Miscellaneous Operations - UNIX

This chapter describes commands and options that are not tied specifically to compressing or extracting or can be done with both of these operations.

Overwriting Files

overwrite

When you add or extract files, the target archive or directory may already contain files that have the same names as the files you are adding or extracting. Use the ***overwrite*** option to tell PKZIP how to proceed.

This table describes the available sub-option choices for ***overwrite***:

Sub-Option	*Description*	*For example*
all	(Default) PKZIP overwrites all same-named files without prompting first	***pkzipc -extract -overwrite=all test.zip *.bmp*** ***pkzipc -add -overwrite test.zip *.bmp***
prompt	PKZIP prompts you whether to overwrite a same-named file before proceeding	***pkzipc -extract -overwrite=prompt test.zip *.bmp*** ***pkzipc -add -overwrite=prompt test.zip *.bmp***
increment	Increment file name to make it unique.	***pkzipc -extract -overwrite=increment test.zip *.bmp*** ***pkzipc -add -overwrite=increment test.zip *.bmp***
never	PKZIP does not overwrite any same-named files	***pkzipc -extract -overwrite=never test.zip *.bmp***

When you add files to an existing archive, PKZIP will, by default, overwrite files with the same name. You must use ***overwrite*** with your preferred sub-option to preserve existing files.

If you use ***extract*** alone, without the ***overwrite*** option, you are prompted to overwrite same-named files. If you use the ***overwrite*** option but do not specify a sub-option, PKZIP overwrites all files without prompting you.

Viewing the Contents of a ZIP File

view

PKZIP allows you to view the contents of a .ZIP file, without performing any action on that .ZIP file (for example, compress or extract). To view a .ZIP file, use the ***view*** option with PKZIP, as in the following example:

pkzipc -view test.zip

When you type this command, information similar to the following appears:

```
Viewing .ZIP: test.zip
 Length Method Size   Ratio Date       Time  CRC-32    Attr Name
 ------ ------ ----   ----- ----       ----  ------    ---- ----
 8369B DeflatN 3084B  63.2% 06/01/2015 4:50a 87b3c388 -a-w- red.txt
 8369B DeflatN 3084B  63.2% 06/01/2015 4:50a 87b3c388 -a-w- tan.txt
 ------        ------ -----                              ----
 16KB          6168B  63.2%                                 2
```

Note: The above ***view*** list was generated from a DOS command line. In a UNIX ***view*** listing, the "Attr " column would be replaced by an attributes "Mode " column.

PKZIP also provides three additional methods for displaying information from a .ZIP file. Specify the desired method as a value in addition to the ***view*** option. These methods include:

- ***brief*** - a compact, less informative view of the .ZIP file.
- ***detail*** - more information than the default view.
- ***security*** - adds information on whether the file is encryption and/or contains a digital signature.

Displaying a Brief View of a ZIP File

To display a more compact (brief) view of a .ZIP file, use the ***brief*** value with the ***view*** option, as in the following example:

pkzipc -view=brief test.zip

When you press ENTER, information similar to the following appears:

```
Viewing .ZIP: test.zip
 Length Method  Size   Ratio    Date    Time  Name
 ------ ------  ----   -----    ----    ----  ----
  8369B DeflatN 3084B  63.2% 06/01/2015 4:50a red.txt
  8369B DeflatN 3084B  63.2% 06/01/2015 4:50a tan.txt
 ------         ------ -----                  ----
 16KB           6168B  63.2%                    2
```

Displaying a Detailed View of the ZIP File

To display a more detailed view of a .ZIP file, use the ***details*** value with the ***view*** option, as in the following example:

pkzipc -view=details test.zip

When you press ENTER, information similar to the following appears:

```
Viewing .ZIP: test.zip

          FileName: red.txt
          FileType: text
        Attributes: -a-w--------
     Date and Time: Jun 01,2015 4:50:00a
Compression Method: DeflatN
   Compressed Size: 3084
 Uncompressed Size: 8369
       Compression: 63.2% - 2.948 bits/byte
  32 bit CRC value: 87b3c388
Version created by: PKZIP: 4.5
 Needed to extract: PKZIP: 2.0 or later

          FileName: tan.txt
          FileType: text
        Attributes: -a-w--------
     Date and Time: Jun 01,2015 4:50:00a
Compression Method: DeflatN
   Compressed Size: 3084
 Uncompressed Size: 8369
       Compression: 63.2% - 2.948 bits/byte
  32 bit CRC value: 87b3c388
Version created by: PKZIP: 4.5
 Needed to extract: PKZIP: 2.0 or later
 -------------------
       Total Files: 2
   Compressed Size: 6168
 Uncompressed Size: 16738
       Compression: 63.2% - 2.948 bits/byte
```

