



Command Line

Users Manual

PKZIP[®] Command Line

SecureZIP[®] Command Line

Copyright © 1997-2011 PKWARE, Inc. All Rights Reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any other language in whole or in part, in any form or by any means, whether it be electronic, mechanical, magnetic, optical, manual or otherwise, without prior written consent of PKWARE, Inc.

PKWARE, INC., DISCLAIMS ALL WARRANTIES AS TO THIS SOFTWARE, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, FUNCTIONALITY, DATA INTEGRITY, OR PROTECTION. PKWARE IS NOT LIABLE FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES.

PKWARE, the PKWARE logo, the zipper logo, PKZIP, PKUNZIP, SecureZIP, and PKSFX are registered trademarks of PKWARE, Inc. Deflate64 is a trademark of PKWARE, Inc.

Trademarks of other companies mentioned in this documentation appear for identification purposes only and are the property of their respective companies.

2011-10-25

Table of Contents

| | |
|--|-----------|
| 1 THE BASICS | 1 |
| About This Manual | 1 |
| Conventions in This Guide..... | 2 |
| An Overview of What PKZIP Does | 2 |
| Supported Archive Types | 2 |
| Your Work Environment: The Command Line | 3 |
| Entering Commands | 4 |
| Creating a New Archive and Adding Files | 5 |
| Archive File Naming Conventions..... | 6 |
| Adding a Single File | 6 |
| Adding Multiple Files..... | 7 |
| Moving Files into an Archive..... | 8 |
| Viewing Files in an Archive..... | 8 |
| Extracting Files from an Archive | 9 |
| Extracting All Files | 9 |
| Extracting Some Files..... | 9 |
| Extracting Files to a Different Directory | 10 |
| Extracting New and Newer Files..... | 10 |
| Using Filters When Selecting Files | 10 |
| Selecting Files by Date | 11 |
| Selecting Files by Age | 11 |
| Selecting Files by Size..... | 12 |
| Selecting Files to Include or Exclude..... | 12 |
| Understanding Commands and Options | 13 |
| Difference between a Command and Option..... | 13 |
| Including an Option in Your Command Line | 14 |
| Abbreviating Commands and Options | 14 |
| Using Multiple Options | 14 |
| Commands and Options with Values..... | 15 |
| Using Strong Encryption | 16 |
| 2 GETTING STARTED | 17 |
| Learning More and Getting Help | 18 |
| Using Help | 18 |
| Getting Version Information | 18 |
| Technical Support | 19 |
| Working With Your License | 19 |

| | |
|--|-------------------------------------|
| Entering License Keys | 19 |
| Setting PKZIP in the Path | 20 |
| 3 ADDING FILES TO AN ARCHIVE | 20 |
| Default Values for Commands and Options | 21 |
| Creating and Updating Archives..... | 21 |
| Adding All Files in a Directory..... | 21 |
| Adding New and Modified Files | 21 |
| Adding Only Files That Have Changed | 22 |
| Incremental Archiving | 22 |
| Encrypting Files That You Add to an Archive | 23 |
| Encrypting Files with a Passphrase..... | 24 |
| Encrypting Files with a Recipient List | 25 |
| Encrypting File Names..... | 27 |
| Encrypting Using Only FIPS-Approved Algorithms | 28 |
| Accessing Recipients in an LDAP Directory | Error! Bookmark not defined. |
| Contingency Keys | Error! Bookmark not defined. |
| Creating OpenPGP Files | 30 |
| Attaching Digital Signatures | 31 |
| Commands and Options for Signing Archives..... | 31 |
| Setting a Default Certificate | 35 |
| Time Stamping Your Signed ZIP Archive | 35 |
| Writing an Archive to STDOUT and Special Files | 36 |
| Writing an Archive to STDOUT | 36 |
| Writing an Archive to a Named Pipe..... | 37 |
| Compressing Files in Subdirectories..... | 38 |
| Compressing Open Files | 38 |
| Storing Directory Path Information | 39 |
| Additional Methods for Storing Directory Path Information..... | 39 |
| Storing and Recreating Directory Path Information | 40 |
| Setting the Compression Level..... | 42 |
| Specifying a Compression Level from 0-9 | 42 |
| Specifying a Compression Level by Name | 43 |
| Compressing Files with a List File | 44 |
| Getting a List of Files from Standard Input | 44 |
| Compressing Files with the Deflate64 Method..... | 45 |
| Compressing Files with the BZIP2 Method..... | 45 |
| Compressing Files with the LZMA Method | 45 |
| Compressing Files Compatible with the Data Compression Library..... | 46 |

| | |
|---|-----------|
| Compressing Files with the PPMd Method | 46 |
| Compressing Files to a Specified Type of Archive | 46 |
| Compressing Files to Diskette | 47 |
| Creating a Spanned Archive..... | 47 |
| Creating a Split Archive | 47 |
| Preserving International Characters in File Names | 48 |
| Creating Multiple, Respective Archives | 49 |
| Storing File Information | 50 |
| Compressing Files with Specified Attributes..... | 50 |
| Extended Attribute Storage..... | 51 |
| Including Additional Information in a ZIP File | 52 |
| Including a Text Comment..... | 52 |
| Including a Header Comment | 53 |
| Specifying the Date of a .ZIP File | 53 |
| Removing File Attributes | 54 |
| Sorting Files Within a .ZIP File | 55 |
| Moving Files to a .ZIP File | 56 |
| Shredding Deleted Files | 56 |
| Working with Self-Extracting (PKSFX) Archives | 57 |
| Converting a Standard Archive to a Self-Extractor | 58 |
| Converting to a Self-Extractor with a Different Name | 59 |
| Options for Creating Self-Extractors | 59 |
| Run Programs with the Self-Extractor | 61 |
| Extraction Options for the Native Self-Extractor | 61 |
| 4 EXTRACTING FILES | 63 |
| Default Values for Commands and Options | 63 |
| Extracting New and Existing Files | 63 |
| Extracting All Files from an Archive | 63 |
| Extracting Newer Versions of Existing Files and New Files | 64 |
| Extracting Only Newer Versions of Files | 64 |
| Checking for Viruses when Extracting | 64 |
| Extracting from an Archive Embedded in an Archive | 65 |
| Extracting Passphrase-Protected Files | 66 |
| Extracting an Archive on STDIN or a Special File | 67 |
| Extracting from an Archive on STDIN..... | 67 |
| Extracting an Archive from a Named Pipe..... | 67 |

| | |
|---|-----------|
| Extracting Files in Lower Case | 70 |
| Preserving File Times | 70 |
| Retaining Directory Structure while Extracting..... | 70 |
| Retaining Zone Identifier Information for Downloaded Files..... | 71 |
| Sorting Files in the Extract Directory | 71 |
| Extracting Files Only for Display | 72 |
| Extracting Files with a List File..... | 72 |
| Authenticating Digital Signatures..... | 72 |
| Extracting Only Trusted Archives | 74 |
| Specifying Trusted Signers..... | 74 |
| 5 WORKING WITH DIGITAL SIGNATURES..... | 76 |
| Public-Key Infrastructure and Digital Certificates | 76 |
| Public-Key Infrastructure (PKI) | 77 |
| How the Keys Are Used..... | 77 |
| X.509..... | 77 |
| Digital Certificates | 77 |
| Certificate Authority (CA) | 78 |
| Private Key..... | 78 |
| Public Key | 78 |
| Certificate Authority and Root Certificates | 78 |
| Using Digital Signatures | 78 |
| Attaching a Signature to an Existing Archive..... | 79 |
| Applying Strict Checking to Certificates..... | 79 |
| Checking for Revoked Certificates | 80 |
| Using Digital Certificates on Windows | 81 |
| Advanced Encryption Options in Windows | 83 |
| Working with OpenPGP Files..... | 84 |
| Overview: OpenPGP vs. X.509 | 84 |
| Setting Up OpenPGP Keyrings..... | 85 |
| Configuring Other OpenPGP Settings | 86 |
| 6 MISCELLANEOUS OPERATIONS | 87 |
| Overwriting Files | 87 |
| Viewing the Contents of a ZIP File..... | 88 |
| Displaying a Brief View of a ZIP File..... | 88 |
| Displaying a Detailed View of the ZIP File..... | 88 |
| Translating End-of-Line Sequence | 89 |
| Converting File Names to a Short Format | 90 |

