERT-ZIP-CONTAINER statement. In this case the converted file is stored as a copy in the output file.

If no name is specified for these output files, BS2ZIP generates standard file names for these output files as follows:

<name>.SAM[.ZIP] or **<name>.PAM[.ZIP]**

where <name> is a basename with a limited length.

Examples

Name of the ZIP container	File format of the ZIP container	Name of the copy
XYZ	PAM	XYZ.SAM
XYZ	SAM (with RECORD-FORMAT=U)	XYZ.PAM
XYZ.PAM	PAM	XYZ.SAM
XYZ.SAM	SAM (with RECORD-FORMAT=U)	XYZ.PAM
XYZ.ZIP	PAM	XYZ.SAM.ZIP
XYZ.ZIP	SAM (with RECORD-FORMAT=U)	XYZ.PAM.ZIP
XYZ.PAM.ZIP	PAM	XYZ.SAM.ZIP
XYZ.SAM.ZIP	SAM (with RECORD-FORMAT=U)	XYZ.PAM.ZIP

Notes

– When the resultant file name, including catalog and user IDs, exceeds 54 characters, BS2ZIP truncates the basename (XYZ in the example).

– The converted file is created with the userid and the default cat-id of the user, even if the original file is on another userid or catid.

2.3 Encrypting data

When a file is added to a ZIP file, it can be encrypted. This encryption offers such a file in the ZIP container a degree of protection against unauthorized access. Anyone wishing to read the data in the file must decrypt it again when it is extracted and requires the appropriate key (the crypto password) to do this. The ZIP container itself is not protected by this encryption.

BS2ZIP supports the Zip 2.0 encryption mechanism and is consequently compatible with WinZip and many other Zip tools. Zip 2.0 encryption is relatively weak and does not offer sufficient protection against special password recovery tools.

BS2ZIP encrypts files when they are added to the archive and decrypts them again when they are extracted, provided encryption was enabled using the MODIFY-ZIP-OPTIONS statement. The appropriate crypto password must also be specified when the function is enabled.

If a ZIP container contains files with different passwords, encryption must be specified again with the requisite password before a file is extracted.
When extraction takes place to non-BS2000 systems, WinZip or the Zip tool would have to be restarted for each password in such cases.

2.4 File types supported

BS2ZIP supports SAM, ISAM and PAM files including PLAM libraries.

ISAM Files

ISAM files with secundary keys are not supported.

Index and data of the resulting ISAM file after the extract statement are never separated. The resulting file after extraction may be greater than the original one.

#The padding (PADDING-FACTOR) associated to an extracted ISAM file is always the default one (DMS limitation).

PLAM Libraries

BS2ZIP processes PLAM libraries as a whole file.

Library elements can also be processed separately. The file information which is stored for the element in the PLAM library (if none is stored the PLAM default settings are used) is transferred to the ZIP archive and evaluated when extraction takes place.

Temporary Files

BS2ZIP enables temporary files to be added and extracted. These files are recorded under their cataloged file names, i.e. including the tempfile prefix.

As the tempfile prefix is assigned by the system on a task-specific basis, no files with a different tempfile prefix can be created in a task. It is only possible to extract a file with a tempfile prefix without renaming it if the file was added in the same task. Otherwise the file must be renamed explicitly when it is extracted.

Load modules of a PAMKEY disk

Load modules of a PAMKEY disk that have been added to a container cannot be extracted on a NK disk, even if the operand BLOCK-CONTROL-INFO=*IGNORE is set, because the generated NK load module does not contain all the necessary information needed to be loaded correctly on a NK disk.

If the operand BLOCK-CONTROL-INFO=*KEEP is set, you will receive the DMS error message:

```
DMS0D80 ALLOCATED DISK SPACE DOES NOT MATCH WITH THE REQUESTED FILE FORMAT.
```

2.5 File Extents and user-specific resources

It can happen, that the maximum of extents for the ZIP container is reached (310). If the amount of data to compress is important, it is advised to estimate the final size of the container and to open it by link to avoid this. See OPEN-ZIP-CONTAINER “Notes” on page 44 for details and example.

Ensure that there is enough space in the user ID where the ZIP container is built. If there is not enough space, the container may be incoherent and so unusable. Its content cannot be accessed any more.

If you want to process a large number of files using BS2ZIP, sufficient user address space must be available under your user ID (see ADDRESS-SPACE-LIMIT in the user entry).

2.6 K2 support

Pressing the K2 key has a limited scope. It allows the program to be interrupted in statement mode (//). After an interruption using the K2 key, the user is in command mode (/).

If K2 is pressed during the ADD-FILE, EXTRACT-FILE or SHOW-FILE-ATTRIBUTES statements, BS2ZIP interrupts processing or output with the query message `SZP0208`. Users then have the following options:

- The can simply continue current processing.
- They can abort current processing. When they do this, they return to statement mode (//). The statement may need to be reissued for files which had not been processed by the time the interruption occurred.

2.7 Crash resistance

If the processing of the ADD-FILE statement is abnormally terminated (e.g. in case of system crash or power failure), a zip container may be inconsistent. This inconsistency occurs because the directory containing the information about the container content has been overwritten.

The following mechanism enables the recovery of such corrupt containers: Before the processing of the first ADD-FILE statement, a copy of the container directory and the name of the container are saved in a backup file named as BS2ZIP.YYYY-MM-DD.HHMMSS.BAK. This file is deleted when the BS2ZIP application is normally terminated.

After abnormal termination the file is not deleted. The next time when the inconsistent container is opened, the system automatically searches for the corresponding backup file and restores the container in the state before the start of the last BS2ZIP session. However, the data added during that session still exist in the archive, but are not accessible. The repaired container is therefore bigger than the one before the crash. To remove those useless data, execute the REORGANIZE-ZIP-CONTAINER statement.

If several user IDs have access to the same container, the following constraint must be observered:

The backup file is always saved under the current user id. If the container is opened under a different user ID after the abnormal termination, the backup file cannot be found automatically. Therefore, the opening of the container open fails. In this case, the system administration has to locate the user ID of the backup file. The container then must be opened and restored under that user ID.

3 Command and statement interfaces

3.1 Starting and terminating the BS2ZIP applications

The BS2ZIP application has an SDF interface. If you enter a question mark, all possible inputs on the relevant level are displayed on the screen.
SDF is described in the manual „SDF Dialog Interface“. You will find the SDF syntax description in the “Commands” manual [2].

3.1.1 Command for starting the BS2ZIP application

The BS2ZIP application is started by means of the /START-ZIP-MANAGER command:

START-ZIP-MANAGER	Alias: **ZIP-MANAGER**
VERSION = ***STD** / <product-version>	
,**MONJV** = ***NONE** / <filename 1..54 without-gen-vers>	
,**CPU-LIMIT** = ***JOB-REST** / <integer 1..32767 *seconds*>	

VERSION = *STD / <product-version>
Specifies the BS2ZIP version which is to be used.
*STD is the default value: the version selected using the SELECT-PRODUCT-VERSION command is loaded. If no version was selected, the highest available version is loaded.

MONJV =
Name of the job variable that is to monitor the program.

MONJV = *NONE
No monitoring job variable is used.

MONJV = <filename 1..54 without-gen-vers>
Name of the job variable that is to monitor the program. This operand is available only to users who have the JV software product (see also „program monitoring“ in the “Job Variables” manual [3]).

CPU-LIMIT = *JOB-REST / <integer 1..32767>
Maximum CPU time, in seconds, allowed for execution of the program.
If the program execution exceeds the specified time, it is interrupted in interactive mode and message EXC0075 is issued. The user can then request a dump, abort the program or continue the program run. In batch mode the program is terminated.

*JOB-REST is the default value: no more than the maximum remaining CPU time is to be used. Detailed information about “Time limits in BS2000” is provided in the “Commands” manual [2].

3.1.2 Terminating the BS2ZIP application

Entering the END statement terminates the BS2ZIP application.

3.1.3 Messages

Information on program execution of BS2ZIP, processing the statements, and errors which have occurred is provided in messages of the message class SZP (message keys `SZP`xxxx).

The command or standard statement HELP-MSG-INFORMATION enables you to query more information on the meaning of the messages.

All messages of BS2ZIP can also be found using the HTML application “System messages” on the Manual Server (URL: *http://manuals.ts.fujitsu.com*) and on the “BS2000 SoftBooks” DVD.

3.2 Statements description

BS2ZIP application offers the following functions:

Statements	**Meaning**
ADD-FILE	Adds one or several BS2000 files to the currently opened ZIP archive.
CONVERT-ZIP-CONTAINER	Converts a ZIP container from PAM to SAM format with REC-FORM=U and vice versa (for data transfer using openFT < V11.0).
DELETE-FILE	Deletes one or several BS2000 files from the currently opened ZIP archive.
END	Closes the currently opened ZIP archive and stops the program.
EXTRACT-FILE	Extracts one or several files from the currently opened ZIP archive.
MODIFY-ZIP-OPTIONS	Defines defaults for BS2ZIP processing, e.g. for encryption
OPEN-ZIP-CONTAINER	Opens a ZIP archive. It eventually creates it if it does not exist.
REORGANIZE-ZIP-CONTAINER	Reorganizes the content of a ZIP container.
SHOW-FILE-ATTRI-BUTES	Displays the contents of the currently opened ZIP archive.
START-TRACE	Activates an internal trace of the processing in a file.
STOP-TRACE	Terminates the trace processing.

In addition to these statements, you can also use SDF standard statements. These are displayed in a statement list which you are shown after you enter a question mark. The SDF standard statements are described in detail in the “SDF Dialog Interface” [1].

3.2.1 ADD-FILE

This statement adds one or several files to the currently opened ZIP file. The statement is rejected when the ZIP container is opened in read mode.

SAM, ISAM, PAM and PLAM files are supported. Elements of a PLAM library can also optionally be added to the ZIP container.

```
ADD-FILE

 FROM-FILE = *ALL / *FROM-FILE(...) / *FROM-LIBRARY-ELEMENT(...) / *LIBRARY-ELEMENT(...) /
                list-poss(20): <filename 1..54 without-gen-vers with-wild (80)>

   *FROM-FILE(...)
    |   LIST-FILE-NAME = <filename 1..54 without-gen-vers>

   *FROM-LIBRARY-ELEMENT(...)
    |
    |   LIBRARY = <filename 1..54 without-gen-vers>
    |
    |  ,ELEMENT = <composed-name 1..64 with-under>

 *LIBRARY-ELEMENT(...)
    |   LIBRARY = <filename 1..54 without-vers>
    |
    |  ,ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
    |
    |     <composed-name 1..64 with-under with-wild(132)>(...)
    |      |   VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT /
    |      |                <composed-name 1..24 with-under with-wild(40)>
    |      |  ,BASE = *STD / <composed-name 1..24 with-under with-wild(40)>
    |
    |  ,TYPE = <alphanum-name 1..8 with-wild(12)>

,TO-FILE = *BY-SOURCE / <composed-name 1..98 with-underscore with-wild-constr(132)>(...)

   <composed-name 1..98 with-underscore with-wild-constr(132)>(...)
    |   WITH-VERSION = *STD / *YES / *NO
    |
    |  ,WITH-TYPE = *STD / *YES / *NO

,CHARACTER-CONVERSION = *BY-CONTAINER-FORMAT / *NO / *TO-EBCDIC / *TO-WIN-ANSI

,COMPRESSION-LEVEL = *STD / *NONE / *BEST-SPEED / *BEST-COMPRESSION

,DATA-TYPE = *NOT-SPECIFIED / *CHARACTER / *BINARY

,EXCEPT-FILE-NAME = *NONE / *FROM-FILE(...) / *FROM-LIBRARY-ELEMENT(...) /
                      list-poss(20): <filename 1..54 without-vers with-wild (80)>

   *FROM-FILE(...)
    |  LIST-FILE-NAME = <filename 1..54 without-gen-vers>

   *FROM-LIBRARY-ELEMENT(...)
    |
    |  LIBRARY = <filename 1..54 without-gen-vers>
    |
    |  ELEMENT = <composed-name 1..64 with-under>

,DELETE-SOURCE = *NO / *YES

,LOGGING = *MINIMUM / *MAXIMUM
```

FROM-FILE =
Specifies which files or which library elements are to be added to the ZIP container. The user can select individual files from the set of files specified here by means of the operand EXCEPT-FILE-NAME.