Displaying Security Information of the ZIP File

To display information on whether a.ZIP file, or files inside, are encrypted or digitally signed, use the ***security*** value with the ***view*** option, as in the following example:

pkzipc -view=security test.zip

When you press ENTER, information similar to the following appears:

```
Viewing .ZIP: test.zip

 Length  Method Size  Encryption Flag Date      Time  Name
 ------  ------ ----  ---------- ---- ----      ----  ----
 8369B  DeflatN 3396B AES (256)  --R  6/1/2015 11:33a red.txt
 8369B  DeflatN 3396B AES (256)  --R  6/1/2015 11:33a tan.txt
 ------         ----                                 ----
 16KB           6792B                                 2
```

The Security view is similar to the Brief view, with the Encryption and Flag columns replacing the Size column. If a file is encrypted, the Encryption column identifies the algorithm used to encrypt. The Flag column identifies whether a file is signed, and the method of encryption (passphrase or recipient). Flags are:

Value	*Description*
S	File is signed
P	File is passphrase-encrypted
R	File is recipient-encrypted

The Encryption column is empty when a file is not encrypted. Unencrypted and unsigned files display dashes (- - -) in the Flag column

Verifying the Encryption Type for an Archive

VerifyEncryption

Perhaps you've received an archive with one or more encrypted files. Before you can decrypt it, you need to know how it was encrypted. Use ***VerifyEncryption*** to determine the type of encryption used.
Whether you're creating an encrypted archive manually or with a script, you may also need to check whether an archive, or files in that archive, were encrypted, and how to decrypt them. This option will help as well.

This option answers yes or no to the question you ask. If you think the archive is encrypted with a recipient list, type:

-verifyEncryption=recipient test.zip

The results appear:

```
* Verify files are strongly encrypted for a recipient
Verify encryption .ZIP: C:\Users\Mike_m\Documents\ZipTest\test.zip
Encryption verification results: 2 passed, 0 failures
```

To determine whether an archive is strongly encrypted, use this command line to ask for both types of X.509-based strong encryption:

pkzipc -verifyEncryption=passphrase,recipient test.zip

Results for this command display whether test.zip passed the test:

```
* Verify files are strongly encrypted with a passphrase
* Verify files are strongly encrypted for a recipient

 Verify encryption .ZIP: C:\Users\Mike_m\Documents\ZipTest\test.zip

 Encryption verification results: 2 passed, 0 failures
```

Value	*Description*
none	Files are not encrypted
pkware	Files use traditional (weak) PKWARE encryption
aex	Files use AE-x encryption
passphrase	Files use strong passphrase encryption
recipient	Files use strong encryption based on X.509 certificates (recipient lists)
nopassphrase	Files do not use passphrase-based encryption
norecipient	Files do not use recipient-based encryption

Verifying Recipients Listed for a File

VerifyRecipient

Use ***VerifyRecipient*** to determine whether an archive, or files inside an archive, was encrypted for a specified person. Identify the X.509 certificate or OpenPGP key associated with the intended recipient.

To confirm whether the certificate using the common name "My Friend," use this command line:

pkzipc -verifyEncryption=recipient -verifyRecipient="My Friend" test.zip

To check for multiple certificates, chain the requests this way:

pkzipc -verifyEncryption=recipient -verifyRecipient=bob.smith@pkware.com -verifyRecipient="My Friend" test.zip

You can also use ***VerifyRecipient*** with OpenPGP files, using the recipient's key id in this command line:

pkzipc -verifyEncryption=recipient -archiveType=pgp-verifyRecipient=31C47555 test.pgp

Value	*Description*
#<file name>	Specifies a PKCS#7 file with its certificates
cn=<common name>	Specifies the common name associated with the certificate

e=<email>	Specifies the email address associated with the certificate
f=<filter>	Specifies an LDAP query filter string
kid=<openPGP keyid>	Specifies full key id of the associated OpenPGP key
default	Specifies the subject name of the X.509 certificate or the email address in the form of 'example@example.com'
8-digit hexadecimal string	Specifies a short OpenPGP key id
16-digit hexadecimal string	Specifies a long OpenPGP key id
20-digit hexadecimal string	Specifies the SHA1 hash of the public key for an X.509 certificate

Renaming Files

rename

The ***rename*** option renames files when you add or extract them. Used with the ***add*** command, the option renames files that you add so that they have a different name in the archive from their original name outside the archive. Used with the ***extract*** command, the option renames files as you extract them to give the extracted copies a different name from the name they have in the archive.

The ***rename*** option only renames the added or extracted copies of files, not the originals.

The ***rename*** option uses regular expressions and operates on both pathnames and file names. Regular expressions are a widely used methodology for transforming strings of symbols, such as text characters, by looking for specified patterns of symbols and replacing any matches with new patterns.

To use ***rename***, you specify two things: a text pattern to match, and a replacement pattern to substitute. Set off the patterns with a slash "/" to delimit them.