| | |
|---|------------|
| Inserting a Timestamp in the Archive File Name | 90 |
| Printing the Contents of a ZIP File..... | 92 |
| Testing the Integrity of an Archive | 92 |
| Pausing on Warnings..... | 93 |
| Treating Warnings as Errors | 93 |
| Previewing Command and Option Operations..... | 94 |
| Fixing a Corrupt ZIP File | 95 |
| Use an Alternate Drive for PKZIP Temporary Files..... | 95 |
| Suppressing Screen Output..... | 96 |
| Setting Internal Attributes | 96 |
| Encoding an Archive to Another Type..... | 97 |
| Removing an Intermediate Archive | 97 |
| Generate a List File | 98 |
| 7 CHANGING DEFAULTS FOR COMMANDS AND OPTIONS | 99 |
| Viewing Configuration Settings | 99 |
| How Default Settings Work..... | 100 |
| Filter Options | 101 |
| Changing a Default Value | 102 |
| Changing Defaults for Filter Options..... | 102 |
| Changing Defaults for Compression Method..... | 102 |
| Using the Options Dialog to Change Defaults | 103 |
| Resetting to Original Defaults..... | 104 |
| Resetting Individual Defaults | 104 |
| Resetting All Defaults..... | 105 |
| Using an Alternate Configuration File..... | 105 |
| Creating an Alternate Configuration File..... | 105 |
| Using an Alternate Configuration File | 105 |
| 8 COMMAND CHARACTERISTICS..... | 107 |
| Changing Date and Time Environment Variables | 107 |
| Changing the List Character for List Files | 107 |
| Changing the Command/Option Character | 108 |

| | | |
|----------|--|------------|
| A | REFERENCE TO COMMANDS AND OPTIONS | 109 |
| B | ERROR AND WARNING MESSAGES..... | 162 |
| | Error Messages..... | 162 |
| | Warning Messages | 167 |
| C | FREQUENTLY ASKED QUESTIONS | 173 |
| D | HOW PKZIP WORKS..... | 176 |
| | Two Processes | 176 |
| | Compression..... | 176 |
| | Information Content | 176 |
| | Binary Data Representation..... | 177 |
| | Speed vs. Size | 179 |
| | Archiving | 179 |
| | How PKZIP builds a .ZIP File..... | 180 |
| | CRC | 182 |
| | Deleting Files from a .ZIP File | 182 |
| | Adding to an Existing .ZIP File..... | 183 |

1

The Basics

Welcome to *PKZIP Command Line* and *SecureZIP for Windows Command Line*. *PKZIP Command Line* and *SecureZIP for Windows Command Line* provide a command-line interface to PKZIP/SecureZIP for use in creating scripts and batch files. With *Command Line*, you execute PKZIP/SecureZIP commands and options by entering them at a character-based command prompt and running the resulting command line.

Command Line supports the full range of compression and archiving features of the graphical *PKZIP for Windows Desktop* and *SecureZIP for Windows Desktop*. *SecureZIP Command Line* also supports both traditional ZIP encryption and strong encryption using digital certificates. Both PKZIP and *SecureZIP Command Line* decrypt files encrypted with any other version of SecureZIP or PKZIP and authenticates digital signatures attached to archives and archived files.

This chapter will get you quickly up and running. After a brief overview of the manual and basic PKZIP concepts, you'll learn how to create ZIP archives and extract (unzip) files from archives. After covering the basic commands, you can get a taste of the power contained within PKZIP command options.

About This Manual

This manual describes how to use both PKZIP and *SecureZIP Command Line*. *SecureZIP Command Line* contains a superset of the commands and options of *PKZIP Command Line*. In general, references to PKZIP in the text apply equally to SecureZIP. If a feature has special requirements or comes only with special licensing, this is noted in the text.

The chapters group related commands and options and describe how to use them. This chapter provides an overview of basic program features. See in particular the section "Understanding Commands and Options" for an explanation of how commands and options work.

You can customize the default behavior of most commands and options. Chapter 7 describes how.

Appendix A contains a complete reference to the commands and options of the program. Experienced users may find that this appendix contains most of the information they need.

Conventions in This Guide

Most commands and options discussed in the following chapters work on all platforms that PKZIP supports. The cases are noted where a command or option is specific to a platform or operating system.

The name of a command or option appears by itself in *bold italic* font immediately under the main heading of the section where the command or option is discussed. In sections devoted to a particular sub-option, or value, of a command or option, the command or option is followed by an *equals* sign (=) and the name of the sub-option—for example, ***extract=all***.

An Overview of What PKZIP Does

PKZIP was developed to handle two basic tasks: It collects (adds) files into a container called an archive, and it pulls out (extracts) files from archives to restore them to their original state. The PKZIP ***add*** command is used to add files, and the ***extract*** command extracts them. These are the two most important PKZIP commands.

When PKZIP adds files to a specified archive, it creates the archive if it does not already exist. Generally, PKZIP compresses the added files so that they take less space, and it can also encrypt them so that they cannot be read by anyone who lacks the means to decrypt them.

As the creator of an archive, you control how its files are to be decrypted and by whom. You can encrypt files using a passphrase, such that the passphrase is required to decrypt them, or, if you have SecureZIP, you can use digital certificates to encrypt them such that only designated recipients can decrypt. SecureZIP also enables you to digitally sign files that you add to an archive, and the archive itself. A digital signature assures that the files really come from you.

Compression, encryption, and signing are done when you add files. When you extract files, PKZIP decrypts the files, decompresses them, and validates any digital signatures.

Most PKZIP options relate to the two main operations of adding and extracting files and are for optional use when you do one of those things. For example, besides the options to encrypt or sign files, there are options for picking the files that you want to compress or encrypt and options for how you want to compress or encrypt them. Commands are also available for managing archives—for example, for testing their integrity and viewing their contents.

Supported Archive Types

An *archive* is a kind of file that can contain other files. Several types of archive files exist. Some can contain only one file, some can contain multiple files, and there can be other differences as well. A ZIP archive can contain multiple compressed files. This is the kind of archive that PKZIP creates by default and is the kind that you will probably use most often. Encryption and digital signing are supported only for ZIP and OpenPGP archives.

PKZIP enables you to create and extract from many other archive types besides ZIP. You do not need to do anything special to use PKZIP with one of these other archive types. PKZIP can tell what type an archive is and will just go ahead and extract its files. If you want to create a new, non-ZIP archive, there are two ways to tell PKZIP what type of archive to create:

- Specify a name for the archive file that uses the file name extension commonly associated with that archive type
- Use the *archivetype* option to specify the type of archive that you want

The following table lists the types of archives that PKZIP can create or extract from and the file name extensions customarily associated with these types. For some archive types, PKZIP can do extractions but cannot create new archives of that type.

| <i>Archive type</i> | <i>PKZIP can create/extract</i> | <i>Usual file name extensions</i> |
|-----------------------------|---|-----------------------------------|
| <i>7Zip</i> | Extract only | .7z |
| <i>ARJ</i> | Extract only | .arj |
| <i>BinHex</i> | Extract only | .hqx |
| <i>BZIP2</i> | Create and extract | .bz2 |
| <i>CAB</i> | Extract only (Not supported on UNIX) | .cab |
| <i>CDR</i> | Extract only | .cdr |
| <i>compress (UNIX, LZW)</i> | Extract only | .Z |
| <i>GZIP</i> | Create and extract | .gz |
| <i>IMG</i> | Extract only | .img |
| <i>ISO</i> | Extract only | .iso |
| <i>JAR</i> | Create and extract | .jar, .ear, .war |
| <i>LZH</i> | Extract only | .lzh |
| <i>OpenPGP</i> | Create and extract | .pgp, .gpg |
| <i>RAR</i> | Extract only | .rar |
| <i>TAR</i> | Create and extract | .tar |
| <i>UUEncoded</i> | Create and extract | .uue |
| <i>XXEncoded</i> | Create and extract | .xxe |
| <i>ZIP</i> | Create and extract | .zip, .zipx |

Your Work Environment: The Command Line

In PKZIP Command Line, your work area is a character-based command line, or shell. You enter a command by typing the command on the command line; to execute the command, you press **Enter**.