FROM-FILE = *ALL
All the supported files of the current userid are added to the ZIP file.

FROM-FILE = <filename 1..54 without-vers with-wild(80)>
The specified file is added to the ZIP file. When wildcards are used, all the supported files matching the pattern are added to the ZIP file.

FROM-FILE = *FROM-FILE(...)
The path names of the files to be added to the ZIP container are to be taken from a file. The nonprivileged caller must be owner or co-owner of this file. This list file must be a SAM file with variable-length records containing one path name per record. Lower-case characters may be used, but they will be assimilated to upper-case characters. File names must be fully qualified with or without catid or userid.
The list file can be created, for instance, by means of the DMS command SHOW-FILE-ATTRIBUTES.

LIST-FILE-NAME = <filename 1..54 without-gen-vers>
Path name of the list file.

FROM-FILE = *FROM-LIBRARY-ELEMENT(...)
The path names of the files which are to be added to the ZIP container are to be taken from a PLAM library element (type S). The library element consists of records of variable length and contains one path name per record. Lower-case characters may be used, but they will be assimilated to upper-case characters. File names must be fully qualified with or without catid or userid.

LIBRARY = <filename 1..54 without-gen-vers>
Name of the PLAM library.

ELEMENT = <composed-name 1..64 with-under>
Name of the type-S element containing the list file. The element of the highest existing version is used.

FROM-FILE = *LIBRARY-ELEMENT(...)
Elements of a PLAM library are to be added to the ZIP container. An element is fully defined by its name, its type, and the version number.

LIBRARY = <filename 1..54 without-vers>
Name of the PLAM library from which elements to be added.

ELEMENT = <composed-name 1..64 with-under with-wild(132)>(...)
The specified element is added to the ZIP container if the element type is supported. Wildcards enable more than one element to be specified.

VERSION = *HIGHEST-EXISTING / *UPPER-LIMIT / <composed-name 1..24 with-under with-wild(40)>
Version of the element which is to be displayed. The default value is *HIGHEST-EXISTING, i.e. the last element in alphabetical order.
*UPPER-LIMIT specifies the highest possible version X'FF' (displayed with the character '@').
If the version is specified using wildcards and library elements with the same name exist in versions which are affected by specifying wildcards, all these library elements are output.

BASE = *STD / <composed-name 1..24 with-under with-wild(40)>
Specifies the base if multiple data elements exist for the specified version.

TYPE = <alphanum-name 1..8 with-wild (12)>
Type of library element. The types D, J, M, P, S and X are supported, as well as the user-defined types based on these.
When the type is specified using wildcards, the name consists of up to 12 alphanumeric characters.

FROM-FILE = list-poss(20): <filename 1..54 without-vers with-wild(80)>
The path names of the files to be added to the ZIP container are specified directly. A list of up to 20 names may be specified.

The file names may be specified as fully or partially qualified names, with or without a catalog/user ID. If required, the file name is extended by the user ID of the request and the default catalog ID.

You can also use wildcard syntax to select the files. In this case, the supported files matching the pattern are added to the ZIP file.

TO-FILE = *BY-SOURCE
Specifies that the input files are to be registered in the container with their current name without catid or userid.

TO-FILE = <composed-name 1..64 with-underscore with-wild-constr(132)>
The file is stored in the container with the specified name.
If wildcards are used in the FROM-FILE operand, wildcards can be used in a construction string to specify how the new names are to be formed in the container (see SDF rules for wildcard construction in the “Commands” manual [2]).

If list of file names is specified in the FROM-FILE operand, only TO-FILES = *BY-SOURCE is accepted or a value without wildcards, but eventually terminated by a period character. In this case, the TO-FILES value is used as a prefix.

Example:

ADD-FILE FROM-FILE=MYFILE,TO-FILE=XXX. is accepted.
All the file names contained in MYFILE are prefixed with XXX.

The default setting for library elements is that the specified name is also assigned a suffix containing the type and version of the element (format: <to-file>.<type>.<version>). The option of creating this suffix can be controlled in the following operands:

WITH-VERSION = <u>*STD</u> / *YES / *NO
Evaluated only for library elements:
Specifies whether the suffix is to contain the element version.
*STD specifies *YES as the default.

i An element with VERSION=*UPPER-LIMIT is assigned a 'U' in the suffix instead of the character '@' (e.g. in the case of type S, TO-FILE=myelem1 becomes MYELEM1.U.S).

WITH-TYPE = <u>*STD</u> / *YES / *NO
Evaluated only for library elements:
Specifies whether the suffix is to contain the element type.
*STD specifies *YES as the default.

CHARACTER-CONVERSION = <u>*BY-CONTAINER-FORMAT</u>
The input files are converted according to the format of the ZIP file.

CHARACTER-CONVERSION = *NO
The input files are not converted.

CHARACTER-CONVERSION = *TO-EBCDIC
The ASCII encoded input files are converted to EBCDIC before compression.
Only SAM/ISAM files are converted.

CHARACTER-CONVERSION = *TO-WIN-ANSI
The EBCDIC encoded input files are converted to WIN-ANSI before compression.
Only SAM/ISAM files are converted.

COMPRESSION-LEVEL =
This operand determines the compression level. It allows adjusting the speed/compression size ratio.

COMPRESSION-LEVEL = <u>*STD</u>
This compression mode selects a compromise between the speed and the compression size.

COMPRESSION-LEVEL = *BEST-SPEED
This compression mode selects the fastest compression. Such a mode results larger container.

COMPRESSION-LEVEL = *BEST-COMPRESSION
This compression mode selects the best compression. Such a mode results smaller container but requires larger processing time.

COMPRESSION-LEVEL = *NONE
The file is added without compression.

DATA-TYPE =
The DATA-TYPE operand controls the file inclusion. This option is only relevant for SAM files to be added to a WinZip compatible container.

DATA-TYPE	Container format WINZIP compatible	Container format BS2000
<u>*NOT-SPECIFIED</u>	– SAM/ISAM files: assumed as text files. A record = a line of text – PAM files: Assumed as binary files.	– SAM/ISAM files: assumed as text files. A record = a line of text – PAM files: Assumed as binary files.
*CHARACTER	Not relevant. Processed as DATA-TYPE=*NOT-SPECIFIED	Not relevant. Processed as DATA-TYPE=*NOT-SPECIFIED
*BINARY	– SAM files: Assumed as binary files. The file can only be extracted as PAM file. – ISAM files: Not relevant. Assumed as text files. A record = A line of text – PAM files: Processed as binary files.	Not relevant. Processed as DATA-TYPE=*NOT-SPECIFIED

EXCEPT-FILE-NAME =
Serves to specify files that are to be excluded from zipping.

EXCEPT-FILE-NAME = <u>*NONE</u>
All files specified with the FROM-FILE operand are to be archived.

EXCEPT-FILE-NAME = *FROM-FILE(...)
The path names of the files to be excluded from zipping are to be taken from a file. The nonprivileged caller must be owner or co-owner of this file. This list file must be a SAM file with variable-length records containing one path name per record. Lower-case characters may be used, but they will be assimilated to upper-case characters. File names must be fully qualified with or without catid or userid.
The list file can be created, for instance, by means of the DMS command SHOW-FILE-ATTRIBUTES.

LIST-FILE-NAME = <filename 1..54 without-gen-vers>
Path name of the list file.

EXCEPT-FILE-NAME = *FROM-LIBRARY-ELEMENT(...)
The path names of the files which are not to be zipped are taken from a PLAM library element (type S). The library element consists of records of variable length and contains one path name per record. Lower-case characters may be used, but they will be assimilated to upper-case characters. File names must be fully qualified with or without catid or userid.

LIBRARY = <filename 1..54 without-gen-vers>
Name of the PLAM library.

ELEMENT = <composed-name 1..64 with-under>
Name of the type-S element containing the list file. The element of the highest existing version is used.

EXCEPT-FILE-NAME = list-poss(20): <filename 1..54 without-vers with-wild(80)>
The path names of the files to be excluded from archival are specified directly. A list of up to 20 names may be specified.

The first character of the file names must not be a hyphen. The file names may be specified as fully or partially qualified names, with or without a catalog/user ID. If required, the file name is extended by the user ID of the request and the default catalog ID.

You can also use wildcard syntax to select the files.

DELETE-SOURCE = <u>*NO</u> / *YES
Specifies whether the original files/library members are to be deleted after they have been added to the ZIP archive. The default *NO retains the original files/library members.

DELETE-SOURCE = *YES
The original files/library members are deleted. A message is issued if deletion is not possible because of the protection attributes.

i

– If the source file is protected against writing, the file is added in the ZIP container and an error message is sent to the user.

– If the source library element is protected against writing, the library element is added in the zip container and an error message is sent to the user.

– If the source library is protected against writing, the library element is added in the zip container and an error message is sent to the user.

Notes

- Compression figures:

Compression mode	**Size after compression** (280630 PP uncompressed)	**Compression ratio**	**CPU time**
*BEST-SPEED	54096 PP	80,7 %	448,5 sec.
*STD	44496 PP	84,2 %	970,2 sec.
*BEST-COMPRESSION	43488 PP	84,5 %	8229,6 sec.

- If the K2 key is pressed during the ADD-FILE statement, processing is interrupted with the query message `SZP0208`:
  - The user can simply continue processing.
  - The user can terminate processing and return to statement mode (//). The files which had not been added by the time the interruption occurred are not added. If required, they must be added again.
- Using the default TO-FILE=*BY-SOURCE file names are always registered in the ZIP container without catid/userid. To register catid/userid, they have to be specified in the TO-FILE operand.

Specification in ADD-FILE	Registered as
FROM-FILE = MYFILE	MYFILE
FROM-FILE = $UID.MYFILE	MYFILE
FROM-FILE = :CAT1:$UID.MYFILE	MYFILE
FROM-FILE = MYFILE, TO-FILE=$UID.MYFILE	$UID.MYFILE
FROM-FILE = MY*,TO-FILE=NEW-*	NEW-FILE
FROM-FILE = #MYFILE.1	S.163.TSN1.MYFILE.1
FROM-FILE = #MYFILE.1,TO-FILE=MYFILE	MYFILE
FROM-FILE = #MYFILE.*,TO-FILE=MYFILE-*	MYFILE-1

- If data encryption had been set using the MODIFY-ZIP-OPTIONS statement, the files are encrypted when they are added. The standard Zip 2.0 encryption used here is compatible with WinZip on Windows-based systems.

LOGGING = <u>*MINIMUM</u> / *MAXIMUM
Controls the amount of the message output.

LOGGING = <u>*MINIMUM</u>
Only error messages will be sent.

LOGGING =*MAXIMUM
All messages will be sent. Currently the [guaranteed] messages SZP0116 and SZP0117 are sent after resp. file addition and library element addition; further messages may be added in the future..

3.2.2 CONVERT-ZIP-CONTAINER

By default a ZIP container has PAM file format.

Alternatively a ZIP container can also be a binary SAM file (with RECORD-FORMAT=U). This format occurs in the following cases (see also section “Windows interoperability” on page 61):

– Only if the ZIP container was transferred to BS2000 using openFT < V11.0 will it have been stored as a binary SAM file (i.e. with RECORD-FORMAT=U). For processing purposes the ZIP container must be in PAM file format.

– A binary SAM file (with RECORD-FORMAT=U) is required for transfer using openFT < V11.0.