For example, the command line below archives all text files in the directory and renames the archived copies of any that have the string blue anywhere in their name so that blue is replaced with green.

pkzipc -add -rename=/blue/green/ mydata.zip *.txt

Sample results:

Name of original file	*Name of copy*
blue.txt	**green**.txt
bluesky.txt	**green**sky.txt
my**blue**sea.txt	my**green**sea.txt
here**blue**\star.txt	here**green**\star.txt

The ***rename*** option is particularly useful when adding or extracting streamed data. (See "Adding Data from STDIN or Special Files" and "Extracting Data to STDOUT or Special Files.")

For example, PKZIP can read data from standard input (STDIN) and add it to an archive. To tell PKZIP to get data from STDIN, a hyphen is used in place of a file name in the command line, and the data is referenced by a hyphen, not a file name, in the archive. Using the ***rename*** option, you can replace the hyphen with a file name when adding the data.

<command> | pkzipc add -stream -rename=/-/output.txt/ data.zip -

The command line above runs some program and pipes its output to PKZIP. PKZIP gets the data from STDIN (the hyphen at the end), renames it by replacing the hyphen with a file name, and archives the data under the file name.

Similarly, if the streamed data is referenced with a hyphen in the archive, you can use a ***rename*** expression to rename an extracted copy:

pkzipc extract-rename=/-/output.txt/ data.zip output.txt

You can use a character other than a slash to delimit the ***rename*** patterns. PKZIP reads the first character after the equals sign "=" as the delimiter. For example, the following command line uses a comma as the delimiter:

pkzipc -extract -rename=,green,blue/red, mydata.zip *.txt

You can also use a backslash to escape a character that would otherwise be mistaken for a delimiter, as in the following example. The backslash escape causes the character to be read as a literal character, not as a delimiter metasymbol:

pkzipc -extract -rename=/green/blue\/red/ mydata.zip *.txt

If the first character after the "=" sign is the list character (@ by default), the rest of the option is assumed to be the name of a list file containing one or more /match/replacement/ patterns.

By default, the pattern to match is case-sensitive. To ignore case when looking for matches, insert an "i" after the replacement pattern:

pkzipc -add -rename=/blue/green/i mydata.zip *.txt

You can use ***rename*** multiple times in a command line. PKZIP attempts all the specified replacements, in the order given, on all added or extracted files. For example, the command line below first replaces blue with green in all files and then replaces txt with doc in all files.

pkzipc -add -rename=/blue/green/ -rename=/txt/doc/ mydata.zip *.txt

The command line above renames *.txt files to *.doc files, but it also replaces any occurrences of txt that are not at the end of the file name. For example, a file named txture.txt is renamed to docure.doc. Regular expression syntax provides metasymbols to help you deal with this sort of situation by specifying position, variables, and quantifiers.

For example, a dollar sign "$" at the end of a ***rename*** pattern to be matched, as in the command line below, specifies that the pattern must occur at the end of the file name.

pkzipc -add -rename=/blue/green/ -rename=/txt$/doc/ mydata.zip *.txt

The preceding command line renames txture.txt to txture.doc. Only the txt at the end is matched and changed.

To specify a literal dollar sign as part of a file name to match—for example, my$money.txt—escape it with a backslash "\". A backslash in front of a regular expression metasymbol matches the literal symbol instead of treating it as a metasymbol. (Similarly with the backslash metasymbol itself.)

... -rename=/my\$money.txt/my$money.doc/ ...

The metasymbols listed below must be escaped with a backslash to use them as literal symbols in a pattern to match. To use them as metasymbols, use without a backslash.

```
\ | ( ) [ { ^ $ * + ? .
```

The following table gives a brief account of what most of these metasymbols do when used in a pattern to match. Regular expression syntax provides for many additional pattern-matching possibilities. PKZIP regular expressions follow the POSIX standard.

Symbol	Meaning
\	The escape character. Turns a non-alphanumeric metasymbol into a literal.
… | ...	Alternation ("or"). Says to match either alternative
(…)	Delimits a group; for example, use around alternatives: (a | b)
[…]	A class expression. Says to match any single symbol in the class. For example: [abc]
^	The beginning of a line (file name). For example: ^txt matches txt only when the letters are the first three in a file name.
$	The end of a line (file name). For example: txt$ matches txt only when the letters are the last three in a file name.
*	A quantifier meaning "zero or more" of the preceding element. For example, bo* matches b, bo, boo, booo, and so on.
+	A quantifier meaning "one or more" of the preceding element. For example, bo+ matches bo, boo, booo, and so on
?	A quantifier meaning "zero or one" of the preceding element. For example, bo* matches b or bo.
.	A variable; matches any single symbol

Translating End-of-Line Sequence

translate

The ***translate*** option translates text end-of-line characters to the character sequence used by a different platform. The option can be used with ***add*** or ***extract***. Specify a sub-option from the following table to translate line endings to the sequence used by the desired platform.

The ***ebcdic*** sub-options are for use with data compressed using SecureZIP for z/OS with the Zip Descriptor Word (ZDW) option to preserve variable length records. If a file is not in ZDW format, these sub-options cause no change to line endings.