To display a command line prompt in Windows, do one of the following:

- Choose Command Prompt from the list of programs in the Start menu

- Choose **Run** from the Start menu, enter `cmd` in the field, and choose **OK**.

Entering Commands

The syntax for commands entered on the command line is shown below. Brackets set off elements that are optional (Do not type the brackets.). Note that both PKZIP and SecureZIP Command Line use the same program name, `pkzipc`, as shown below.

```
pkzipc [command] [options] zipfile [@list] [files...]
```

Examples:

| <i>To do this</i> | <i>Command line</i> |
|---|---|
| <i>Add specified files to an archive</i> | <code>pkzipc -add zipfile.zip addfile.txt addfile2.doc</code> |
| <i>Add to an archive all files in current directory</i> | <code>pkzipc -add zipfile.zip</code> or: <code>pkzipc -add zipfile.zip *</code> |
| <i>Add to an archive all files in a specified directory</i> | <code>pkzipc -add zipfile.zip subdir*</code> |
| <i>Add files with the fast compression option</i> | <code>pkzipc -add -fast zipfile.zip</code> |
| <i>View list of files in archive</i> | <code>pkzipc zipfile.zip</code> |
| <i>View list of files whose names begin with "f" in archive</i> | <code>pkzipc zipfile.zip f*</code> |
| <i>Extract all files from an archive</i> | <code>pkzipc -extract zipfile.zip</code> |
| <i>Extract specified files from an archive</i> | <code>pkzipc -extract zipfile.zip readme.txt mystuff.doc</code> |

A PKZIP command line has these main elements:

- **The name of the program executable**—*pkzipc*. This command runs PKZIP and must appear first.
- **A PKZIP command** for the main task you want PKZIP to do—for example, add files to an archive. Precede the command with a hyphen: `-add`
- **Any PKZIP options** that you want to use. For example, when adding files to an archive, you can use the *maximum* option to have PKZIP take a little extra time to compress them as much as possible. You can include zero or more options. Precede each with a hyphen: `-maximum`
- **The name of an archive file**, such as a ZIP file, to create or operate on.
- **The names of files** to operate on—for example, to add to an archive, to act on a file in an archive (for example, to delete it), or to extract from an archive. Alternatively, you can give a file name pattern such as `*.doc` to specify these files, or the name of a file that contains a list of such files.

The name of the archive file must precede any other file names or file name patterns.

To reference multiple file names and/or patterns to operate on, separate the names with spaces.

- The pathname of a destination folder to extract to. PKZIP extracts to the current folder by default. To extract to a different folder, specify the folder's pathname.

Note: When identifying a pathname that includes a space, always put the pathname in quotation marks. For example, if you are archiving all the files in the Important Documents folder in Windows, type the following:

```
pkzipc -add zipfile.zip "Important Documents\**"
```

The only elements that are required in any command line are the name of the executable *pkzipc* and a PKZIP command. Other elements may be required depending on the particular commands or options used.

The order of appearance of the elements is not important except that:

- *pkzipc* must appear at the beginning of the command line
- The name of an archive file, if given, must appear before the name of any other file or folder

Creating a New Archive and Adding Files

Use the *add* command to add files to a new or existing archive.

For example, to add a file called `test.txt` to an archive file called `temp.zip`, use a command line like the following:

```
pkzipc -add temp.zip test.txt
```

If the archive does not already exist, PKZIP creates it.

You can optionally encrypt files when you add them. See “Encrypting Files That You Add to an Archive” in Chapter 3.

The following sections describe several ways to add files and how to display a listing of the files an archive contains.

Archive File Naming Conventions

Conventionally, archive files are named with a file name extension (the last part of the name, after the dot) that indicates the kind of archive. Thus a .ZIP archive generally has a name of the form *myarchive.zip*, where the file name extension is *.zip*. A BZIP2 archive generally has a file name extension of *.bz2*.

PKZIP can both create and extract from a variety of archive types—including BZIP2. Because the file name extension is generally a good guide to the type of archive, PKZIP can use this information to determine what sort of archive you want to create. Here are the rules PKZIP uses to determine the type of archive to create:

- If you specify an archive name with an extension—for example, *myarchive.zip* or *myarchive.bz2*, or *myarchive.exe*, PKZIP creates an archive of that name. Also, by default, PKZIP uses the file extension to select the type of compression to use. For example,

```
pkzipc -add myarchive.zip
```

results in a ZIP-format archive containing files compressed using standard ZIP-style compression (that is, using the Deflate compression algorithm). Alternatively, the following command line creates a BZIP2 archive. A BZIP2 archive is created using the BZIP2 compression algorithm and can contain only a single file.

```
pkzipc -add myarchive.bz2 myfile.doc
```

- If you specify an archive name with *no* file extension, by default PKZIP creates a ZIP archive and adds a *.zip* extension to its name. For example:

```
pkzipc -add myarchive
```

produces a ZIP archive called *myarchive.zip*.

Note: The *archivetype* option lets you explicitly tell PKZIP the type of archive you want to create. See “Compressing Files to a Specified Type of Archive” in Chapter 3.

- If you specify an archive name that has no file extension but does have a *trailing dot*—that is, a dot as the last character in the file name: for example, “*filename.*”—PKZIP does *not* append an extension to the file name. For example:

```
pkzipc -add myarchive.
```

produces (by default) a ZIP archive called *myarchive* without an extension.

Note: Systems that do not support more than one “dot” in a file name suppress the extension if any dot is present in the file name, even if it is not a trailing dot.

Note: The *noarchiveextension* option suppresses automatic adding of a file name extension on all systems.

Adding a Single File

To add a single file to an archive, use the add command and list on the command line the name of the archive and the name of the file to add. For example:

```
pkzipc -add test.zip red.txt
```

The command line adds file *red.txt*, in the current directory, to archive *test.zip*. Archive *test.zip* is created (in the current directory) if it does not already exist, or it is updated if it does exist.

The original of the added file `red.txt` still remains in the current directory. Adding a file to an archive only compresses and adds a *copy* (unless you use the `move` option to delete the original).

Adding Multiple Files

You can specify multiple files to add either by explicitly naming the files or by using wildcard characters in a file name pattern.

Specifying Multiple Files by Name

To specify multiple files by name, list them on the command line, separated by spaces, after the name of the archive:

```
pkzipc -add test.zip green.doc blue.fil purple.txt
```

Specifying File Names that Match a Pattern

You can use file name patterns to specify, for example, all files whose names begin with `p`, or all `.txt` files. A file name pattern picks out all files whose names match the pattern.

You can use these wildcard characters in file name patterns:

| Wildcard character | Matches |
|--------------------|------------------------------|
| Asterisk (*) | Zero or more characters |
| Question mark (?) | Zero or one single character |

For example, the following command line adds all files that have a particular file name extension (such as `.txt`):

```
pkzipc -add test.zip *.txt *.doc
```

The pattern `*.htm?` in the command line below matches all files that end in `.htm` or `.html`:

```
pkzipc -add test.zip *.htm?
```

Consult the documentation for your operating system to learn more about using wildcards.

Adding All Files in the Current Directory

If you want to add all files in the current directory, you do not need to specify any files to add. Just use the `add` command with the name of the target archive:

```
pkzipc -add test.zip
```

This shorthand works only for adding all files in the current directory. To add all files in some other directory, you must use wildcards (or specify the files).

For example, both of the following command lines do the same thing: they add all files in the `samples` directory:

```
pkzipc -add test.zip samples\*
pkzipc -add test.zip samples\*.*
```

Adding All Files in a Different Directory

To add files in a directory other than the current directory, specify the path to the files. You can use either an absolute path or a path relative to the current directory.

For example, these command lines use an absolute path to specify files to add:

```
pkzipc -add test.zip F:\sales_reports\*.xls
pkzipc -add test.zip "\Documents and Settings\john_d\My
Documents\samples\*.txt"
```

Enclose the path in quotes, as shown above, if it contains spaces.