The CONVERT-ZIP-CONTAINER statement converts a ZIP container to the other file format (from PAM to SAM with RECORD-FORMAT=U and vice versa). The ZIP container is retained unchanged, and the converted ZIP container is stored in the output file specified.

```
CONVERT-ZIP-CONTAINER

 FROM-FILE = *CURRENT / <filename 1..54 without-gen-vers> / *LINK(…)
    *LINK(…)
      | LINK-NAME = <structured-name 1..8>
,TO-FILE = *STD / <filename 1..54 without-gen-vers> / *LINK(…)
    *LINK(…)
      | LINK-NAME = <structured-name 1..8>
,WRITE-MODE = *CREATE / *REPLACE-ONLY / *ANY
```

FROM-FILE = *CURRENT / <filename 1..54 without-gen-vers> / *LINK(…)
Specifies the ZIP container which is to be converted.

FROM-FILE = *CURRENT
The container is the one which is presently open.

FROM-FILE = <filename 1..54 without-gen-vers>
Converts the specified ZIP container.

FROM-FILE = *LINK(…)
The ZIP container to be converted is specified by means of a link name.

LINK-NAME = <structured-name 1..8>
Link name which is assigned to the ZIP container.

TO-FILE = *STD / <filename 1..54 without-gen-vers> / *LINK(…)
Specifies the output file in which the converted ZIP file is stored.

TO-FILE = *STD
The output file is assigned a standard name which contains the file name of the ZIP container which is to be converted and a suffix indicating the file format involved (SAM or PAM).
The output file is created under the user-id and default cat-id of the user. An user-id and/or cat-id specified in the FROM-FILE parameter will be ignored for the output file.

The standard name is formed in accordance with the following rules:

– The ZIP container is in PAM file format.

Name of the ZIP container	**Name of the output file**
partialname.PAM	partialname.SAM
partialname.PAM.ZIP	partialname.SAM.ZIP
partialname.ZIP	partialname.SAM.ZIP
filename	filename.SAM

– The ZIP container is in SAM file format with RECORD-FORMAT=U.

Name of the ZIP container	**Name of the output file**
partialname.SAM	partialname.PAM
partialname.SAM.ZIP	partialname.PAM.ZIP
partialname.ZIP	partialname.PAM.ZIP
filename	filename.PAM

In the case of a *partialname*.ZIP or *filename* ZIP container, the path name for the output file can be longer than 54 characters when the suffix is appended. In this case the *partialname* or *filename* in front of the suffix is truncated by the required number of characters.

TO-FILE = <filename 1..54 without-gen-vers>
The converted ZIP container is stored in the specified file.

TO-FILE = *LINK(…)
The output file is specified via a link name.

LINK-NAME = <structured-name 1..8>
Link name which is assigned to the output file.

WRITE-MODE =
Specifies whether the output file should be created or only overwritten.

WRITE-MODE = <u>*CREATE</u>
The output container must be created. The statement is rejected if it already exists.

WRITE-MODE = *REPLACE-ONLY
The output file must exist and is replaced. The statement is rejected if the file does not exist.

WRITE-MODE = *ANY
The output file is overwritten if it exists, otherwise it is created.

3.2.3 DELETE-FILE

This statement deletes one or several files from the ZIP file which is currently open. The statement is rejected when the ZIP container is opened in read mode.

The size of the ZIP container does not change when files contained in it are deleted. Memory space which is no longer required is released only when the ZIP container is reorganized (see the OPEN-ZIP-CONTAINER statement).

DELETE-FILE
FILE-**NAME** = ***ALL** / <composed-name 1..98 with-under with-wild(132)> / <c-string 1..1024 with-low>

FILE-NAME = *ALL
All the files are deleted from the ZIP file.

FILE-NAME = <composed-name 1..98 with-under with-wild(132)>
The specified file is deleted from the ZIP file. When wildcards are used, all files matching the pattern are deleted from the ZIP file.

FILE-NAME = <c-string 1..1024 with-low>
All files which match the specified string (wildcards according to the SDF rules for wildcard selection (see SDF syntax in the “Commands” manual [2]) are permitted) are deleted from the ZIP container. Specification as a C string must be used if the container was created in a non-BS2000 system and the file names concerned do not comply with BS2000 syntax (e.g. upper/lower case).

3.2.4 END

This statement closes the currently opened ZIP file and terminates the program.

END

3.2.5 EXTRACT-FILE

This statement extracts files from the currently opened ZIP file.

```
EXTRACT-FILE

 FILE-NAME = *ALL / <composed-name 1..98 with-under with-wild(132)> / <c-string 1..1024 with-low>
,TO-FILE = *BY-SOURCE / <filename 1..54 without-gen-vers with-wild-constr(80)>
,WRITE-MODE = *CREATE / *REPLACE-ONLY / *ANY
,DATA-TYPE = *NOT-SPECIFIED / *CHARACTER / *BINARY / *SAM-BINARY
,CHARACTER-CONVERSION= *BY-CONTAINER-FORMAT / *NO / *TO-WIN-ANSI / *TO-EBCDIC
,BLOCK-CONTROL-INFO = *KEEP / *IGNORE
,LOGGING = *MINIMUM / *MAXIMUM
```

FILE-NAME = *ALL
All the files included in the container are extracted.

FILE-NAME = <composed-name 1..98 with-under with-wild(132)>
The specified file is extracted from the container. When wildcards are used, all files matching the pattern are extracted from the container.

FILE-NAME = <c-string 1..1024 with-low>
All files which match the specified string (wildcards according to the SDF rules for wildcard selection (see SDF syntax in the “Commands” manual [2]) are permitted) are extracted from the ZIP container. Specification as a C string must be used if the container was created in a non-BS2000 system and the file names concerned do not comply with BS2000 syntax (e.g. upper/lower case).

TO-FILE = *BY-SOURCE / <filename 1..54 without-gen-vers with-wild-constr(80)>
According to the origin of the zipped files, the output name will respect the following rules:

- For BS2000 files, the output name is built according to the SDF rules for wildcard construction (see SDF syntax in the “Commands” manual [2]).

- For other files (PC, Unix system), the output name is built by replacing the '*' character in the TO-FILE operand by the file name registered in the container. The eventual path of the file is ignored in the file name construction process..

TO-FILE=	File origin BS2000	File origin not BS2000
*BY-SOURCE	The output file name will be the file name as registered in the container. – If the file has been registered with a catid/userid, it will be extracted under this catid/userid. – If the file has been registered without catid/userid, it is extracted under the current userid/catid.	The output filename will be the file name without access path prefixed by the current catid/userid. File names which do not comply with the syntax in BS2000 are renamed by BS2ZIP (see below).
<filename 1..54 without-gen-vers with-wild-constr(80)>	Valid format is: – a file name without wildcard. – a file name with wildcards: the output name is built according to the SDF rules for wildcard construction. If the new file name does not respect BS2000 syntax, the extract of the file is rejected.	Valid format is: – a file name without wildcard. – [<PREFIX>]*[<SUFFIX>]: the output file name will be [<PREFIX>]filename[<SUFFIX>] where file name is the registered file name without directory path. If the resultant file name does not comply with the syntax in BS200, it is renamed by BS2ZIP (see below).

If some zipped file names are not BS2000 compliant, it is necessary to extract them one by one, specifying for each of them a valid output name. If extraction results in an invalid file name, the file concerned is assigned the following alternative file name:

FILExxxx.yyyymmdd.hhmmss

where xxxx is a sequence number, yyyymmdd.hhmmss is the current date and time. BS2ZIP shows that the file has been renamed with the message `SZP0090`. Example:

```
%  SZP0090 Warning.  File name 'TEST_KDO_1.HTM' is not BS2000 compliant.
The file will be extracted under the name 'FILE0001.20111108.161442'
```

The catid /userid of those files are taken from the <PREFIX>. If they are not specified the catid/userid of the current user are used by default.

WRITE-MODE = *CREATE
The output file must not exist and is created. The statement is rejected if the file exists already.

WRITE-MODE = *REPLACE-ONLY
The output file must exist and is replaced. The statement is rejected if the file does not exist.

WRITE-MODE = *ANY
The output file is overwritten if exists or created otherwise.

DATA-TYPE =
This operand controls the record structur of the files to be extracted.

DATA-TYPE	**Container format WINZIP compatible**	**Container format BS2000**
*NOT-SPECIFIED	If no file info found in ZIP container then *CHARACTER is assumed. Else file info is used and character conversion is done or not according to file type and CHARACTER-CONVERSION operand	If no file info found Then error Else file info is used - no conversion
*CHARACTER	File is extracted as a SAM file Records are assumed to be delimited by 0D0A Character conversion is performed according to the CHARACTER-CONVERSION operand	Statement rejected
*BINARY	File is extracted as a PAM file No conversion is performed. In case of original PAM files, the extract is rejected with error message SZP0121	Statement rejected
*SAM-BINARY	File is extracted as a binary SAM file (REC-FORM=U) No conversion performed	Statement rejected

CHARACTER-CONVERSION =
This operand controls the WIN-ANSI/EBCDIC conversion. It is only supported when extracting from a WINZIP-COMPATIBLE container and for the following files:

– files extracted with DATA-TYPE=*CHARACTER
– files extracted with DATA-TYPE=*NOT-SPECIFIED and which are not indicated as PAM file in the file-info.

CHARACTER-CONVERSION	Conversion behaviour
<u>*BY-CONTAINER-FORMAT</u>	Default value. Behavior determined by the ZIP container open format
*NO	No conversion is forced
*TO-WIN-ANSI	WIN-ANSI conversion is forced (only for extraction to SAM/ISAM file)
*TO-EBCDIC	EBCDIC conversion is forced (only for extraction to SAM/ISAM file)

BLOCK-CONTROL-INFO =
Controls the block control attribute of the resulting file. This allows particularly to extract original PAMKEY file on a NK disk.

BLOCK-CONTROL-INFO = <u>*KEEP</u>
The resulting file keeps the same block control attribute than the original file

BLOCK-CONTROL-INFO = *IGNORE
The resulting file is created with the default block control of the disk where it is saved.

Notes

- If data encryption had been set using the MODIFY-ZIP-OPTIONS statement, encrypted files are decrypted again when they are extracted. The standard Zip 2.0 encryption used here is compatible with WinZip on Windows-based systems.
- Files extracted from a container created on the BS2000, are created with the same organization characteristics as the original file, except the padding factor and block-control. The padding factor is the default DMS padding value. This implies that the size of extracted SAM and ISAM file can be different from the size of the original files.
- Files extracted from a container created in a foreign environment, are created
  – as SAM files with BUF-LEN=STD(16), provided that DATA-TYPE=*NOT-SPECIFIED or *CHARACTER,
  – as PAM files with BUF-LEN=STD(16) provided that DATA-TYPE=*BINARY and
  – as SAM files with REC-FORM=U provided that DATA-TYPE=*SAM-BINARY.