Sub-Option	*Description*
none	Does not change line endings
dos	DOS/Windows (carriage return/newline)
mac	MacOS 9 and earlier (carriage return)
unix	UNIX and MacOS 10.x (newline)

ebcdic,nl	With ZDW files, substitute EBCDIC newline (0x15)
ebcdic,lf	With ZDW files, substitute EBCDIC linefeed (0x25)
ebcdic,crlf	With ZDW files, substitute EBCDIC carriage return/linefeed (0x0D25)
ebcdic,lfcr	With ZDW files, substitute EBCDIC linefeed/carriage return (0x250D)
ebcdic,crnl	With ZDW files, substitute EBCDIC carriage return/newline (0x0D15)

The following command line translates text line endings to UNIX on extraction:

pkzipc -extract -translate=UNIX test.zip

Converting File Names to a Short Format

shortname

The ***shortname*** option enables you to convert file names in long file name format to DOS-format short (8+3) file names on the copies of the files added to an archive. Use ***shortname*** with the ***dos*** sub-option, or no sub-option at all, to specify DOS format:

pkzipc -add -shortname=dos save.zip

pkzipc -add -shortname save.zip

Or, abbreviated:

pkzipc -add -short save.zip

The option can be configured to be on by default.

Use ***shortname*** with the ***none*** sub-option to turn short name formatting off if it's configured on.

Inserting a Timestamp in the Archive File Name

substitution

The ***substitution*** option causes PKZIP to insert a timestamp in the name of an archive created or updated (or refreshed) by the ***add*** command. You specify the elements of the timestamp and its placement in the archive name.

The ***substitution*** option can also insert a timestamp in the same way in the name of a destination directory specified as a sub-option of the ***archiveeach*** option.

Note: See "Time Stamping Your Signed ZIP Archive" for information on using an independent Time Stamp Authority to securely establish when a file was created or modified.

Construct the timestamp using tokens (replaceable elements) from the table below. When embedded in an archive file name, the tokens serve as named parameters. The ***substitution*** option causes PKZIP to replace the tokens with the corresponding values listed in the table. (If the ***substitution*** option does not appear in the command line, the tokens become literal parts of the file name.)

Token	***Replaced by***
{id}	A job ID specified separately with the ***jobid*** option. For example, if run in 2006: ***pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc*** produces a ZIP file named: `myJob2006.zip`
{mm}	Month, 2-digit
{m}	Month, 1-digit (if possible)
{dd}	Day, 2-digit
{d}	Day, 1-digit (if possible)
{yyyy}	Year, 4-digit
{yy}	Year, 2-digit
{HH}	Hour, 2-digit, 24-hour format

{H}	Hour, 1-digit (if possible), 24-hour format
{hh}	Hour, 2-digit, 12-hour format
{h}	Hour, 1-digit (if possible), 12-hour format
{MM}	Minute, 2-digit
{M}	Minute, 1-digit (if possible)
{SS}	Second, 2-digit
{S}	Second, 1-digit (if possible)
{ampm}	a.m. or p.m. indicator to identify current 12-hour segment of the day

For example, the following archive name contains several tokens. The name is enclosed in quotes to group the elements, including the spaces:

"Design Spec {yyyy}-{mm}-{dd}-{h}-{MM}-{SS}{ampm}.zip"

Note: Most UNIX shells treat { and } as metacharacters, which need to be escaped for the command line to work properly. To be safe, put the whole file name or path name in quotation marks when using the ***substitution*** option.

The following command line adds files to an archive having this name and includes the ***substitution*** option to tell PKZIP to replace the tokens with their system values:

pkzipc add -substitution "Design Spec {yyyy}-{mm}-{dd}-{h}-{MM}-{SS}{ampm}.zip" plan.doc

If the current date and time are August 09, 2014 12:06:29 a.m., the resulting archive will be named `Design Spec 2014-08-09-12-06-29am.zip`.

The ***substitution*** option can also be used to embed a timestamp in the name of a destination directory specified with the ***archiveeach*** option. For example:

pkzipc add -substitution -archiveeach="{HOME}/newzips {yyyy}-{mm}-{dd}-{h}-{MM}-{SS} {ampm}" {HOME}/myfiles*.*

The preceding command line causes each file zipped from the `myfiles` directory to be added to its own archive in a directory named `newzips 2014-08-09-12-06-29am.zip` if the date and time are August 09, 2014 12:06:29 a.m.

The ***substitution*** option can be configured to be used by default

Testing the Integrity of an Archive

test

You can test an archive to confirm that it is not damaged and that its files can be extracted. Testing also authenticates any digital signatures attached.

Testing extracts the contents of an archive but discards the output instead of saving it to disk.

It's a good idea to test an archive before you delete your only copy of an important file you placed in the archive.