These command lines use a relative path to specify files to add:

```
pkzipc -add test.zip samples\sales_reports\*.xls
pkzipc -add test.zip ..\records\jobs\*.doc
```

Working with an Archive in a Different Directory

If the target archive is not in the current directory, specify its location in the same way that you specify the location of files to add: include the path in the command line. You can use either an absolute or relative path.

```
pkzipc -add F:\sales_reports\test.zip *.xls
pkzipc -add samples\test.zip sales_reports\*.xls
```

PKZIP still assumes that a relative path to files to add starts from the current directory even if the target archive is somewhere else. How you specify the location of the files is not affected by the location of the archive.

If a path contains spaces, enclose it in quotes.

Moving Files into an Archive

Normally, after you add files to an archive, PKZIP leaves the original files on your hard drive. If you would like PKZIP to delete the original files after adding copies to an archive, you can include the **move** option in the command line when you add the files.

```
pkzipc -add -move confidential.zip sales*.xls
```

The **move** option is useful if you want to remove files that you no longer expect to use or if you do not want to leave behind unencrypted copies of files that you have placed in an encrypted archive.

CAUTION: Be sure to keep backups of your important files. If you move your only copy of a file into an archive, and the archive becomes lost or damaged, you may be unable to recover your file.

For information on working with PKZIP options, see the section “Understanding Commands and Options” later in this chapter.

Viewing Files in an Archive

The **view** command produces a list of the files in an archive and various pieces of information about the files. You can use the command to verify that files were added as expected or simply to find out what files an archive contains. It is also useful to see what path information is saved with a file. Path information is saved as part of the file name and so must be taken into account when you reference the file to extract it.

```
pkzipc -view myfiles.zip
```

The display generated by the **view** command looks like this:

```

Length Method      Size  Ratio   Date   Time   CRC-32  Attr   Name
-----
  0B Stored         0B   0.0%  4/4/2006  7:25p 00000000 ---wD  orderStatus_fi
les/
```



```

3557B DeflatN 3496B 1.8% 4/4/2006 7:24p 23ce6c93 -a-w- orderStatus_fi
les/bw_logo.gif
1653B DeflatN 847B 48.8% 2/9/2006 11:06a 891d9c90 -a-w- caroline.txt
71B DeflatN 66B 7.1% 1/27/2006 11:41a fa66929c -a-w- dummy_list.txt

420B DeflatN 128B 69.6% 3/10/2006 6:23p 4b63fc2a -a-w- filelist.txt
420B DeflatN 128B 69.6% 3/10/2006 6:23p 4b63fc2a -a-w- filelist2.txt
420B DeflatN 128B 69.6% 3/10/2006 6:23p 4b63fc2a -a-w- filelist3.txt
420B DeflatN 128B 69.6% 3/10/2006 6:23p 4b63fc2a -a-w- filelist4.txt
308B DeflatN 122B 60.4% 5/10/2005 3:14p 5f177b65 -a-w- files.txt
24B DeflatN 16B 33.4% 1/24/2006 2:27p f22154bb -a-w- mylist.txt
7915B DeflatN 1701B 78.6% 10/27/2005 12:08p 7b38176a -a-w- shared.txt
1463B DeflatN 816B 44.3% 1/9/2006 6:54p 2ef75758 -a-w- verisign.txt
878B DeflatN 432B 50.8% 8/26/2005 10:40a d1c700e7 -a-w- What's New.txt

-----
17KB 8008B 54.4% -----
13

```

The listing above was generated from a Windows command line.

For more information on the `view` command, see “Viewing the Contents of a ZIP File” in Chapter 6.

See Chapter 0 for information on other options you can use when adding files, including options to set the level of compression, add encryption, and so on.

Extracting Files from an Archive

To get a copy of a file out of an archive in its original form so that you can use it again, use the `extract` command. Extracting decrypts the file if it was encrypted, decompresses it, and validates any digital signature attached when the file was added.

You can extract all the files in an archive, or just selected files. As with adding files, PKZIP gives you numerous options for picking files and for choosing how to extract them. See Chapter 4.

Extracting All Files

To extract *all* files in an archive, include in the command line just the `extract` command and the name of the archive.

```
pkzipc -extract temp.zip
```

The files are extracted to the current directory.

Extracting Some Files

To extract only a selection of files, additionally specify the files to extract. For example, the following command line extracts all `.txt` files in the archive into the current directory.

```
pkzipc -extract temp.zip *.txt
```

You can also extract multiple files by explicitly listing their pathnames, separated by a space:

```
pkzipc -extract temp.zip green.doc blue.fil purple.txt
```

How you identify files in an archive depends on the path information that was archived with them. In an archive, path information is treated as part of a file name for purposes of identification. (Use the `view` command to see any path information saved with files.) For example, if you want to extract file `august.xls`, and the pathname of the file in the archive is `records\august.xls`, either of the following

command lines will extract the file. The command line that contains the `*` wildcard character also extracts all other `.xls` files whose pathnames start with `r`.

```
pkzipc -extract temp.zip records\august.xls
pkzipc -extract temp.zip r*.xls
```

Extracting Files to a Different Directory

By default, files are extracted to the current directory. To extract files to a different location, specify a path. For example, the following command line uses the two-dots (`..`) notation to specify a path to the parent of the current directory, one level up.

```
pkzipc -extract temp.zip *.txt ..
```

A destination pathname can occur in the command line anywhere after (to the right of) the name of the archive. For example, the following command line extracts all files in `data.zip` to the `january` subdirectory of the current directory:

```
pkzipc -extract data.zip january
```

To create a `january` subdirectory if one does not already exist, append a backslash (`\`):

```
pkzipc -extract data.zip january\
```

A folder name can appear before or after names of files to be extracted. Both of the following command lines extract `report.xls` to `january`:

```
pkzipc -extract data.zip report.xls january
pkzipc -extract data.zip january report.xls
```

PKZIP evaluates file or folder possibilities in the order they appear, from left to right, after the name of the archive. The first one found that is the name of a folder determines the destination folder.

Extracting New and Newer Files

By default, the `extract` command extracts all files if you do not specify particular files. You can also configure the `extract` command to extract only files that are newer versions of files already in the target directory, or only files that are newer versions or do not already exist in the directory.

For example, the following command line uses the `update` sub-option of the `extract` command to tell PKZIP to extract only files that are newer versions or do not already exist in the directory:

```
pkzipc -extract=update temp.zip
```

Sub-options are explained in the section “Commands and Options with Values,” later in this chapter.

Using Filters When Selecting Files

You can use various criteria to identify a specified particular set of files to add or extract, so that you only select the subset of files that meets the filter criterion.

For example, the command line below specifies all text files to add, but uses the filter option `after` to add a constraint; namely, that a file must also have been modified after the specified date (`mmdyyy`). As a result, only those text files that meet the additional requirement imposed by the `after` option are added.

```
pkzipc -add -after=03152011 myfiles.zip *.txt
```

All the filter options described in this section work with both *add* and *extract* commands.

Selecting Files by Date

before, after

The *before* option selects files that were modified before a specified date. The *after* option selects files that were modified on or after a specified date.

In the United States, enter dates in one of the following formats:

- mmddyy
- mmddyyyy

The order in which you enter the month, date, and year depends on your *locale* setting. For more information on the *locale* setting, see Chapter 8.

The following sample command line adds files dated before February 24, 2011:

```
pkzipc -add -before=02242011 test.zip
```

The command line below adds files dated February 24, 2011, or later:

```
pkzipc -add -after=02242011 test.zip
```

Selecting Files by Age

older, newer

The *older* and *newer* options select files that are older or newer than a specified age. You can list the age in days (the default), hours, minutes, or seconds using the abbreviations shown in the following table.

| <i>Time unit</i> | <i>Abbreviation</i> |
|-----------------------------------|---------------------|
| <i>Days</i> (<i>default</i>) | d (or nothing) |
| <i>Hours</i> | h |
| <i>Minutes</i> | m |
| <i>Seconds</i> | s |

For example, the following command lines each add files that are no more than five days old:

```
pkzipc -add -newer=5 test.zip *
pkzipc -add -newer=5d test.zip *
```

The command lines below add files that are older than five days:

```
pkzipc -add -older=5 test.zip *
pkzipc -add -older=5d test.zip *
```

The following command line uses both options to select files to extract:

```
pkzipc -extract -newer=10 -older=5 test.zip *
```

With a time unit of days, the interval (for example, five days) is measured from the beginning of the current day. So, for example, if it is currently 3:34 p.m. on June 15, setting *newer* or *older* to 5 sets the cutoff to 12:00 a.m. June 10. The *older* option gets files dated earlier than this; the *newer* option gets files dated on or after this.