- K and NK disks
  - When a zipped file with PAMKEY block control has to be extracted on a NK disk, use the operand BLOCK-CONTROL-INFO=*IGNORE. The file will be converted into NK format. However, in case of SAM or ISAM files with records occupying all the available space in blocks, data truncation will occur. In this case, an error will be detected and the extract processing is aborted for the current file. The output file is erased.
  - To extract NK disk files (especially load modules), that have been added to a container, on a PAMKEY disk, the option BLOCK-CONTROL-INFO=*KEEP must be set.
- If the K2 key is pressed during EXTRACT-FILE statement, processing is interrupted with the query message `SZP0208`:
  - The user can simply continue processing.
  - The user can terminate processing and return to statement mode (//). The files which had not been added by the time the interruption occurred are not extracted. If required, they must be extracted again.
- To select files from an archive coming from a foreign platform, take into account that the registered file names are case sensitive. So, use the c-string format to select filenames containing lower cases.
- Rules for naming extracted files:

  BS2000 files:
  (please refer to SDF rules for wildcard construction in the “Commands” manual [2])

FILE-NAME	**TO-FILE**	**Registered file names**	**Resulting file names**[1]
MYFILE1	*BY-SOURCE	MYFILE1	:ccid:$cuid.MYFILE1
*	*BY-SOURCE	MYFILE1 MYFILE2	:ccid:$cuid.MYFILE1 :ccid:$cuid.MYFILE2
MY*	EXT-*	MYFILE1 MYFILE2	:ccid:$cuid.EXT-FILE1 :ccid:$cuid.EXT-FILE2
MYFILE1	*BY-SOURCE	:XXXX:$UID.MYFILE1	No file found
:XXXX:$UID.	*BY-SOURCE	:XXXX:$UID.MYFILE1	:XXXX:$UID.MYFILE1
$UID.	*BY-SOURCE	$UID.MYFILE1	:ccid:$UID.MYFILE1

[1] where $cuid = current userid and :ccid: = catid of the userid

Not BS2000 files:

FILE-NAME	TO-FILE	Registered file names	Resulting file names[1]
MYFILE1	*BY-SOURCE	/temp/data/myfile.txt	No file found
*	*BY-SOURCE	/temp/data/myfile1.txt /temp/data/myfile2.txt	:ccid:$cuid.MYFILE1.TXT :ccid:$cuid.MYFILE2.TXT
'*myfile*'	EXT-*	/temp/data/myfile1.txt /temp/data/myfile2.txt	:ccid:$cuid.EXT-MYFILE1.TXT :ccid:$cuid.EXT-MYFILE2.TXT

[1] where $cuid = current userid and :ccid: = catid of the userid

LOGGING = <u>*MINIMUM</u> / *MAXIMUM
Controls the amount of the message output.

LOGGING = <u>*MINIMUM</u>
Only error messages will be sent.

LOGGING =*MAXIMUM
All messages will be sent. Currently the [guaranteed] message SZP0122 is sent after each file extraction; further messages may be added in the future.

3.2.6 MODIFY-ZIP-OPTIONS

This statement defines the defaults for the current BS2ZIP program run.

You can specify that BS2ZIP files should be encrypted when they are added and decrypted when they are extracted. BS2ZIP creates the keys which are required from the crypto password specified,

```
MODIFY-ZIP-OPTIONS

ENCRYPTION = *UNCHANGED / *NO / *YES(...)

  *YES(...)
   |  CRYPTO-PASSWORD = <c-string 1..100 with-low> / <x-string 1..200> / *SECRET
   |
   | ,CONFIRM-PASSWORD = *NOT-SPECIFIED / <c-string 1..100 with-low> / <x-string 1..200> /
   |                     *SECRET
```

ENCRYPTION = *UNCHANGED / *NO / *YES(…)
Specifies whether or not added files must be encrypted, and extracted files must be decrypted.

ENCRYPTION = *UNCHANGED
The current setting is retained. ENCRYPTION=*NO is set when BS2ZIP is started.

ENCRYPTION = *NO
Specifies that the encryption may not be used. If a password has been specified by a previous statement, the encryption keys will be removed.

ENCRYPTION = *YES(...)
Specifies that the encryption must be used, for adding files and for extracting encrypted files.

CRYPTO-PASSWORD = <c-string 1..100 with-low> / <x-string 1..200> / *SECRET
Specifies the password to be used for encrypting and decrypting data.
The operand has the following special characteristics:
- The password entered is not logged.
- The input field is automatically blanked out in the guided dialog.
- If *SECRET or ^ is specified, in unguided dialog and in foreground procedures SDF provides a non-displaying entry field for concealed entry of the password.

CONFIRM-PASSWORD = *NOT-SPECIFIED / <c-string 1..100 with-low> / <x-string 1..200> / *SECRET
Permits the entry check for the crypto password. Entering the password twice is to avoid a password containing a typing error being assigned when password entry is nondisplaying.
If a value other than the default *NOT-SPECIFIED is entered, then is must be identical with the entry made for CRYPTO-PASSWORD, otherwise the statement is rejected.
The operand has the following special characteristics:
– The password entered is not logged.
– The input field is automatically blanked out in the guided dialog.
– If *SECRET or ^ is specified, in unguided dialog and in foreground procedures SDF provides a non-displaying entry field for concealed entry of the password.

Note

As soon as the password is recognized, it is treated by BS2ZIP in order to store encryption keys in memory. The password itself is deleted from the memory.

3.2.7 OPEN-ZIP-CONTAINER

This statement opens a ZIP container. If a ZIP container had already been opened, it is closed again first.

By default the ZIP container is opened in read mode. If files are to be added to or deleted from it, it must be opened in write mode.

```
OPEN-ZIP-CONTAINER

 CONTAINER = <filename 1..54 without-gen-vers> / *LINK(...)
   *LINK(...)
     |  LINK-NAME = <structured-name 1..8>
,MODE = *READ / *UPDATE(...)
   *UPDATE(...)
     |  STATE = *ANY / *NEW
,FORMAT = *STD / *WINZIP-COMPATIBLE / *BS2000
```

CONTAINER = <filename 1..54 without-gen-vers>
Name of the ZIP file to open.

CONTAINER = *LINK(...)
ZIP file to open is referenced by a link name.

> **LINK-NAME = <structured-name 1..8>**
> Link name associated to the ZIP file to open.

MODE = *READ
The ZIP file is opened in read mode. The file must exists.

MODE = *UPDATE(...)
The ZIP file is opened in write mode.

> **STATE = *ANY**
> If the specified ZIP file does not exists, it is created and opened in update mode, otherwise, it is opened in update mode.
>
> **STATE = *NEW**
> A new ZIP file is created and opened in update mode. If the ZIP file already exists, an error occurs and the statement is rejected.

FORMAT = *STD
If the container does not exist and must be created, i.e. when MODE=*UPDATE, the file is created in BS2000 format. If the container exists i.e. when MODE=*READ or *UPDATE(STATE=*ANY), the file is opened in the format it has been created.

FORMAT = *BS2000
The ZIP container is opened in BS2000 format. ZIP containers in this format can only be used in a BS2000 environment.

FORMAT = *WINZIP-COMPATIBLE
The ZIP container is opened in WinZip-compatible format. ZIP containers in this format can be used in the open world as they are compatible with PKZIP V4.5.

Notes

- If a ZIP container was transferred to BS2000 using openFT < V11.0, it will have SAM file format with RECORD-FORMAT=U. Before this container is opened in BS2ZIP, it must be converted to PAM file format using the CONVERT-ZIP-CONTAINER statement. In the other direction a ZIP container must be converted from PAM to SAM file format with RECORD-FORMAT=U before it can be transferred to another system using openFT < V11.0. See “CONVERT-ZIP-CONTAINER” on page 30.
- An error message will be displayed when the file is not recognized as a valid ZIP file.
- A ZIP container created in WINZIP-COMPATIBLE format cannot be reopened in BS2000 format and a ZIP container created in BS2000 format cannot be reopened in WINZIP-COMPATIBLE format.
- A ZIP container created under Windows or Unix system can only be opened in WinZip compatible mode.
- GZIP archives can only be opened in MODE=*READ. Only display or extraction is thus possible. The container format has not to be specified. Only the default FORMAT=*STD is allowed.
- Opening a container while another one is currently opened leads to the close of the current one. The current one is also closed even if the open of the second one fails.
- The OPEN-ZIP-CONTAINER statement cannot be interrupted by K2.
- Maximum ZIP container extents reached.

  The user can influence the memory space allocation of a ZIP container by means of an appropriate primary and secondary allocation when the file is created using the CREATE-FILE command. If the file does not yet exist, BS2ZIP creates the ZIP container with an initial size of 1920 PAM pages (primary allocation), and a secondary allocation of 192 PAM pages.

  If the container is expected to be large, the user should define a higher secondary allocation. Otherwise there is a danger that because it is constantly being expanded the container will receive the maximum possible number of extents and can then not be expanded any more. A file created using BS2ZIP default values has a maximum size of around 120 MB.

Example of estimation

```
DATAFILE1 100,000pp
DATAFILE2  50,000pp
Total =   150,000pp
```

With a compression ratio of 60%, we can estimate the size of the ZIP container to 60000 PAM pages. You can estimate less than this value. So, BS2ZIP will extend the file if necessary.

Execute the following CREATE-FILE command:

```
/CREATE-FILE MYCONT.ZIP,SUPPORT=*PUBLIC(SPACE=*RELA(60000,100))
/START-ZIP
//OPEN-ZIP-CONTAINER CONTAINER=MYCONT.ZIP,MODE=*UPDATE(STATE=*ANY)
//ADD DATAFILE*
//END
```

In the reverse direction, the initial size of 1920 PAM pages can be excessive, even if it is temporary. To avoid this problem, the user is advised to make an estimation of the resulting size, and specify adequate Space parameters in the CREATE-FILE command. For very small libraries, the necessary primary space allocation can be as small as 6 PAM pages.

- If the available space is lower than 1920 PAM pages for a userid, it is necessary to create the container before opening it with BS2ZIP.
- The size of a container is variable, for the same contents, depending on the manipulations that have been done, including the REORGANIZE-ZIP-CONTAINER statement.

3.2.8 REORGANIZE-ZIP-CONTAINER

This statement reduces the amount of disk space required for a ZIP container. The statement reorganizes a ZIP container in such a way that as much as possible of disk space not used by deleted files is left from the ZIP container.

REORGANIZE-ZIP-CONTAINER
CONTAINER = <u>***STD**</u> / <filename 1..54 without-gen-vers> / ***LINK**(...)
***LINK**(...)
⎮ **LINK**-NAME = <structured-name 1..8>

CONTAINER = <u>*STD</u>
The opened ZIP container is reorganized. If no ZIP container has been opened yet, the statement is rejected.

CONTAINER = <filename 1..54 without-gen-vers>
Name of the ZIP container to be reorganized.

CONTAINER = *LINK(...)
ZIP container to be reorganized is referenced by a link name.

LINK-NAME = <structured-name_1..8>
Link name associated to the ZIP container to be reorganized.

Notes

- During the reorganization of a ZIP container, no other access to this ZIP container is permitted.
- If the ZIP container has already been opened by another task, it cannot be reorganized.

3.2.9 SHOW-FILE-ATTRIBUTES

This statement displays the list of the files included in the currently opened ZIP file.

This statement supports structured output in S variables.

```
SHOW-FILE-ATTRIBUTES

FILE-NAME = *ALL / <composed-name 1..98 with-under with-wild(132)> / <c-string 1..1024 with-low>
,INFORMATION = *SUMMARY / *ALL
,TEXT-OUTPUT = *SYSOUT / *SYSLST(...) / *NONE
   *SYSLST(...)
      | SYSLST-NUMBER = *STD / <integer 1..99>
,STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
   <composed-name 1..255>(...)
      | WRITE-MODE = *REPLACE / *EXTEND
```

FILE-NAME = *ALL
All the files included in the container are listed.

FILE-NAME = <composed-name 1..98 with-under with-wild(132)>
All the files matching the pattern are listed.

FILE-NAME = <c-string 1..1024 with-low>
All files which match the specified string (wildcards according to the SDF rules for wildcard selection (see SDF syntax in the “Commands” manual [2]) are permitted) are extracted from the ZIP container. Wildcards may be specified in the input string.
All the matching files are listed. Specification as a C string must be used if the container was created in a non-BS2000 system and the file names concerned do not comply with BS2000 syntax (e.g. upper/lower case).

INFORMATION = *SUMMARY
Only the name of the archived file plus its origin is displayed. The display ends with the message SZP0087, which shows the total number of files listed. See also “Layout of the output information” on page 49.

INFORMATION = *ALL
Complete information about the archived files is displayed. The display ends with the message SZP0087, which shows the total number of files listed. See also “Layout of the output information” on page 49.

TEXT-OUTPUT = *SYSOUT / *SYSLST(...) /*NONE
Specifies where the information is to be output (output as text string).