The following sample command line tests test.zip:

pkzipc -test test.zip

When you press ENTER, information similar to the following will appear:

```
Testing files from .ZIP: test.zip

Testing: readme.txt OK
Testing: whatsnew.txt OK
```

As each file is tested, an OK is displayed next to the name. If the archive has been damaged, use the ***fix*** command to try to repair it.

Handling Warnings

PKZIP issues warnings or errors when it encounters a problem or if something unexpected happens. These reports can bring problems of their own, especially when running PKZIP in a script. The ***warning***, ***error***, and ***IgnoreWarnings*** options allow you to deal with these issues in various ways.

Pausing on Warnings

warning

PKZIP issues an error or a warning when it encounters a problem or unexpected condition. In general, PKZIP issues a warning when the condition does not prevent PKZIP from completing its operation, and an error when it does. For example, PKZIP issues a warning if a digitally signed file in an archive cannot be authenticated; this condition does not prevent PKZIP from extracting the file. PKZIP issues an error if it cannot find a specified archive or is unable to open it.

The ***warning*** option causes PKZIP to pause after issuing a warning and to prompt you whether to proceed. The option can be set for specified warning conditions. If used without any specified values, the ***warning*** option causes PKZIP to pause on every warning. For example:

pkzipc -extract -warning save.zip *

To have PKZIP pause and prompt on particular warnings, list the warning numbers with the option. For example, the following command line directs PKZIP to pause on warning 43 (*Certificate not found*):

pkzipc -add -warning=43 -recipient=xxx foo.zip *.doc

To specify multiple warning conditions, separate the warning numbers with commas. For example, the following command line tells PKZIP to pause and prompt on either warning condition 42 (*Certificate was revoked*) or 43:

pkzipc -add -warning=42,43 -recipient=xxx foo.zip *.doc

You can use the ***configuration*** command to specify warning numbers as default values for the ***warning*** option. If default warning values are specified, you do not need to explicitly include the ***warning*** option in a command line to pause on those warnings.

To override a configured default warning setting for the ***warning*** option in the current command line, precede the warning number with a hyphen. For example, the following setting (in a command line) overrides a configured value of (warning) 43. The example causes PKZIP *not* to pause on warning 43.

-warning=42,-43

The ***warning*** option can be used with the ***add***, ***extract***, ***test***, and ***view*** commands. See this page for a list of error and warning conditions.

Treating Warnings as Errors

error

The ***error*** option enables you to designate warnings, by number, to treat as errors such that PKZIP halts processing if a specified warning condition is encountered.

A designated warning is treated as error number 73, *Warning configured as an error*.

Multiple warning numbers can be specified, separated by commas:

-error=42,43

For example, the following command line tells PKZIP to treat the conditions that produce warnings 42 (*Certificate was revoked*) and 43 (*Certificate not found*) as error conditions:

pkzipc -add -error=42,43 -recipient=xxx foo.zip *.doc

If a specified warning is generated, PKZIP halts processing. Both the triggered warning and an error 73 are issued.

For example, if warning 43 is generated, the display looks like this:

```
PKZIP: (W43) Warning! Certificate not found: xxx
PKZIP: (E73) Warning configured as an error
```

You can use the ***configuration*** command to specify warning numbers as default values for the ***error*** option. If default warning values are specified for the ***error*** option, you do not need to explicitly include the ***error*** option in a command line to treat those warnings as errors.

You can override a configured default warning setting for the ***error*** option in the current command line. To override a warning setting, precede the warning number with a hyphen.

The following example (in a command line) overrides a configured value of (warning) 43. The example causes warning 43 *not* to be treated as an error.

-error=42,-43

The ***error*** option can be used with ***add***, ***extract***, ***test***, and ***view***. See this page for a list of error and warning conditions.

Ignoring Warnings

IgnoreWarnings

The ***IgnoreWarnings*** option enables you to designate warnings, by number, to display, but not affect the program's return code. That is, PKZIP will not return the value ***1*** when ignored warnings occur.

For example, if you want to designate an LDAP group as a recipient, you can tell SecureZIP to ignore the "Multiple certificates found" warning with this command line:

pkzipc -add -ignore=59 -recipient=<group> foo.zip *.doc

Multiple warning numbers can be specified, separated by commas:

-ignore=40,41,79

You can use the ***configuration*** command to specify warning numbers as default values for the ***IgnoreWarnings*** option. If default warning values are specified for ***IgnoreWarnings***, you do not need to explicitly include ***IgnoreWarnings*** in a command line.

You can override a configured default warning setting for ***IgnoreWarnings*** in the current command line. To override a warning setting, precede the warning number with a hyphen.

The following example (in a command line) overrides a configured value of (ignoring) 41. The example causes warning 41 *not* to be ignored. If this warning appears, it will be processed normally.

-ignore=42,-41

The ***IgnoreWarnings*** option can be used with ***add***, ***extract***, ***test***, and ***view***. See this page for a list of error and warning conditions.