With time units of hours, minutes, or seconds, the interval is measured from the current system time. So, for example, the following command line selects files modified within the last 48 hours:

```
pkzipc -add -newer=48h test.zip *
```

Selecting Files by Size

larger, smaller

The *larger* and *smaller* options select files that are larger than or equal to, or smaller than or equal to, a size specified in bytes.

The following command line adds files whose size is in the range 5000-7000 bytes, inclusive:

```
pkzipc -add -larger=5000 -smaller=7000 test.zip
```

Selecting Files to Include or Exclude

include

The *include* option has two uses:

- To specify a file name pattern to use by default when selecting files to add or extract
- To override, in the current command line, a configured default setting that excludes files from being selected

Ordinarily, to select files whose names match a pattern (for example, *.doc), simply specify the pattern on the command line:

```
pkzipc -add test.zip *.doc
pkzipc -extract test.zip *.doc
```

To include one or more file patterns automatically when selecting files, you can configure a default value for *include*. For example, if you want to automatically include all files with the extension of .doc when adding files, enter the following:

```
pkzipc -config -add -include="*.doc"
```

This configured default causes a command line like the following to zip all .doc files in addition to the *.txt files explicitly specified.

```
pkzipc -add test.zip *.txt
```

You can also use *include* to override a default setting of the *exclude* option.

For example, if you have configured PKZIP to exclude *.txt files by default when adding, you can include such files in a particular case with the command line below:

```
pkzipc -add -include="*.txt" test.zip
```

If you do not need to override a default configuration setting, you do not need to specify the *include* option in your command: the file pattern by itself is enough.

For more information on modifying default configuration values, see Chapter 7.

exclude

The ***exclude*** option has two uses:

- To specify a file name pattern or list file to use to exclude files by default when selecting files to add or extract
- To override, in the current command line, a configured default setting that includes files

To exclude one or more file patterns automatically when selecting files, you can configure a default value for ***exclude***. For example, if you want to automatically exclude all files with the extension of `.doc` when adding files, enter the following:

```
pkzipc -configuration -add -exclude="*.doc"
```

The command line below has the same effect but abbreviates the ***configuration*** option:

```
pkzipc -config -add -exclude="*.doc"
```

The configured default value for ***exclude*** causes a command line like the following to zip all files except `.doc` files.

```
pkzipc -add test.zip *.*
```

To exclude a list of files, specify the list file as the value of the ***exclude*** option:

```
pkzipc -add -exclude=@lst.txt test.zip
```

You can also use ***exclude*** to override a default setting of the ***include*** option. For example, if you have configured PKZIP to include `*.txt` files by default, you can exclude them in a particular case with the command line below:

```
pkzipc -add -exclude="*.txt" test.zip
```

For more information on modifying default configuration values, see Chapter 7.

Understanding Commands and Options

A PKZIP command line includes a command and can also include options that affect how the command is done or specify things to be done in conjunction with it. Many commands and options also have sub-options that determine how the command or option behaves.

Difference between a Command and Option

A command tells PKZIP *what* to do; an option tells PKZIP to do the main task in a particular way or to do some additional task in the course of doing the main task.

For example, the ***add*** command tells PKZIP to add files to an archive. You can use the ***maximum*** option with the ***add*** command to tell PKZIP to use maximum compression when adding the files. If you want to delete the original files after they are added, you can include the ***move*** option too:

```
pkzipc -add -maximum -move myarchive.zip *.doc
```

A command line must always contain a command; it can contain any number of options. A command stands alone in a command line, without requiring (or permitting) any other command. For this reason, it is sometimes referred to as a *standalone* to indicate that it is not an option. An option can be used only with a command.

A few options bend the rules in that they can be used either as options or as commands. These include *comment*, *header*, *sfx*, *sign*, and some of the *mail...* options. For example, *comment* prompts you for a comment to attach to an archive. This option can be used with the *add* command to attach a comment to a new archive, or it can be used by itself to attach a comment to an archive that already exists.

Including an Option in Your Command Line

To use an option, prefix it with a hyphen and insert it in the PKZIP command line after the main command.

For example, the following command line uses the *maximum* option with the *add* command. This option tells PKZIP to use maximum compression:

```
pkzipc -add -maximum test.zip white.doc
```

The following example uses the *overwrite* option to turn off the usual prompting to overwrite files with the same names as files to be extracted. The command line directs that extracted files simply overwrite any files that have the same names, without prompting:

```
pkzipc -extract -overwrite test.zip
```

Abbreviating Commands and Options

In a command line, you can abbreviate commands and options by leaving off letters at the end as long as you give enough of the name for PKZIP to know what command or option you mean.

For example, you can abbreviate the name of the *maximum* option to *max*, as in the command line below, because no other option name starts with those letters.

```
pkzipc -add -max test.zip white.doc
```

The command line below abbreviates the name of the *extract* command to *ext*:

```
pkzipc -ext test.zip
```

It's good practice to avoid abbreviating commands and options when writing scripts, as PKWARE adds new features with each new version. Using full commands ensures that your scripts will work regardless of what other commands may be introduced.

Using Multiple Options

To use multiple options in the same command line, separate them by spaces.

For example, the following command line includes both the *maximum* and *comment* options. These tell PKZIP to use maximum compression and to prompt you for a comment for each newly added file:

```
pkzipc -add -maximum -comment test.zip *.doc
```

The order in which options appear is not important.

Not all options can be used with all commands. For example, you cannot use *maximum* with the *extract* command. Appendix A lists the commands with which each option can be used.

Commands and Options with Values

Some commands and options have different possible values, called sub-options, that let you customize how the command or option behaves. For example, the *level* option enables you to specify how much compression you want to use (more compression takes longer). When you use *level*, you specify a value for a particular level of compression. For example:

```
pkzipc -add -level=9 myarchive.zip
```

To specify a sub-option or value with a command or option, attach it to the command/option with an equal sign, as in the last example.

Commands as well as options can have sub-options. For example, you can use the *add* command to add all selected files to an archive, or to add only files that are newer versions of files that the archive already contains. You indicate how you want *add* to work by specifying a sub-option. To have the command add only newer versions of files that the archive already contains, use the command with the *freshen* sub-option:

```
pkzipc -add=freshen myarchive.zip *.*
```

Most commands and options that have multiple possible predefined values or sub-options use one of the values as a default. Some options are disabled by default, but if an option has a default value, that value is implicitly used in any command line that does not explicitly list the option.

For example, the *level* option has a default value of 5 (normal compression). The following command line does not explicitly include the *level* option, but because the option is not disabled and has a default value, the command line applies the option at its default value and uses normal compression:

```
pkzipc -add myarchive.zip *.*
```

PKZIP uses the default value for a *command* (as opposed to an option) whenever the command is used with no sub-option specified. In the preceding example, PKZIP uses the default value for *add*.

You can replace original default settings with your own by using the *configuration* command. See Chapter 7.

For a list of all commands and options together with their sub-options, see Appendix A.

Using Strong Encryption

PKZIP allows you to use either of two kinds of encryption to encrypt ZIP archives: the older, traditional PKZIP encryption, or strong encryption. Strong encryption is much more secure than traditional PKZIP encryption.

PKZIP and SecureZIP v14 add new support for encrypting and decrypting files using the OpenPGP (RFC 4880) standard. You can open and decrypt any OpenPGP files you receive with PKZIP and SecureZIP. Create OpenPGP-based archives and use its encryption on any file (not just ZIP archives) with SecureZIP and PKZIP.