TEXT-OUTPUT = *SYSOUT
The information is output to the system file SYSOUT.

TEXT-OUTPUT = *NONE
No information is output except error messages.

TEXT-OUTPUT = *SYSLST(...)
The information is output to the system file SYSLST.

SYSLST-NUMBER = *STD / <integer 1..99>
Specifies whether the information is to be output to the system file SYSLST or to a SYSLST file from the set SYSLST01 through SYSLST99.
The default is *STD, i.e. output is directed to the system file SYSLST.

STRUCTURE-OUTPUT = *SYSINF / *NONE / <composed-name 1..255>(...)
Controls structured output in S variables (see also "Structured output in S variables" on page 51).

STRUCTURE-OUTPUT = *SYSINF
The structured output is directed to the S variable stream SYSINF. The information is accessible if SYSINF was previously assigned to an S variable using the ASSIGN-STREAM command (see "Command output in S variables" in the "Commands" manual [2]).

STRUCTURE-OUTPUT = *NONE
No structured output is to be generated.

STRUCTURE-OUTPUT = <composed-name 1..255>(...)
Name of the variable into which the structured output is to be made. This variable must be declared as a list variable:

```
/DECLARE-VARIABLE VAR-NAME=<var>(TYPE=*STRUCTURE),MULTIPLE-ELEMENTS=*LIST
```

WRITE-MODE = *REPLACE / *EXTEND
Specifies whether the list variable is to be overwritten or extended.
*REPLACE means that the list is overwritten.

Layout of the output information

*Layout with INFORMATION=*SUMMARY*

```
CURRENT CONTAINER : @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@            @@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  + BS2000 : @@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  + BS2000 : @@@
...
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  + BS2000 : @@@
%  SZP0087 'n' file(s) are matching your request
```

The first line displays the name of the currently opened ZIP container and its format: BS2 (= BS2000 format) or WIN (= WinZip compatible format). The file name and the BS2000 output field are displayed in one line for each selected file. Encrypted files are marked with a “+” in front of the BS2000 output field. File names which are longer than 64 characters are displayed in more than one line.

*Layout with INFORMATION=*ALL:*

```
CURRENT CONTAINER : @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@            @@@
--------------------------- FILE INFORMATION ---------------------------------
FILENAME    : @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
              ...
              @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
BS2000      : @@@
MODIFIED    : YYYY-MM-DD HH:MM:SS
SIZE        : NNNNNNNNNNNNN
PACKED      : NNNNNNNNNNNNN
RATIO       : NN.N %
ENCRYPTED   : @@@
------------------------------- COMMENTS -------------------------------------
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
...
%  SZP0087 'n' file(s) are matching your request
```

Fields description

CURRENT CONTAINER
is the name of the currently opened ZIP container and its format:
BS2 for BS2000 format or
WIN for WinZip compatible format.

FILENAME is the name of the file included in the ZIP container.

In case of BS2000 file, this name has the following format:
[:cid:][$uid.]filename according to the TO-FILE operand of the ADD-FILE statement. It is always in uppercase.

		In case of non-BS2000 files, this name has the format with which it has been saved. For example, /dir1/dir2/Myfile.txt. It is case sensitive. Up to 64 characters of the file name are displayed in one line. When file names are longer than 64 characters, the remaining characters are displayed in continuation lines. Up to 1024 characters can be displayed.
BS2000		Indicates the origin of the file.
	YES	means that the file has been included in the container by BS2ZIP program in a BS2000 system
	NO	otherwise.
MODIFIED		is date and time of the last file modification.
SIZE		is the size in bytes of the original file.
PACKED		is the size in bytes of the file after compression.
RATIO		is the compression ratio. It is computed according to the formula: (SIZE – PACKED) * 100 / SIZE. It can occur that the packed size is greater than the original size when no compression is required at ADD-FILE statement. In this case, the RATIO is 0.
ENCRYPTED		Specifies whether the file was encrypted when it was added to the container.
	YES	displays an encrypted file.
	NO	displays a not encrypted file.
COMMENTS		is the comments associated to the file. In the case of BS2000 files, the comments begin with a `*BS2:` string and then list the DMS properties of the original file as operands of a FILE macro (see the “DMS Macros” manual [4]).

Note

The container format is also available in the catalog entry of the container in the field USER-INFO (Organization section):

`USER-INFO = BS2ZIP-B` for BS2000 format

`USER-INFO = BS2ZIP-W` for WinZip compatible

Structured output in S variables

The INFORMATION=*SUMMARY operand supplies only the S variables for the file name and the file origin (`FILENAME` and `BS2000`) with values. INFORMATION=*ALL causes all the S variables to be supplied with values.

Output information	Name of the S variable	T	Contents
Alternate index (ISAM file)	var(*LIST).ALTERNATE-INDEX	S	” *YES
Block control information	var(*LIST).BLKCTRL	S	NO PAMKEY DATA2K DATA4K DATA
Buffer size	var(*LIST).BLKSIZE	S	” <integer> STD(n)
Origin of the file	var(*LIST).BS2000	S	*YES *NO
Name code table	var(*LIST).CODED-CHAR-SET	S	” <ccs>
File encryption	var(*LIST).ENCRYPTED		*YES *NO
File structure (access method used to create the file)	var(*LIST).FCBTYPE	S	SAM ISAM PAM
File name; in the case of non-BS2000 files possibly as a path name	var(*LIST).FILENAME	S	<filename> <pathname 1..1024>
Indicator for the file type PLAM library	var(*LIST).FILETYPE	S	” PLAM-LIB
Performance requirements for I/O operations	var(*LIST).IOPERF	S	” HIGH STD VERY-HIGH
Operation(s) affected by the I/O performance requirements	var(*LIST).IOUSAGE	S	” READ RDWRT WRITE
Length of the ISAM key	var(*LIST).KEYLEN	I	<integer>
Position of the ISAM key in the record	var(*LIST).KEYPOS	I	<integer>
Length of the logical flag in the ISAM index	var(*LIST).LOGLEN	I	<integer>
Date of last file modification	var(*LIST).MODIFIED-DATE	S	<yyyy-mm-dd>

Output information	Name of the S variable	T	Contents
Time of last file modification	var(*LIST).MODIFIED-TIME	S	<hh:mm:ss>
Size of compressed file (byte)	var(*LIST).PACKED	I	<integer>
Compression ratio	var(*LIST).RATIO	S	nn.n
File record format	var(*LIST).RECFORM	S	F U V
File record length	var(*LIST).RECSIZE	I	<integer>
Secondary allocation for file extensions	var(*LIST).SEC-ALLOC	I	<integer>
Original file size (byte)	var(*LIST).SIZE	I	<integer>
Length of the value flag in the ISAM index	var(*LIST).VALLEN	I	<integer>
Treatment of the value flag within a data or index block (for K-ISAM files)	var(*LIST).VALPROP	S	” MAX MIN

Example

```
/START-ZIP-MANAGER
//OPEN-ZIP-CONTAINER MYCONT.ZIP
//SHOW-FILE-ATTRIBUTES
CURRENT CONTAINER : MYCONT.ZIP
$DIAGDUMP.A0478578.SLED.S210.VM2.110611                          BS2000 : YES
%  SZP0087 '1' file(s) are matching your request
//SHOW-FILE-ATTRIBUTES *ALL,INFORMATION=ALL
CURRENT CONTAINER : MYCONT.ZIP
--------------------------- FILE INFORMATION ---------------------------------
FILENAME    : $DIAGDUMP.A0478578.SLED.S210.VM2.110611
BS2000      : YES
MODIFIED    : 2011-06-11 18:49:32
SIZE        : 227454976
PACKED      : 35012077
RATIO       : 84.6 %
ENCRYPTED   : NO
------------------------------- COMMENTS ------------------------------------
*BS2: ,FCBTYPE=PAM,BLKSIZE=(STD,1),CODED-CHAR-SET=EDF03IRV,BLKCTRL=NO
%  SZP0087 '1' file(s) are matching your request
```

3.2.10 START-TRACE

This statement activates an internal trace.

START-TRACE
TRACE-FILE = ***STD** / <filename 1..54 without-gen-vers>

TRACE-FILE = *STD
The trace file name by default is SYSTRC.BS2ZIP.<yyyy-mm-dd>.<hhmmss> and is located under the userid that executes the BS2ZIP program.

TRACE-FILE = <filename 1..54 without-gen-vers>
A trace file with the user specified file name is created. If the file already exists, it is overwritten.

3.2.11 STOP-TRACE

This statement deactivates the internal trace. The trace file is closed.

STOP-TRACE

3.3 Examples

BS2000 example

```
/show-file-attr test.
%        3 :1OSN:$BS2ZIP.TEST.ISAM-STD1-PAMKEY-VN0-EDF03IRV-8-5
%       18 :1OSN:$BS2ZIP.TEST.ISAM2
%        6 :1OSN:$BS2ZIP.TEST.ISO88591
%        3 :1OSN:$BS2ZIP.TEST.NKEY
%       12 :1OSN:$BS2ZIP.TEST.NKFILE
%       12 :1OSN:$BS2ZIP.TEST.PAM
%       51 :1OSN:$BS2ZIP.TEST.PAM.16VP-32768
%       36 :1OSN:$BS2ZIP.TEST.PAM16-MAX.2
%       21 :1OSN:$BS2ZIP.TEST.PAM2
%       48 :1OSN:$BS2ZIP.TEST.RECFORMU
%      192 :1OSN:$BS2ZIP.TEST.SAM.16VP-32768
%       18 :1OSN:$BS2ZIP.TEST.SAM1-MAX
/start-zip-manager
%  SZPLOAD Program 'BS2ZIP',Version 'V01.2G15' of '2014-09-15' loaded
%  SZPCOPY Copyright (C) 2014 Fujitsu Technology Solutions GmbH All Rights
Reserved
//open-zip-container container=mycont,mode=*update(state=*new)
//add-file test.
//show-file-attr
CURRENT CONTAINER : MYCONT
TEST.ISAM-STD1-PAMKEY-VN0-EDF03IRV-8-5                          BS2000 : YES
TEST.ISAM2                                                      BS2000 : YES
TEST.ISO88591                                                   BS2000 : YES
TEST.NKEY                                                       BS2000 : YES
TEST.NKFILE                                                     BS2000 : YES
TEST.PAM                                                        BS2000 : YES
TEST.PAM.16VP-32768                                             BS2000 : YES
TEST.PAM16-MAX.2                                                BS2000 : YES
TEST.PAM2                                                       BS2000 : YES
TEST.RECFORMU                                                   BS2000 : YES
TEST.SAM.16VP-32768                                             BS2000 : YES
TEST.SAM1-MAX                                                   BS2000 : YES
%  SZP0087 '12' file(s) are matching your request
```

```
//show-file-attr file-name=test.isam-std.*,inf=*all
CURRENT CONTAINER : MYCONT
 --------------------------- FILE INFORMATION -------------------------------
 FILENAME   : TEST.ISAM-STD1-PAMKEY-VN0-EDF03IRV-8-5
 BS2000     : YES
 MODIFIED   : 2014-11-25 10:21:22
 SIZE       : 231
 PACKED     : 56
 RATIO      : 75.8 %
 ENCRYPTED  : NO
 ------------------------------- COMMENTS -----------------------------------
 *BS2: ,FCBTYPE=ISAM,BLKSIZE=(STD,1),KEYPOS=5,KEYLEN=8,RECFORM=(V,N),
IOPERF=STD,IOUSAGE=RDWRT,CODED-CHAR-SET=EDF03IRV,BLKCTRL=PAMKEY
%  SZP0087 '1' file(s) are matching your request
//extract-files file-name=test.pam.*,to-file=ext-test.pam.*
//end
/show-file-attr ext-*
%        51 :1OSN:$BS2ZIP.EXT-TEST.PAM.16VP-32768
%        33 :1OSN:$BS2ZIP.EXT-TEST.PAM16-MAX.2
%        21 :1OSN:$BS2ZIP.EXT-TEST.PAM2
```