Previewing Command and Option Operations

preview

PKZIP allows you to preview the results of a set of commands and options. The commands and options specified will be completed and the resulting output will display, but no changes will be made that result in creating a new .ZIP file or in modifying an existing .ZIP file. For example, if you wish to preview an add operation without creating or modifying any files, enter the following:

pkzipc -add -preview test.zip *.txt

When you press ENTER, information similar to the following appears on your console:

```
  Using Preview Option
 Creating .ZIP: test.zip
   Adding File: readme.txt Deflating (62.0%), done.
   Adding File: whatsnew.txt Deflating (59.2%), done.

 The compressed .ZIP file size would be: 2237 bytes
```

The information, including the size of the resulting .ZIP file, is displayed. However, PKZIP has not actually modified any of your files. The ***preview*** option will work with ***add***,***delete***,***header***,***sfx,*** and ***comment***.

Fixing a Corrupt ZIP File

fix

The ***fix*** command attempts to repair a damaged ZIP archive so that its files can be extracted.

For example, if you have determined that test.zip is damaged, type the following to attempt to fix it:

pkzipc -fix test.zip

When you press ENTER, information similar to the following appears on your console:

```
 Enter a new .ZIP file name (pkfixed): test1.zip

 Running PKZipFix utility.

 Scanning .ZIP file: test.zip
 Building new directory.
 Writing new .ZIP file: test1.zip

 Recovered 2 files.
```

When you enter the ***fix*** command, PKZIP prompts you to enter a new ZIP file name. The example above used test1ZIP. If you do not enter a file name, the name pkfixed.ZIP is used. PKZIP scans the original file, attempts to repair the archive, and saves the updated file with the new name. The original, damaged file is not updated.

Note: The ***fix*** command can only fix ZIP archives that are physical files. It cannot fix ZIP archives read from STDIN or special files (named pipes, sockets). Nor can it output fixed archives to such targets.

Use an Alternate Drive for PKZIP Temporary Files

temp

The ***temp*** option enables you to specify an alternate location for the temporary file that PKZIP needs to create to update an existing ZIP file or create a spanned archive. PKZIP also creates a temporary file when writing an archive to a data stream (see "Writing an Archive to STDOUT and Special Files").

When you, for example, update a ZIP file, PKZIP first creates and updates a temporary copy of the file. When the update is completed, PKZIP replaces the original archive with the updated copy.

In the case of an archive written to a data stream, PKZIP compresses and encrypts the data (if encryption is specified) before writing it to the temporary file, so no security vulnerability is created. The temporary file is needed to get size information for local headers, which are written out before file data.

The amount of disk space PKZIP needs for the temporary file is equal to the size of the original ZIP file plus the compressed size of any files to be added. So, for example, if you have an existing ZIP file of 500K, and you are updating it with another file that is 10K compressed, you need a work space of at least 510K for PKZIP to do the update.

Ordinarily, the temporary file is created in the system's default temporary folder. With the ***temp*** option, you can span, update, or stream ZIP files that are larger than the space available to create a temporary file in the default location.

Specify the drive and/or path for the temporary file as a sub-option of ***temp***. For example, the following command lines specify a custom temporary file location to update `big_file.zip`.

pkzipc -add -temp=/usr/tmp big_file.zip myfile.doc

Notes:

- You need to provide a path in addition to the drive letter only if you have a reason to specify a subdirectory—for example, space or access constraints on a local area network.
- The ***shred*** option cannot erase temporary files created using the ***temp*** option to specify a location on a removable or network drive.

Suppressing Screen Output

silent

The ***silent*** option suppresses screen output when compressing or extracting. This option is useful when compressing or extracting files as part of .BAT, .CMD, or shell script operations. Messages that normally appear when compressing or extracting are not displayed. Sub-options provide control over whether to display error messages, warning messages, requests for input, and so on.

pkzipc -add -silent test.zip *.doc

To suppress confirmation messages printed by the ***configuration*** command, use the ***configuration*** command with its own ***silent*** sub-option.

Setting Internal Attributes

ASCII/BINARY

The ***ASCII*** and ***BINARY*** option is used to override the data type of a file. Normally, PKZIP will determine whether the data of a file is ASCII or Binary. If this option is used with no sub-option, you will be prompted for each file that is added to set to ASCII, BINARY or if PKZIP should determine the best type. The following examples show the different uses for this option.

To set all the internal attributes to ASCII for each file added:

pkzipc -add -ascii="*" test.zip

To set all the internal attributes for the file test.txt to BINARY and auto detects the other files:

pkzipc -add -binary=test.txt test.zip *

To prompt the type for each file:

pkzipc -add -ascii test.zip *

Encoding an Archive to another Type

encode

With the ***encode*** option, you can convert an archive from one type to another.

The ***encode*** option is useful to encode a binary archive type to a text format such as UUEncode or XXEncode. It can also be used to convert a non-compressed archive to a compressed archive type.