Two methods exist to ensure that encrypted files are only opened by the right people: Symmetric keys (Passphrases) or public/private key pairs. The following table shows the methods each encryption type supports.

| <i>Encryption type</i> | <i>Passphrases</i> | <i>Public/Private Keys /Certificates</i> |
|------------------------------|--------------------|--|
| PKZIP Traditional Encryption | • | |
| PKZIP Strong Encryption | • | • |
| OpenPGP Encryption | • | • (SecureZIP only) |

Traditional PKZIP encryption is passphrase-based and is applied using the ***passphrase*** option. Strong encryption can be done with either a passphrase or a digital certificate. When you encrypt using a digital certificate, only the owner of the certificate—called a *recipient*—can decrypt.

You use the ***passphrase*** option to apply either traditional or strong passphrase-based encryption, including OpenPGP.

To do certificate-based strong encryption, you use the ***recipient*** option to specify the owners of the certificates for whom you want to encrypt. You must also have a copy of each recipient's certificate that contains the certificate's public key.

With both certificate- and passphrase-based strong encryption, you use the ***cryptalgorithm*** option to specify an encryption algorithm and key length (for example, AES, 256 bits).

You need version 6.0 or later of PKZIP (or ZIP Reader) to decrypt archives that were strongly encrypted using PKZIP. You may need SecureZIP to strongly encrypt archives yourself. To learn much more about encryption in PKZIP, see “Encrypting Files That You Add to an Archive” in Chapter 0, “Extracting Passphrase-Protected Files” in Chapter 4, and “Working with OpenPGP Files” in Chapter 5.

2

Getting Started

Welcome to PKZIP/SecureZIP Command Line. PKZIP Command Line and SecureZIP Command Line provide a command-line interface to PKZIP and SecureZIP that enables you to access the functions of these two powerful data security and data archiving programs in scripts and batch files.

SecureZIP Command Line is an enhanced version of PKZIP Command Line. Both programs enable you to create and manage ZIP files and archives of other types, and both programs enable you to decrypt archives encrypted with either program. But SecureZIP Command Line provides additional features—most notably, commands and options for using digital certificates to do strong encryption and attach digital signatures.

This table and the following sections describe the additional features included with SecureZIP Command Line that are not in PKZIP Command Line.

| <i>Feature</i> | <i>PKZIP</i> | <i>SecureZIP</i> |
|---|--------------|------------------|
| Large file size support | X | X |
| Very large archive support | X | X |
| Self-extracting files for end-users and other platforms | X | X |
| Decryption of PKI public-key encrypted ZIP archives | X | X |
| Attaching digital signatures to archives | | X |
| Strong passphrase-based AES and 3DES data file protection | X | X |
| Strong encryption using a digital certificate instead of a passphrase | | X |
| Strong, certificate-based file name encryption | | X |
| Creating OpenPGP (RFC 4880) encrypted files | X | X |
| Opening OpenPGP files | X | X |
| Add digital timestamp from secure Time Stamp Authority | | X |

| <i>Feature</i> | <i>PKZIP</i> | <i>SecureZIP</i> |
|---|--------------|------------------|
| Error reporting for both attended and unattended operations | X | X |
| Application Integration | X | X |
| Preserving Zone Identifier information added by Internet Explorer | X | X |

Learning More and Getting Help

This manual is not the only way to learn about PKZIP and SecureZIP. You can find additional information inside the program itself, and on the World Wide Web.

Using Help

PKZIP provides a help system for the PKZIP commands and options. The help system describes syntax and shows sample command lines.

Access the help system directly from the command line:

- At the command prompt, type the following and press ENTER:

```
pkzipc -help
```

A screen with PKZIP version and usage information appears. You can get help for any PKZIP command or option from here.

- To bypass the command/option menu and go directly to a help file for a particular command or option, type the **help** command followed by an equal sign (=) and the command or option for which you want information.

For example, to access online help for the **add** command, type the following at the command prompt and press ENTER:

```
pkzipc -help=add
```

The help information for the **add** command appears.

Getting Version Information

version

To list the version of PKZIP that you are using, use the **version** command:

```
pkzipc -version
```

This command line outputs two lines like the following after the usual header information:

```
Program File Version (pkzipc): 12. 50. 1087
Product Version: 12. 50. 0005
```

The first line lists major, minor, and step version numbers of the *program*:

```
Program File Version (pkzipc): <major>. <minor>. <step>
```

The second line lists the major and minor version numbers and the build number of the *product*.

Product Version: <major>. <minor>. <build>

Major and minor version numbers of the program are always the same as those for the product.

In addition to producing this display output, the *version* command returns a version number as a value to the shell. The version number returns as a positive integer value less than 256. This value is only returned to the shell and is not displayed in normal output. It can be used to verify PKZIP version numbers in a .BAT file or shell script.

Sub-options of the *version* command (described in the following table) determine which version number is returned. The major version number is returned by default.

| <i>Sub-Option</i> | <i>PKZIP Returns</i> | <i>For example</i> |
|-------------------|--|--|
| <i>major</i> | The major release number. For example, if the version number is 12.10.1054, the value returned is 12. This is the default return. | pkzipc -version pkzipc -version=major |
| <i>minor</i> | The minor number of the release. For example, if the version number is 12.10.1054, the value returned is 10. | pkzipc -version=minor |
| <i>step</i> | The step or patch value (minus 1000 if ≥ 1000). For example, if the program version is 12.10.1054, the value returned is 54. | pkzipc -version=step |
| <i>product</i> | The build number of the product. For example, if the product version is 12.10.0003, the value returned is 3. | pkzipc -version=product |

Technical Support

For support, visit our Web site at:

www.pkware.com/support

Working With Your License

Entering License Keys

Ordinarily, you enter license keys during installation. If you need to enter a license key after installation—for an add-on module, for example—use the *enterlicensekey* command.

To enter a license key:

1. At the command prompt, type the following and press ENTER:

```
pkzipc -enterlicensekey
```

PKZIP prompts you for a product license key.

2. Enter a product license key and press ENTER.

Repeat these steps for each license key.

Setting PKZIP in the Path

The installation puts PKZIP on your system's search path so that you can access the program from any directory without specifying a path. However, if for any reason you need to specify the path yourself, you can.

The search path in Windows is normally specified in the system's Environment Variables. To add the PKZIP installation directory to your search path, follow the steps below (some items may have different labels depending on your version of Windows).

1. Close any open Command Prompt windows.
2. Select Settings | Control Panel from the Start Menu.
3. In the Control Panel, double click the System icon (or click Advanced System Settings). The System (Properties) dialog appears.
4. Click the Advanced tab and then click the Environment Variables button.
5. Select the PATH variable in the System (Environment) Variables or User (Environment) Variables boxes. If you are unable to locate the PATH variable, enter the following in the Variable box:

`path`

6. In the Value box, enter (in quotes) the path to the folder where PKZIP is installed.

For example, assuming that PKZIP (`pkzipc.exe`) is installed in the default location, enter:

`c:\program files\pkware\pkzipc`

If necessary to separate the path from another path designation, precede your path with a semicolon.

7. Click the **Set** (or **OK**) button.
8. Click the **OK** button.

You may now access PKZIP from any directory without specifying a path. This change will take effect the next time you open a Command Prompt window to run PKZIP.

If necessary, consult your systems administrator for further information on setting the path environment variable.

3 Adding Files to an Archive

This chapter contains detailed information on the features and options available when you add files to an archive.

Default Values for Commands and Options

For each operation in this chapter, the command or option that represents that operation has a default value. The default value determines the way that the command or option is done when the command or option is used on the command line by itself, with no sub-option explicitly specified.

For example, the initial default value for the *add* command is *a11*, which causes the command to add all files. See Chapter 7 for information on how to change default settings.

Creating and Updating Archives

add

The *add* command adds files to an archive.

To add files to a new or existing archive, specify the name of the archive on the command line, then list one or more files to add. If the archive does not already exist, PKZIP creates it.