WinZip example

```
/start-ftp
.
.
.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
05-15-14 03:19PM              978691 server_1.zip
226 Transfer complete.
53 bytes received in 0.08 seconds (0.62 Kbytes/s)
ftp> bin
200 Type set to I.
ftp> get server_1.zip SERVER-1.PC.ZIP
200 PORT command successful.
150 Opening BINARY mode data connection for server_1.zip(978691 bytes).
226 Transfer complete.
978691 bytes received in 6.90 seconds (1.4e+02 Kbytes/s)
ftp> bye
221
%  CCM0998 CPU TIME USED: 0.0925 SECONDS
/show-file-attr server-1.pc.zip,inf=all
%0000000480 :2OSG:$USER1.SERVER-1.PC.ZIP
%  ------------------------------- HISTORY      ------------------------------
%  CRE-DATE   = 2014-05-15  ACC-DATE   = 2014-05-15  CHANG-DATE = 2014-05-15
%  CRE-TIME   =   15:46:32  ACC-TIME   =   15:47:45  CHANG-TIME =   15:46:39
%  ACC-COUNT  = 3           S-ALLO-NUM = 0
%  ------------------------------- SECURITY     ------------------------------
%  READ-PASS  = NONE        WRITE-PASS = NONE        EXEC-PASS  = NONE
%  USER-ACC   = OWNER-ONLY  ACCESS     = WRITE       ACL        = NO
%  AUDIT      = NONE        FREE-DEL-D = *NONE       EXPIR-DATE = 2014-05-15
%  DESTROY    = NO          FREE-DEL-T = *NONE       EXPIR-TIME =   00:00:00
%  SP-REL-LOCK= NO          ENCRYPTION = *NONE
%  ------------------------------- BACKUP       ------------------------------
%  BACK-CLASS = A           SAVED-PAG  = COMPL-FILE  VERSION    = 1
%  MIGRATE    = ALLOWED
%  ------------------------------- ORGANIZATION ------------------------------
%  FILE-STRUC = PAM         BUF-LEN    = STD(1)      BLK-CONTR  = PAMKEY
%  IO(USAGE)  = READ-WRITE  IO(PERF)   = STD         DISK-WRITE = IMMEDIATE
%  AVAIL      = *STD
%  WORK-FILE  = *NO         F-PREFORM  = *K          SO-MIGR    = *ALLOWED
%  ------------------------------- ALLOCATION   ------------------------------
%  SUPPORT    = PUB         S-ALLOC    = 16          HIGH-US-PA = 478
%  EXTENTS     VOLUME     DEVICE-TYPE      EXTENTS     VOLUME     DEVICE-TYPE
%      1       GVS2.5       D3435
%  NUM-OF-EXT = 1
%:2OSG: PUBLIC:      1 FILE  RES=       480 FRE=        2 REL=        0 PAGES
```

```
/start-zip
%  SZPLOAD Program 'BS2ZIP',Version 'V01.2G15' of '2014-09-15' loaded
%  SZPCOPY Copyright (C) 2014 Fujitsu Technology Solutions GmbH All Rights
Reserved
//open-zip-container server-1.pc.zip
//show-file-attr '*\/<ADD,Glos,Ind>*.htm'
CURRENT CONTAINER : SERVER-1.PC.ZIP
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_1/ADD_CJC-ACTION.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_1/ADD_FILE_LINK__TFT_Eintr.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_2/ADD_CJC_ACTION.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_2/ADD_FILE_LINK.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/Glossary.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Skin/  BS2000 : NO
Index.htm
%  SZP0087 '6' file(s) are matching your request
//show-file-attr *ADD/CJC*
CURRENT CONTAINER : SERVER-1.PC.ZIP
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_1/ADD_CJC-ACTION.htm
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/test_kdo_2/ADD_CJC_ACTION.htm
%  SZP0087 '2' file(s) are matching your request
//show-file-attr '*.css'
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/Resources/Stylesheets/test_kdo_1.css
server_1/osd_allg/projekte/test_kdo_1/Output/new/MyWebHelp/Conte  BS2000 : NO
nt/SkinSupport/MadCap.css
%  SZP0087 '2' file(s) are matching your request
//extract-file file-name=*ADD/CJC*,to-file=webhelp.*.copy
%  SZP0090 Warning.  File name 'WEBHELP.ADD_CJC-ACTION.HTM.COPY' is not
BS2000 compliant.  The file will be extracted under the name
'FILE0001.20140515.155645'
%  SZP0090 Warning.  File name 'WEBHELP.ADD_CJC_ACTION.HTM.COPY' is not
BS2000 compliant.  The file will be extracted under the name
'FILE0002.20140515.155645'
//extract-file file-name='*.css',to-file=webhelp.*.copy
%  SZP0090 Warning.  File name 'WEBHELP.TEST_KDO_1.CSS.COPY' is not BS2000
compliant.  The file will be extracted under the name
'FILE0001.20140515.155759'
//end
```

```
/show-file-attr <file,webhelp>*
%       33 :2OSG:$USER1.FILE0001.20140515.155645
%       48 :2OSG:$USER1.FILE0001.20140515.155759
%       33 :2OSG:$USER1.FILE0002.20140515.155645
%       33 :2OSG:$USER1.WEBHELP.MADCAP.CSS.COPY
%:2OSG: PUBLIC:      4 FILES RES=       147 FRE=       19 REL=        0 PAGES
```

4 Interoperability

4.1 Windows interoperability

Windows programs supporting the ZIP format (WinZip) are able to read BS2ZIP containers provided those BS2000 containers have been created in WinZip compatible mode.

BS2ZIP is able to read Windows ZIP archives provided they are opened and BS2ZIP is able to read them in compatible mode.

The transfer of such containers must be performed in binary format. The following shows an example of transfer.

Under DOS command prompt:

```
D:\>ftp
ftp>open 999.999.999.999
connected to 999.999.999.999
user : bs2zip
password : *******
account : xxxx
ftp>binary
ftp>quote ftyp binary
ftp>get bs2000.zip
ftp>put pc.zip
ftp>bye
```

`D:\bs2000.zip` can be opened by WinZip and `$BS2ZIP.PC.ZIP` can be opened by BS2ZIP.

i The length of the SAM and ISAM file records is limited by the buffer length of the file (BUFFER-LENGTH). This does not apply for records of Windows text files. To minimize the risk of too long records, BS2ZIP always specify a buffer length *STD(16) which corresponds to 32 KB - 8 Byte. This must be taken into account by the applications reading such files.

If longer records are found, they are split in as many records as necessary.

Transferring ZIP files with openFT

When transferring ZIP files from/to Windows with openFT< V11.0, the ZIP files must be converted into the correct file format, either before transfer (transfer to Windows) or after transfer (transfer from Windows).

Indeed,

1. BS2ZIP only manages PAM files withBUFFER-LENGTH=STD(x)
2. PAM files may not be transferred to Windows with openFT< V11.0
3. the files must be transferred in binary mode (mandatory for zipped files)
4. the resulting file transferred (from Windows) with openFT< V11.0 is a SAM file with BUFFER-LENGTH=STD(x) and RECORD-FORMAT=U

SAM/PAM conversion

You can execute conversion in BS2ZIP using the CONVERT-ZIP-CONTAINER statement. See the description of “CONVERT-ZIP-CONTAINER” on page 30.

For compatibility reasons the converter can also be called using the /START-SAM-PAM-CONVERTER command:

You are prompted to specify the input file (first parameter) and the output file (second parameter). The converter converts the input file as follows:

– a PAM file with BUF-LEN=STD(x) into a SAM file with BUF-LEN=STD(8)
– a SAM file with BUF-LEN=STD(x) into a PAM file with BUF-LEN=STD(16)

4.2 Unix system interoperability

GZIP or GZ Unix system tools generate GZIP archives. Those archives must be transferred in binary to the BS2000 and then can be read by BS2ZIP. Those archives cannot be modified with BS2ZIP.

By default, BS2ZIP extracts the archived file as a SAM file with BUF-LEN=STD(16). It is assumed that the line separator is a Line Feed (LF). Files can be also be extracted in binary PAM files with BUF-LEN=STD(16) are thus created.

Only gz files are supported; tar.gz files are not supported.

4.3 Linux interoperability

Archives created by ZIP Linux tool are readable by BS2ZIP. Those archives must be transferred in binary to the BS2000. By default, BS2ZIP extracts the archived file as a SAM file with BUF-LEN=STD(16). It is assumed that the line separator is a Carriage Return / Line Feed (CRLF). Files can be also be extracted in binary. PAM files with BUF-LEN=STD(16) are thus created.

On the other hand, BS2ZIP containers built in compatible mode can be transferred in binary to a Linux platform and then be exploited by the UNZIP Linux tool.

5 Application Program Interface

BS2ZIP can be used as sub-program in TU applications. To that purpose, an API and a run time module are provided in the SYSLNK.BS2ZIP.<version>.RTE library:

- The C++ header file SZPZIP.H: API of BS2ZIP
- The C++ header file SZPZOUT.H: layout of the output buffers for the *ListFiles* function
- The run time module BS2ZIPPR