For example, a TAR archive can contain multiple files but does not compress them, and a GZIP archive compresses but can contain only one file. You can use ***encode*** with ***add*** to create (or update) a TAR archive and encode it to GZIP format:

pkzipc -add -encode=gzip myfiles.tar

The example creates two archives: a TAR file and a GZIP file `myfiles.tar.gz`.

If you want only the archive created by ***encode*** (the GZIP archive in the example), you can include the ***movearchive*** option to delete the intermediate (TAR) archive:

pkzipc -add -encode=gzip -movearchive myfiles.tar

You can also use ***encode*** as a command to convert an existing archive. To do so, use the ***encode*** command by itself on the command line, without the ***add*** command, and specify the archive to convert. For example, the following command line creates an archive `save.tar.gz`:

pkzipc -encode=gz save.tar

ASCII armor (Radix-64) is a character format that creates an ASCII character stream that could be used in transferring OpenPGP files through transport mechanisms that can only handle character data (for example, email body text). You can use ***encode*** to create these files, as in this command line:

pkzipc -archivetype=pgp -encode

Note: The ***encode*** command/option can only convert physical archive files. It cannot read an archive to be converted from STDIN or a special file (named pipe, socket). Nor can it write an encoded archive to STDOUT or a special file.

Removing an Intermediate Archive

movearchive

The ***movearchive*** option deletes an archive that is created only as an intermediate archive—for example, to be converted by the ***encode*** option to an archive of a different type.

When you add files with the ***encode*** option, PKZIP creates two archives: an intermediate archive created by the ***add*** command, and an archive of the type specified with the ***encode*** option. The encoded archive is created from the intermediate archive.

If you do not want to keep the intermediate archive, you can include the ***movearchive*** option to delete it. For example:

pkzipc -add -encode=gzip -movearchive myfiles.tar

The command line above creates a TAR archive, encodes a copy of this archive as a GZIP archive, and then deletes the intermediate TAR archive.

Generate a List File

listfile

The ***listfile*** option is used with ***add*** and ***extract*** to create a list of the files that *would be* added or extracted if the command line were run without the ***listfile*** option. A command line that contains the ***listfile*** option just creates a list file; it does not add or extract any files.

For example, the following command line creates a file `mylist.txt` with the names of all the files that would be added to, or updated in, `myarchive.zip` if the ***listfile*** option were omitted from the command line:

pkzipc -add=update -listfile=mylist.txt myarchive.zip *.*

When ***listfile*** is used with ***add***, you can omit the archive name unless you want to reference a particular archive. For example, the following command line creates a list of the files that the command line would add to any new archive:

pkzipc -add -listfile=mylist.txt *.*

On the other hand, if you want to see what files would be updated in some particular archive, as in the following command line, you must name the archive:

pkzipc -add=freshen -listfile=mylist.txt myarchive.zip *.txt

When used with ***add*** (though not with ***extract***), the ***listfile*** option takes account of other options—for example, the options ***path***, ***recurse***, and ***directories*** that specify path information to save with the added files.

For example, the ***path*** option in the following command line causes full path names to be saved with added files, so this information is saved in the list file as well:

pkzipc -add -path=full -listfile=mylist.txt myarchive.zip *.*

When used with ***extract***, the ***listfile*** option lists files with any path information saved for them in the archive even if current option settings would otherwise extract the files without using saved path information.

For example, the following command line creates a list file that includes any path information in the archive even though the ***path*** option directs that files be extracted without using saved path information:

pkzipc -extract -path=none -listfile=mylist.txt myarchive.zip

Logging Events

LogError, JobID, Log, logaudit, LogOptions

You can have PKZIP log records of warnings, errors, and normal operations. Records can be written to STDOUT, STDERR, the native system logging facility (syslog) for your platform, or to a file.

The ***log*** option controls the logging of messages that relate to normal operations of PKZIP. The ***logerror*** option controls the logging of messages that relate to errors and warnings. Sub-options enable you to direct output.

On UNIX, log entries sent to the syslog are prefixed with a datestamp followed by PKZIP[XX], where XX is the number of the process ID of the PKZIP invocation that generated the message. With this information, you can identify messages that relate to PKZIP and determine which PKZIP messages belong to which process when more than one PKZIP process is running.

The server logs to the LOG_USER facility, with an application name of PKZIP. Entries are logged with the priorities listed below:

Log entry type	*Priority*
Errors	LOG_ERR
Warnings	LOG_WARNING
All else	LOG_NOTICE

Executing a single command line can produce multiple syslog entries. For example, updating an archive may produce one entry for adding a file and additional entries for copying existing files. You can also use the ***jobid*** option to attach your own job ID to each of a set of related entries in the syslog.

The ***logaudit*** option lets you integrate PKZIP with Security Incident and Event Management (SIEM) applications. Configure this command to have PKZIP log audit messages and specify a file to store the audit log.