The command line below adds all *.txt* files in the current directory to *myarchive.zip*.

```
pkzipc -add myarchive.zip *.txt
```

Adding All Files in a Directory

You can choose to compress all files in a particular directory with a single command. To do this, you do not have to specify each file. Simply type *pkzipc -add*, and the name of your ZIP file, as shown below:

```
pkzipc -add test.zip
```

In the example above, all files in the current directory are compressed into the *test.zip* file. (To learn how to compress files that appear in subdirectories, see, "Compressing Files in Subdirectories" later in this chapter.)

You can also specify files from a different directory if you wish. For example, if you were in a parent directory to a directory called *temp* and you wanted to compress all the files in the *temp* directory, you could type the following:

```
pkzipc -add test.zip temp/*
```

The resulting *test.zip* file is stored in the current directory (the parent directory to the *temp* directory in our example).

Note: The *add* command adds all files in a specified directory to your archive file by default. You do not need to specify the *a11* sub-option with the *add* command to compress all files unless you have used the *configuration* command to modify the default setting for *add*.

For information on how to modify default values for commands and options, see Chapter 7.

Adding New and Modified Files

add=update

PKZIP allows you to specify that only new or modified files are added to an archive. When the *update* sub-option is used, dates on the files specified for archiving are compared against dates of files having the same name already present in the archive. A file is added only if no file with the same name is already in the archive or if the file to be added is newer.

The *update* sub-option can save time when you repeatedly archive the same files. The sub-option differs from the *freshen* sub-option in that it adds files which are not in the archive already.

To compress only updated files or files not already archived in a specific .ZIP file, use the *update* sub-option with the *add* option, as shown below:

```
pkzipc -add=update test.zip *.doc
```

In this example, a .ZIP file called test.zip is created in the current directory. All files in the current directory matching the file specification (*.doc) will be added or updated into the test.zip archive.

Adding Only Files That Have Changed

add=freshen

The *freshen* value allows you to compress only changed files that exist in the .ZIP file. No new files will be added to the .ZIP file. To update files that have changed, use the *freshen* value with the *add* option, as shown below:

```
pkzipc -add=freshen test.zip
```

The following command line abbreviates the value but has the same effect:

```
pkzipc -add=fre test.zip
```

If you only want to re-compress specific files, simply include those files in your command. For example, if you wanted to re-compress a file called resume.doc, you would type something like this:

```
pkzipc -add=freshen test.zip resume.doc
```

In the above example, only *resume.doc* will be re-compressed into the test.zip file. This assumes that the version of resume.doc being added is newer than the version of resume.doc that already exists in the .ZIP file.

Incremental Archiving

A file has various attributes, or items of information about it, such as its date. One such attribute is called the *archive* attribute. This attribute is set ON when a file is created or altered. A backup program that uses this attribute switches the attribute off when the file is backed up. By using the archive attribute to select files, you can get all (and only) files that are new or changed since the last backup. A backup that uses the attribute in this way is called an incremental backup.

add=incremental

If you wish to add files to a .ZIP file that have the archive attribute set and subsequently clear the archive attribute on the original files, use *add* with the *incremental* sub-option. If you wish to add files to a .ZIP file that have the archive

attribute set and *not* clear the archive attribute on those files, use *add* with the *-incremental* sub-option.

The *incremental* and *-incremental* sub-options can be very useful when backing up files. If, for example, the *incremental* sub-option is specified, only files with the archive attribute will be compressed, and the archive attribute will be set to OFF when the ZIP operation is complete for these files.

In the following command line example, PKZIP will add only those files to `test.zip` with the archive attribute set. Additionally PKZIP will clear the archive attribute on any of the source files that have been added to `test.zip`.

```
pkzipc -add=incremental test.zip
```

The next time you run this command, only those files that have the archive attribute set (new or updated files) will be added to the `test.zip` file.

add=archive

By using this option, you can create a complete backup of your disk, while clearing the archive attributes to make way for incremental archiving.

Incremental archiving makes use of the archive attribute to take only the files which have been modified since the last backup. For this process to work smoothly, you must first have a complete backup and clear the archive attribute for all files.

```
pkzipc -add=archive -dir f:backup.zip
```

This prepares the files set for future incremental backups. For future incremental backups, use

```
pkzipc -add=incremental test.zip
```

Use the *archive* sub-option only if you are doing a full backup of your disk to prepare for doing incremental backups.

Encrypting Files That You Add to an Archive

You can encrypt files when you add them to an archive. When you encrypt files, only people that you designate or who know a passphrase that you assign can decrypt and extract the files.

Depending on whether you have PKZIP or SecureZIP, you can encrypt using either traditional ZIP encryption or strong encryption. Strong encryption is far more secure than the older, traditional ZIP encryption, but people who want to decrypt your files are likely to need access to PKZIP. Other ZIP utilities generally cannot decrypt strongly encrypted files.

The *passphrase* and *recipient* options control encryption when you add files to an archive.

- With the *passphrase* option, you specify a passphrase to use to decrypt the files. The *passphrase* option is available in both PKZIP and SecureZIP. It is used to do both strong and traditional ZIP passphrase-based encryption.

A passphrase is just a password. It is called a passphrase in the program to emphasize that PKZIP and SecureZIP support passwords that can contain spaces and other non-alphanumeric symbols.

- With the *recipient* option, you specify a recipient list. A recipient list is a list of digital certificates that belong to people whom you want to allow to

decrypt. PKZIP automatically decrypts the files for the owners of the certificates when the owners extract the files. You will learn more about digital certificates in Chapter 5.

The *recipient* option is used only to do strong encryption and is available only in SecureZIP. Both PKZIP and SecureZIP can decrypt files encrypted with either kind of strong encryption (passphrase or recipient list).

When you use strong encryption, you also have the option to encrypt not only the contents but the names of files and folders that you add to an archive. When you encrypt file names, you essentially encrypt the archive itself: the archive cannot even be opened except by someone who can decrypt its contents.

Encrypting Files with a Passphrase

passphrase

Use the *passphrase* option (with the *add* command) to encrypt files so that users can use a passphrase to decrypt them. You can do either strong or traditional ZIP encryption with the *passphrase* option.

To include a passphrase on the command line, use the *passphrase* option and enter a passphrase of at least eight characters (preceded by an equal sign). For example (where the passphrase is *mypassphrase*):

```
pkzipc -add -passphrase=mypassphrase test.zip
```

Note: Passphrases are case sensitive.

For more security, you can enter your passphrase separately from the command line, at a prompt. This method prevents other users from learning your passphrase by reviewing previously entered PKZIP command lines.

To have PKZIP prompt for a passphrase, include the *passphrase* option in the command line but do not specify a passphrase. For example:

```
pkzipc -add -passphrase test.zip
```

When you press ENTER, a prompt like the following appears:

```
Passphrase?
```

Type your passphrase. The characters appear on your screen as asterisks. Press ENTER. PKZIP asks you to confirm the passphrase:

```
Re-enter passphrase for verification.
Passphrase?
```

Re-enter the passphrase and press ENTER. If your entry matches the original one, PKZIP proceeds and compresses the files. If the passphrases do not match, PKZIP prompts you again:

```
Passphrases don't match! Please try again.
Passphrase?
```

Another way to enter a passphrase is to point PKZIP to a text file that contains one. For example:

```
pkzipc -add -passphrase=@secret.txt test.zip
```

The file (*secret.txt* in the example) should contain just the passphrase, on a line by itself.

For best security, choose a passphrase that is not easy for someone to guess. Ideally, a passphrase should be at least eight characters long, should contain a mix

of numbers and upper- and lower-case letters, and should not be a word in the dictionary.

Note: Use a passphrase of no more than 245 characters for files to be decrypted using PKZIP or SecureZIP for z/OS on a mainframe, or PKZIP or SecureZIP for i5/OS on the AS/400, iSeries, or i5.

Specify an Encryption Method

listcryptalgorithms, cryptalgorithm

When you use strong encryption,) PKZIP gives you a choice of encryption algorithms to use. To list the available algorithms, use the *listcryptalgorithms* command.

```
pkzipc -listcryptalgorithms
```

The following output from *listcryptalgorithms* lists all supported algorithms:

```
AES, 256      AES (256-bit)
AES, 192      AES (192-bit)
AES, 128      AES (128-bit)
3DES, 168     3DES (168-bit)
```

Use the *cryptalgorithm* option to specify a particular algorithm. .

```
pkzipc -add -passphrase -cryptalgorithm=aes,128 test.zip
```

By default, *cryptalgorithm* specifies AES,256. If you do not use *cryptalgorithm* when encrypting with a passphrase, PKZIP applies traditional PKWARE encryption.