5.1 SZPZIP.H

```
#ifndef __SzpZip_h__
#define __SzpZip_h__
/*******************************************************************
 * BEGIN-INTERFACE  SZPZIP.H
 *
 *  TITLE   (/ zip program interface  /)
 *  Classes        CSzpZip
 *  NAME           szpzip.h
 *  DOMAIN         BS2ZIP
 *  LANGUAGE       CXX
 *  Copyright      FUJITSU TECHNOLOGY SOLUTIONS 2014
 *                 All rights reserved
 *  COMPILATION-SCOPE USER
 *  INTERFACE-TYPE    CALL
 *  RUN-CONTEXT       TU
 *
 *  VERSION        133
 *  CRDATE         2014-04-08
 *  AUTHOR         (/ J. Beaume, OSL41 /)
 *  UPDATE         (/ BS2ZIP v1.2G10 /)
 *
 * END-INTERFACE
 *******************************************************************/
/*
 *  Version        132
 *  Date           2013-02-01
 *  Author         P. Louis - OSL EPS
 *  Update         BS2ZIP v1.2E05
 *
 *  Version        130
 *  Date           2011-05-18
 *  Author         J. Beaume - OSL EPS
 *  Update         BS2ZIP v1.2D05
 *
 *  Version        129
 *  Date           2009-09-30
 *  Author         P. Louis - OSL EPS
 *  Update         BS2ZIP v1.2C05
 *
 *  Version        128
 *  Date           2008-11-27
 *  Author         P. Louis - OSL EPS
 *  Update         BS2ZIP v1.2B05
 *                 Introduction CpuExhauster (A0571814)
 *
 *  Version        126
 *  Date           2008-10-14
 *  Author         Ph/Dumont- OSL EPS
 *  Update         BS2ZIP v1.2A25
 *                 Introduction CpuExhauster (A0570787)
 *
 *  Version        121
 *  Date           2008-04-18
 *  Author         P.Louis  - OSL EPS
 *  Update         BS2ZIP v1.2A10
 *
 *  Version        118
 *  Date           2007-12-01
 *  Author         P.Louis  - OSL EPS
 *  Update         BS2ZIP v1.2A
 *
 *  Version        116
 *  Date           2006-12-01
 *  Author         Ph.Dumont  - OSL EPS
 *  Update         BS2ZIP v1.1C
 *
 *  Version        115
 *  Date           2005-06-23
 *  Author         L. Tambour - OSL EPS
 *  Update         BS2ZIP v1.1B
 *
 *  Version        110
 *  Date           2004-09-30
 *  Author         L. Tambour - OSL EPS
 *  Update         BS2ZIP v1.1A
 *
 *  Version        100
 *  Date           2003-06-02
 *  Author         J. Beaume - OSL EPS
 *  Update         BS2ZIP v1.0B
 *********************************************************************/
#include "szpzout.h"
// SzpZip return codes:
#define CSZPZIP_LIB_EXCEPT_ERROR      114    //133
#define CSZPZIP_FILE_EXCEPT_ERROR     113    //133
#define CSZPZIP_LIB_SELECTION_ERROR   112    //133
#define CSZPZIP_FILE_SELECTION_ERROR  111    //133
#define CSZPZIP_DELETE_ORILIBEL_ERROR 110
#define CSZPZIP_DELETE_ORIGFILE_DMS   109
#define CSZPZIP_DELETE_ORIGFILE_ERROR 108
#define CSZPZIP_LMS_ERROR             101
#define CSZPZIP_PARAMETER_ERROR       100
#define CSZPZIP_INCOHERENT_FORMAT      99
#define CSZPZIP_OPEN_ERROR             98
#define CSZPZIP_CONTAINER_EXISTS       97
#define CSZPZIP_NO_CONTAINER_EXISTS    96
#define CSZPZIP_LINKNAME               95
#define CSZPZIP_NO_FILE                94
#define CSZPZIP_FILE_EXISTS            93
#define CSZPZIP_NO_FILE_EXISTS         92
#define CSZPZIP_INTERNAL_ERROR         91
#define CSZPZIP_RSV1                   90
#define CSZPZIP_EXTRACT_ERROR          89
#define CSZPZIP_CATALOG                88
#define CSZPZIP_NO_CONTAINER_OPENED    86
#define CSZPZIP_READMODE               85
#define CSZPZIP_ALREADY_ZIPPED         84
#define CSZPZIP_ALREADY_ZIPPED_WLDC    83
#define CSZPZIP_ADD_ERROR              82
#define CSZPZIP_ILLEGAL_NEW_NAME       81
#define CSZPZIP_ILLEGAL_LINK           80
#define CSZPZIP_INVALID_RENAMING       79
#define CSZPZIP_IMPOSSIBLE_RENAMING    39
#define CSZPZIP_REORG_ERROR            78
#define CSZPZIP_DELETE_ERROR           77
#define CSZPZIP_INCON_FORMAT_ERROR     76
#define CSZPZIP_OUCON_DMS_ERROR        75
#define CSZPZIP_INCON_DMS_ERROR        74
#define CSZPZIP_OUCON_EXIST_ERROR      73
#define CSZPZIP_OUCON_NOEXIST_ERROR    72
#define CSZPZIP_INCON_NOEXIST_ERROR    71
#define CSZPZIP_OUCON_LINK_ERROR       70
```

```
#define CSZPZIP_INCON_LINK_ERROR       69
#define CSZPZIP_OUCON_FLINK_ERROR      68
#define CSZPZIP_INCON_FLINK_ERROR      67
#define CSZPZIP_INCON_OPEN_ERROR       66
#define CSZPZIP_PSWORDCHECK_ERROR      42
#define CSZPZIP_PSWORDNOTCOR_ERROR     41
#define CSZPZIP_PSWORDNOTGIV_ERROR     40
#define CSZPZIP_DATATYPE_BS2000        37
#define CSZPZIP_DMS_ERROR              31
#define CSZPZIP_INT_ERROR              19
#define CSZPZIP_STD_NAME                2
#define CSZPZIP_SHORT_STD_NAME          1
// Message id
#define MSG_K2                     "SZP0207"
#define MSG_LIB_EXCEPT_ERROR       "SZP0114"  //133
#define MSG_FILE_EXCEPT_ERROR      "SZP0113" //133
#define MSG_LIB_SELECTION_ERROR "SZP0112"  //133
#define MSG_FILE_SELECTION_ERROR "SZP0111" //133
#define MSG_DELETE_ORILEL_FILE  "SZP0110"  //133
#define MSG_DELETE_ORIDMS_FILE  "SZP0109"
#define MSG_DELETE_ORIGIN_FILE  "SZP0108"
#define MSG_LMS_ERROR              "SZP0101"
#define MSG_PARAMETER_ERROR        "SZP0100"
#define MSG_INCOHERENT_FORMAT      "SZP0099"
#define MSG_OPEN_ERROR             "SZP0098"
#define MSG_CONTAINER_EXISTS       "SZP0097"
#define MSG_NO_CONTAINER_EXISTS "SZP0096"
#define MSG_LINKNAME               "SZP0095"
#define MSG_NO_FILE                "SZP0094"
#define MSG_FILE_EXISTS            "SZP0093"
#define MSG_NO_FILE_EXISTS         "SZP0092"
#define MSG_INTERNAL_ERROR         "SZP0091"
#define MSG_WARNING1               "SZP0090"
#define MSG_EXTRACT_ERROR          "SZP0089"
#define MSG_CATALOG                "SZP0088"
#define MSG_NO_CONTAINER_OPENED "SZP0086"
#define MSG_READMODE               "SZP0085"
#define MSG_ALREADY_ZIPPED         "SZP0084"
#define MSG_ALREADY_ZIPPED_WLDC "SZP0083"
#define MSG_ADD_ERROR              "SZP0082"
#define MSG_ILLEGAL_NEW_NAME       "SZP0081"
#define MSG_ILLEGAL_LINK           "SZP0080"
#define MSG_INVALID_RENAMING       "SZP0079"
#define MSG_IMPOSSIBLE_RENAMING "SZP0039"
#define MSG_REORG_ERROR            "SZP0078"
#define MSG_DELETE_ERROR           "SZP0077"
#define MSG_INCON_FORMAT_ERROR  "SZP0076"
#define MSG_OUCON_DMS_ERROR        "SZP0075"
```

```
#define MSG_INCON_DMS_ERROR      "SZP0074"
#define MSG_OUCON_EXIST_ERROR    "SZP0073"
#define MSG_OUCON_NOEXIST_ERROR "SZP0072"
#define MSG_INCON_NOEXIST_ERROR "SZP0071"
#define MSG_OUCON_LINK_ERROR     "SZP0070"
#define MSG_INCON_LINK_ERROR     "SZP0069"
#define MSG_OUCON_FLINK_ERROR    "SZP0068"
#define MSG_INCON_FLINK_ERROR    "SZP0067"
#define MSG_INCON_OPEN_ERROR     "SZP0066"
#define MSG_PSWORDCHECK_ERROR    "SZP0042"
#define MSG_PSWORDNOTCOR_ERROR   "SZP0041"
#define MSG_PSWORDNOTGIV_ERROR   "SZP0040"
#define MSG_DATATYPE_BS2000      "SZP0037"
#define MSG_DMS_ERROR            "SZP0031"
#define MSG_INT_ERROR            "SZP0019"
/**
This class is the C++ api for managing a BS2ZIP container file.
This class is able to handle the K2 but not to intercept it.
**/
class CSzpZip {
public:
   /**
   **/
   CSzpZip();
   /**
   **/
   CSzpZip(bool bOutputMsg);
   /**
   **/
   virtual ~CSzpZip();
   enum szpOpenMode {
      read,
      updateAny,
      updateNew
   };
   enum szpFormat {
      defaut,
      compatible,
      bs2000
   };
   enum szpConvertMode {
      std,
      none,
      ascii,
      ebcdic
   };
   enum szpWriteMode {
      create,
      replace,
      any
   };
   enum szpLink {
      no,
      yes
   };
   enum szpDataType {
      notSpecified,
      character,
      binary,
      sambinary
   };
   enum szpBlkCtrlInfo {
      keep,
      ignore
   };
   enum szpLevel {
      noCompression = 0,
      bestSpeed = 1,
      bestCompression = 9,
      defaultCompression = -1
   };
   enum szpInfo  {
      infoNone,
      infoAll,
      infoSummary
   };
// 132
   enum szpDelete {
      DeleteOption_No,
      DeleteOption_Yes
   };
   enum szpTofileOption {
       TofileOptionVersStd_TypeStd = 0,
       TofileOptionVersStd_TypeYes = 1,
       TofileOptionVersStd_TypeNo = 2,
       TofileOptionVersYes_TypeStd = 10,
       TofileOptionVersYes_TypeYes = 11,
       TofileOptionVersYes_TypeNo = 12,
       TofileOptionVersNo_TypeStd = 20,
       TofileOptionVersNo_TypeYes = 21,
       TofileOptionVersNo_TypeNo = 22
   };
   /**
   Open a zip container.
   It returns 0 if ok.
   char *pContainer = name of the zip container file
```

```
this file name must be valid
if "link=linkname", the container is opened
using this link name
int iOpenMode    = read, update, create
int iFormat      = bs2000 or compatible
**/
int OpenZip(char *pContainer, int iOpenMode, int iFormat);
/**
Close the zip container and release eventually file link.
It returns 0 if ok.
**/
int CloseZip();
/**
Add files in the zip container.  The filename may contain wildcards.
By this way several files can be zipped in a single operation.  Eventual
data conversion is required as well as the compression level.
Files are compressed one by one.  If an error occurred, the data already
written in the container remain in it but the entry is not visible.  If
several files have to be written, the processing goes on even in case of
error on a previous file.  The same behavior is implemented if K2 is
   handled
during the processing of a file but if there are still files to be
   included,
those ones are not processed.
To keep the container coherent, the central header is rewritten after each
added files and then reread to rebuild the structure in memory.
It returns 0 if ok.
V01.0B: introduction of the parameter data-type
This operand is only significant for adding sam files in compatible
   format,
otherwise, it is ignored. In char mode, each record of a sam file is
interpreted as a line.  In binary mode, data are interpreted as a simple
byte stream.  Such a file is extracted as a pam std 16.
char *pFileNames = file name (incl. wild cards ev.)
int iConvertMode = convert mode
int iLevel       = compression level
int iDataType    = data type (not specified, character, binary)
char *pToFiles   = target construction file name
int iDeleteOption = source file delete option                        132
**/
int AddFiles(char *pFileNames, int iConvertMode, int iLevel, int
   iDataType, char *pToFiles, int iDeleteOption);
/**
Add files in the zip container.  Old format for program compatibility.
**/
int AddFiles(char *pFileNames, int iConvertMode, int iLevel, int
   iDataType, char *pToFiles);
/**
```

```
Add PLAM elements in the zip container.  The element names, version, base
    and type
may contain wildcards.  Library name may not.
By this way several elements can be zipped in a single operation.
    Eventual
data conversion is required as well as the compression level.
Elements are compressed one by one.  If an error occurred, the data
    already
written in the container remain in it but the entry is not visible.  If
several elements have to be written, the processing goes on even in case
    of
error on a previous file.  The same behavior is implemented if K2 is
    handled
during the processing of a file but if there are still files to be
    included,
those ones are not processed.
To keep the container coherent, the central header is rewritten after each
added elements and then reread to rebuild the structure in memory.
It returns 0 if ok.
V01.0B: introduction of the parameter data-type
This operand is only significant for adding sam files in compatible
    format,
otherwise, it is ignored. In char mode, each record of a sam file is
interpreted as a line.  In binary mode, data are interpreted as a simple
byte stream.  Such a file is extracted as a pam std 16.
char *pLibName = library name (excl. wild cards)
char *pElements = element name selector (incl. wild cards ev.)
char *pVersion = element version selector (incl. wild cards ev.)
char *pType = element type selector (incl. wild cards ev.)
char *pBase = version base selector
int iConvertMode = convert mode
int iLevel       = compression level
int iDataType    = data type (not specified, character, binary)
char *pToFiles   = target construction file name
int iTofileOption = specifies if libr element version and/or type are
    saved in the name
int iDeleteOption = library element delete option                    132
**/
int AddLibElements(char *pLibName, char *pElements, char *pVersion,
        char *pType,char *pBase, int iConvertMode, int iLevel, int
    iDataType,
        char *pToFiles, int iTofileOption, int iDeleteOption);
/**
Add PLAM elements in the zip container.  Old format for program
    compatibility.
**/
int AddLibElements(char *pLibName, char *pElements, char *pVersion,
        char *pType,char *pBase, int iConvertMode, int iLevel, int
    iDataType,
        char *pToFiles, int iTofileOption);
/**
Extract one or several files selected by the pFileNames patterns into
BS2000 files named according the rule specified by the pToFiles pattern.
If an error occurred during the extract of a file, the extract of this
file is interrupted but the other files are processed.
If K2 is handled, the current file process is stopped, the output file
is deleted and the other extract files are not processed if any.
char *pFileNames = file pattern used to select the files that must
be extracted from the container
char *pToFiles   = pattern specifying the output name of the extracted
    files
int iWriteMode   = new (default) or replace
int iDataType    = data type of the file (not specified (default), char or
binary)
int iConvertMode = convert mode (std (default), none, ascii or ebcdic)
int iBlkCtrlInfo = specifies if the original block control must be used
or not
**/
int ExtractFiles(char *pFileNames, char *pToFiles,
int iWriteMode = 0, int iDataType = 0, int iConvertMode = 0,
int iBlkCtrlInfo = 0);
/**
Returns the information about the zipped files selected by the pFileNames
pattern that may contain wild cards.  Information summary or full is
    possible.
The layout of the output is described in the header file szpzout.h.
Data are returned in a buffer requested by this method.  Its address and
    size
are returned to the caller who is in charge to release it after use.
The number of matching files is returned.  -1 is returned in case of
    error.
char *pFileNames = file selection pattern
char **pBuf      = address where the output buffer address is returned
int *iSize       = address of an int where the output buffer size is
    returned
int iInfo        = information type selection (infoAll default,
    infoSummary)
**/
int ListFiles(char *pFileNames, char **pBuf, int *iSize = 0, int iInfo =
    infoAll, bool ListFirst = true);
  /**
Delete files from a zip container.  The filename may contain wildcards.
By this way several files can be deleted in a single operation.
If several files have to be deleted, the processing goes on even in case
    of
```

```
error on a previous file.  The same behavior is implemented if K2 is
   handled
during the processing of a file but if there are still files to be
   deleted,
those ones are not processed.

It returns 0 if ok.
char *pFileNames = file name (incl. wild cards ev.)
**/
int DeleteFiles(char *pFileNames);
/**
CONVERT  ********************************************

It returns 0 if ok.
char *pFromContainer = Output file name
char *pToContainer = Input file name
int iHuser = Home userid length
int iHcat  = Home catid length
int iWriteMode   = new (default) or replace
**/
int CnvZip(char *pFromContainer, char *pToContainer, int iHuser, int
   iHcat, int iWriteMode = 0);

/**
MODIFY-ZIP-OPTIONS  ****************************************
It returns 0 if ok.
char *pCrypto = crypto password
int iLen      = length of crypto password
int iEncrypt  = encryption or not
**/
int ModZipopt(char *pCrypto, int iLen, int iEncrypt);

/**
// reorganize the file by rewriting only the files having a header in the
   central directory
It returns 0 if ok.
char *pContainer = name of the zip container file
this file name must be valid
if "link=linkname", the container is opened
using this link name
**/
int ReorganizeZip(char *pContainer);
/**
Activate the trace processing.
It returns 0 if ok.
char *pFileName = name of the trace file.  If NULL, default name is used
**/
int ActivateTrace(char *pFileName);
```

```
      /**
      Deactivate the trace
      **/
      int DeactivateTrace();
      /**
      Return the zip container comments into the input buffer with the specified
      input size.  If this size is not sufficient, the data are truncated.
      char *Comments    = buffer address
      int iCommentsSize = buffer size
      **/
      int GetComments(char *Comments, int iCommentsSize);
      /**
      Allows to set or reset that K2 has been intercepted at interface level.
      bool b = indicates that K2 indicator must be set or reset
      **/
      void K2given(bool b);
      /**
      Activated if the 'Cpu Exhausted' event occurs.
      **/
      void CpuExhausted();
      /**
      Activated if the 'Term' event occurs.
      **/
      void Term();
      /**
      Global return code
      **/
      //void *pRcErr; // CRcErr m_RcErr;
      //#define RcErr (*((CRcErr*)pRcErr))
      /**
      Return the last error maincode.  Use for LIST problem when -1 is returned
      **/
      int GetLastError();
   // private:
      /**
      Get the file name according to the link name.  The file name
      is returned in the ContainerName variable.
      ReadTFT rc is returned.
      char *pLink = link name
      **/
      long GetFileFromLink(char *pLink, char *pContainer);
      /**
      Check if the file exists.
      It returns 1 if the file exists, -1 if it exists but is empty,
      0 if it does not exist.
      char *pFileName = file name
      **/
      int FileExists(char* pFileName);
```

```
/**
Build a file name for output: :catid:$uid.FILEyyyymmdd.hhmmss
**/
void BuildFileName(char *fn, int i, char *catid=0, char *userid=0);
/**
Validate if the input file name is a valid file name
Return true is bs2000 file name compliant.
const char *fn = file name
**/
bool IsBS2000FileName(const char *fn);
/**
Not implemented
**/
static void fsRout(const char* fn, int i);
/**
Convert input string in upper case
char *s = string to convert
**/
void ToUpperCase(char *s);
/**
Catalog a container (PAM, STD16)
char *cont = container name
char *link = link name
int format = container format
bool space = true (define space in prg)
**/
int CatalogContainer(char *cont, char *link, int format, bool space);
/**
Get new file name using SDF wildcard construction
char *selection
char *construction
char *srcname
char *newname
**/
int GetNewFileName(char *selection, char *construction, char *srcname,
    char *newname);
/**
Get new file name using SDF wildcard construction
char *selection
char *construction
void *src = fileitem object
char *newname
**/
int GetNewFileName(char *selection, char *construction, void *src, char
    *newname);
/**
Common function adding files or lib elements in ZIP container
char *pFileNames = files or lib file name
```

```
    char *pElements = null ptr or element selector
    char *pVersion = null ptr or version selector
    char *pType = null ptr or type selector
    char *pBase = null ptr or base version base selector
    int iConvertMode = convert mode
    int iLevel       = compression level
    int iDataType    = data type (not specified, character, binary)
    char *pToFiles   = target construction file name
    int iTofileOption = specifies if libr element version and/or type are
       saved in the name
    int iDeleteOption = library element delete option                     132
    int iLogOption = logging option                                       133
    long nbr = number of added files                                      133
    **/
    int AddItems(char *pFileNames, char *pElements, char *pVersion, char
       *pType,
             char *pBase, int iConvertMode, int iLevel, int iDataType,
     char *pToFiles, int iTofileOption, int iDeleteOption, int iLogOption,
       unsigned long &nbr, void *flist);

    /* methods added for selection enhancement 1.2G */

    /* internal method
       pre-requisite CAddParam object has been created with all the parameters
       of the
       add statement
    */
    int AddToZip(void *paddparam);

    void SetExtractLogging(bool log);

    char    ContainerName[55];
    int     iCFormat;
    void *pZip;  //CZipArch m_Zip;
    #define Zip (*((CZipArch*)pZip))
    bool bRelLink;
    int m_OpenMode;
    bool k2Pressed;
    void UpdateUserInfo();
    bool bLib;
    int m_TofileOption;
    void *pAddParam; // CAddParam instance
    bool bExtractLoggingMax;                  // 134
};
// extern bool glb_sysout;  // msg on sysout by default
#endif // __SzpZip_h__
```