The ***logoptions*** option enables you to write to the syslog an entry containing the current command line each time PKZIP starts. You can also set ***logoption*** to log an entry in the syslog to report encryption information on each processed file, and when PKZIP finishes.

The options ***log***, ***logerror***, ***logaudit*** and ***logoptions*** can all be configured for use by default.

You can log entries to syslog and/or any *one* of the following destinations: STDOUT, STDERR, or a file. So, for example, the ***log*** setting in the following example is allowed, but a setting of log=stderr,mylog.txt would produce an error:

pkzipc -log=syslog,mylog.txt -logoptions=start -jobid=my_id2 -add myarchive.zip bookmark.htm

Events logged to a file overwrite the file for each command line executed. Entries for events logged to a file are not prefixed with *PKZIP*, a job id, or a process ID.

Sending Information to an SNMP Host

SnmpTrapHost

SNMP (Simple Network Management Protocol) is a standard protocol for monitoring and managing activity on a network. It provides for applications to send messages, called *SNMP traps*, to an SNMP receiver application on a network server to inform the receiver application of selected events such as error and warning conditions. PKZIP can also send traps on application startup and shutdown.

An SNMP receiver can be configured to respond to traps in various ways—for example, page someone, send an email, log certain kinds of messages, or forward the traps to another receiver.

The ***SnmpTrapHost*** option enables you to specify an SNMP host machine running an SNMP receiver for PKZIP to send SNMP traps to. The option can be configured for use by default. You specify the traps you want to send by setting sub-options for the ***log***, ***logoptions***, and ***logerror*** options.

Note: PKWARE sends SNMP version 2 traps, using the UDP protocol (User Datagram Protocol). Version 2 traps are not encrypted; PKWARE SNMP traps are intended to be used within a mostly trusted internal network, not across the Internet at large.

The ***SnmpTrapHost*** option takes a three-part value consisting of an SNMP host name or IP address, an optional community name, and an optional port number. The syntax is as follows (optional fields set off by brackets; do not type the brackets):

-snmptraphost=[community@]host[:port]

where:

- community (optional) is the community name; default is ***public***
- host is the SNMP host name or IP address
- port (optional) is the port number. The default SNMP trap port is 162.

The following sample command lines use ***SnmpTrapHost*** to specify an SNMP host to receive traps sent for an add and an extract operation, respectively. The type of trap that may be sent—informational, warning, or error—depends on how the ***log*** and ***logerror*** options are set (see the next section, "Kinds and Contents of SNMP Traps Sent").

pkzipc -add mydocs.zip *.doc -snmptraphost=nmsnode1
pkzipc -extract backup1.zip -snmptraphost=private@hostxyz:20001

Kinds and Contents of SNMP Traps Sent

A trap sent by PKZIP always contains the IP address and time on the machine running PKZIP; it contains other information besides, depending on the trap. For example, a trap may contain a message ID and/or message text, the PKZIP command line executed, a PKZIP job ID, and so on.

The following table lists the kinds of events for which PKZIP sends traps and the option settings that cause traps to be sent. (See the section "The PKWARE MIB," below, for information about trap names such as `pkZipInfoTrap`.)

Table: SNMP Traps

Event	*Type of Trap*	*Option Setting to Send Trap*
Application startup	Informational (`pkZipInfoTrap`)	***-logOptions=start***
Application shutdown	Informational (`pkZipInfoTrap`)	***-logOptions=stop***
Normal operation Sends a trap for each normal operation that generates a message to STDOUT	Informational (`pkZipInfoTrap`)	***-log=snmp***
Warning condition	Warning (`pkZipWarnTrap`)	***-logerror=snmp***
Error condition	Error (`pkZipErrTrap`)	***-logerror=snmp***

The PKWARE MIB

In the table of SNMP traps in the preceding section, the names listed for types of traps (`pkZipInfoTrap`, for example) are defined in the PKWARE MIB (Management Information Base) file.

The MIB file describes the structure of an SNMP message and its elements. When parsed by any of various standard MIB tools, the file enables an application to provide friendly, human-readable names for SNMP message elements. Natively, in an SNMP message, these are expressed as a series of digits—for example, 1.3.6.1.4.1—where each digit represents a branch in a hierarchical tree of known (that is, officially registered) SNMP names. The series 1.3.6.1.4.1, for example, corresponds to the branch iso.org.dod.internet.private.enterprise, where 1 in the first position represents iso, 3 in the second position represents org, and so on.

On installation, the PKWARE MIB file is placed in the main product directory (with the `readme.txt` file).

Setting Execution Priority

priority

The ***priority*** option sets the execution priority of PKZIP with regard to other programs running on the same system.

On UNIX, the priority levels range from 0-39; 20 is the default.

The ***priority*** option can be used only to lower the priority on UNIX.

Adjusting priority of execution can affect the performance of PKZIP but not always in a predictable fashion.

The ***priority*** option can be configured for use by default but must be included on the command line to take effect.