5.2 SZPZOUT.H

This C++ header file describes the contents of the buffer returned by SHOW-FILE-ATTRIBUTES operation (*ListFiles ()* function). The buffer returned by the function must be released by the caller.

```
#ifndef __SzpZout_h__
#define __SzpZout_h__
/*********************************************************************
 *  Classes         --
 *  File            szpzout.h
 *
 *  Copyright       (c) 'FUJITSU TECHNOLOGY SOLUTIONS' '2009'
 *
 *  Description     Output Find structure layouts
 *
 *  Version         120
 *  Date            2008-03-02
 *  Author          Ph. Dumont - OSL EPS
 *  Update          BS2ZIP v1.2A
 *
 *  Version         110
 *  Date            2004-09-30
 *  Author          L. Tambour - OSL EPS
 *  Update          BS2ZIP v1.1A
 *
 *  Version         100
 *  Date            2003-06-02
 *  Author          J. Beaume - OSL EPS
 *  Update          BS2ZIP v1.0B
 *********************************************************************/
struct szpOutSummary {
   unsigned int uRecSize;        // total size of a returned item
   void *pHeader;                // pointer to CFileHdr object
   unsigned short uFileNameSize; // file name size
   unsigned short BS2Flag;       // 1 = BS2000 file, 0 = others
   unsigned short ENCFlag;       // 1 = Encrypt. file, 0 = others (V120)
   // File name
};
struct szpOutFull {
   unsigned int uRecSize;        // total size of a returned item
   void *pHeader;                // pointer to CFileHdr object
   unsigned short uFileNameSize; // file name size
   unsigned short iBS2Flag;      // 1 = BS2000 file, 0 = others
   unsigned short iENCFlag;      // 1 = Encrypt. file, 0 = others (V120)
   char uModTime[8];             // last mod file time
   char uModDate[10];            // last mod file date
   //unsigned short uFiller;       // unused   V120
   long long uComprSize;         // compressed size
   long long uUncomprSize;       // uncompressed size
   unsigned short uCommentSize;  // file comment size
   unsigned short uExtraFieldSize; // extra field length
   unsigned int uFiller2;         // unused

   // Filename
   // Extrafield
   // Comment
};
#endif // __SzpZout_h__
```

Output buffers examples

1. INFORMATION=*SUMMARY

szpOutSummary structure	File name 1	szpOutSummary structure	File name 2
SzpOutSummary structure	File name 3		

2. INFORMATION=*ALL

szpOutFull structure	File name 1	Extra Fields 1	Comments 1
szpOutFull structure	File name 3	Comments 2	

5.3 BS2ZIPPR LLM

The run time BS2ZIPPR must be linked with the TU application that wants to use BS2ZIP as a sub-program.

5.4 Program example

```
#include "tstzip.h"
#include "SzpZout.h"
#include <string.h>
#include <stdio.h>

void main() {
   // create a CSzpZip object with error reported in rc and log file.
   CSzpZip *zip = new CSzpZip(false);

   // create a new container (BS2000 format by default)
   int rc = zip->OpenZip("MYCONT.ZIP", CSzpZip::updateNew, CSzpZip::defaut);

   // add a file to the container
   rc = zip->AddFiles("MYFILE.TXT", CSzpZip::std,
                          CSzpZip::defaultCompression);

   // list the contents of the container
   char *pBuf = 0;
   int iSize = 0;
   int iNumber = 0;
   // Get first element and loop while rc = 1;
   rc = zip->ListFiles("*", &pBuf, &iSize, CSzpZip::infoSummary);
   while(rc == 1) {
          iNumber++;

          // do something with buffer
          ...
          delete [] pBuf;

          rc = zip->ListFiles("*", &pBuf, &iSize, CSzpZip::infoSummary);
   }
   printf("Number of matching files = %d\n", iNumber);

   // extract file from the container - extracted file = EXT-MYFILE.TXT
   rc = zip->ExtractFiles("*", "EXT-*", CSzpZip::any, CSzpZip::notSpecified,
                          CSzpZip::std, CSzpZip::keep);

   // close zip container
   rc = zip->CloseZip();
   delete zip;
   }
```

Related publications

You will find the manuals on the internet at *http://manuals.ts.fujitsu.com*. You can order printed copies of those manuals which are displayed with an order number.

[1] **SDF** (BS2000)
SDF Dialog Interface
User Guide

[2] **BS2000/OSD-BC**
Commands
User Guide

[3] **JV** (BS2000)
Job Variables
User Guide

[4] **BS2000/OSD-BC**
DMS Macros
User Guide

Index