Encrypting Files with a Recipient List

recipient

Use the *recipient* option (with the *add* command) to strongly encrypt files and specify a recipient list. A recipient list is a list of digital certificates that belong to the people whom you want to allow to decrypt.

Note: The *recipient* option is available only with SecureZIP. You will learn more about digital certificates and recipient lists in Chapter 5.

To encrypt using a recipient list, you must have a digital certificate, containing a public key, for each intended recipient. Any recipient on the list—that is, any person whose system has access to the private key for that certificate—can decrypt and extract the files simply by using the *extract* command. No one else can decrypt (unless a passphrase was also specified).

If you use the *recipient* option together with the *passphrase* option, PKZIP decrypts automatically for listed recipients when they extract the files, and other people can decrypt if, and only if, they have the passphrase.

Note: Ordinarily, PKZIP decrypts automatically for anyone on a recipient list. However, if necessary, a recipient can tell PKZIP where to find a private key that is not in one of the usual places. See the *keyfile* and *keypassphrase* options.

Specifying Recipients

You can specify a list of recipients either by specifying each recipient individually on the command line, or by specifying a file that contains a recipient list.

Be sure to specify yourself as a recipient if you want to be able to use your own certificate to decrypt.

By default, SecureZIP searches for certificates for listed recipients only in the system's local certificate stores.

Use any of the following criteria to specify recipients:

| Criterion | To use | For example |
|----------------------|---|--|
| <i>Common name</i> | Specify, in quotes, the common name of the subject of the certificate (that is, the <i>cn</i> field in a string representation of a certificate); optionally, precede with: cn= By default, SecureZIP searches for recipients by common name unless another sub-option is used or the value appears to be an email address. | -recipient=cn="John Public" -recipient="John Public" |
| <i>Email address</i> | Specify the email address of the certificate (that is, the <i>e</i> field in a string representation of a certificate); optionally, precede with: e= | -recipient=e=john.public@xyz.com -recipient=john.public@xyz.com |

For example, if the common name of the subject is *John Q. Public*, you can specify that certificate as a recipient as follows:

```
pkzipc -add -recipient="John Q. Public" test.zip
```

You can specify multiple recipients by using the *recipient* option multiple times:

```
pkzipc -add -recipient="John Q. Public" -recipient="Mary  
Samplename" test.zip
```

You can also reference a recipient by email address:

```
pkzipc -add -recipient=john.public@nowhere.com test.zip  
pkzipc -add -recipient=e=john.public@nowhere.com test.zip
```

The prefix *e=* when using an email address is optional. SecureZIP automatically looks for an email address if the string contains an @ and a dot and looks like an email address.

Note that a certificate must contain an email address in order to be found by this method. Not all certificates embed an email address.

Specifying a File That Contains a Recipient List

PKZIP can extract a recipient list from these kinds of files:

- An ordinary text file that lists the common name of each recipient's certificate on a line by itself

To use the *recipient* option with a text file list of recipients as a sub-option, prefix the file name with the listfile character (@, by default):

```
pkzipc -add -recipient=@recipient_list_file.txt test.zip
```

- Key container files: These kinds of files contain one or more actual certificates, and conform to one of two standards. PKCS#7 files have the file name extensions `.p7b` and `.p7c` and do not contain private keys, only public ones. PKCS#12 files have the file name extensions `.pfx` and `.p12` and may contain private keys as well as public keys.

To use *recipient* to specify a key container file to define a recipient list, prefix the file name with a hash (#) character:

```
pkzipc -add -recipient=#recipient_list_file.p7b test.zip
```

The recipient list will contain the owners of all certificates included in the key container file.

Specifying an Encryption Method with a Recipient List

With the *passphrase* option, you can select either strong encryption or weaker, traditional ZIP encryption. The *recipient* option, however, always causes SecureZIP to use strong encryption. If you do not use the *cryptalgorithm* option to explicitly specify a strong encryption method with a recipient list, and no encryption method is configured for use by default, SecureZIP uses the first method listed in the output from the *listcryptalgorithm* command.

The *listcryptalgorithm* command and the *recipient* and *cryptalgorithm* options are available only in SecureZIP.

Encrypting with OpenPGP

You can also use OpenPGP keyrings to define a recipient list. You must first configure SecureZIP to enable OpenPGP on your system. See “Setting Up OpenPGP Keyrings” in Chapter 5.

When OpenPGP is enabled, and you use the *-recipient* command, SecureZIP will search for your keyring using your system’s default keyring directories and environment variables. If you store your public and private keyrings in a non-default location, you should set the `PGP_HOME_DIR` environment variable to identify the location. Consult your OpenPGP application’s documentation for instructions.

Do not use the *cryptalgorithm* option to explicitly specify a strong encryption method, as ZIP archives with PGP files must use the RSA encryption algorithm.

Encrypting File Names

cd

The *cd* option uses strong encryption and is available only with SecureZIP.

Someone who cannot decrypt the contents of an archive may still be able to infer sensitive information just from the unencrypted names of files and folders. To prevent this, you can encrypt the names of files (and folders) in addition to their contents. Encrypted file names can be viewed in the clear—that is, unencrypted—only when the archive is opened by an intended recipient if the archive was encrypted using a recipient list, or by someone who has the passphrase, if the archive was encrypted using a passphrase.

Use the *cd* option (stands for “archive *central directory*”) with the *add* command to encrypt file names. The *cd* option applies strong encryption to an archive’s central directory, where file names and virtually all other metadata about the archive are stored.

An archive that contains encrypted file names requires PKZIP or SecureZIP version 8.0 or later to open it.

The `cd` option has two sub-options:

| <i>Sub-Option</i> | <i>Effect</i> | <i>Example</i> |
|-------------------|--|--------------------------|
| <i>encrypt</i> | Encrypts file names and the archive's central directory. This is the default sub-option, used if you enter <code>-cd</code> and do not explicitly specify a sub-option. | <code>-cd=encrypt</code> |
| <i>normal</i> | Does not encrypt file names; produces a normal ZIP file. Use to override a configured default setting that would otherwise encrypt file names. | <code>-cd=normal</code> |

You must use strong encryption when you use the `cd` option. You can use either strong passphrase encryption or a recipient list (or both), but you must use one of the strong encryption methods. You cannot encrypt file names using traditional, passphrase encryption.

The following sample command line encrypts file names using a recipient list:

```
pkzipc -add -recipient="John Q. Public" -cd test.zip
```

The sample command line below encrypts file names using a passphrase. When you use the `cd` option with a passphrase, SecureZIP uses the default strong encryption algorithm (ordinarily AES 256) if you do not explicitly specify an algorithm.

```
pkzipc -add -passphrase=mysecret -cryptalgorithm=aes,256 -cd test.zip
```

Encrypting File Names in an Existing Archive

You can encrypt file names in either a new or an existing archive.

- If you add files to an archive that already contains files with unencrypted file names and specify `cd` to encrypt file names, SecureZIP encrypts the names of all files in the archive, not just names of newly added files.

If the archive contains files whose contents are already encrypted, SecureZIP decrypts these files and then re-encrypts them, and their names, using the currently specified encryption method (passphrase/recipient list) and algorithm.

If SecureZIP cannot decrypt the files, SecureZIP does not update the archive: no files are added, and file names are not encrypted.

- If you update an archive in which file names are encrypted, SecureZIP encrypts the newly added files and their names using the same passphrase or recipient list originally used to encrypt file names in the archive.

Encrypting Using Only FIPS-Approved Algorithms

fipsmode

“FIPS” is an abbreviation for “Federal Information Processing Standards,” a set of standards for information processing in federal agencies. The standards are

published by NIST (National Institute of Standards and Technology), a branch of the US government. The FIPS 140-2 standard defines security requirements for cryptographic modules and specifies the algorithms that federal agencies may use for cryptographic operations—encrypting, decrypting, signing, and authenticating digital signatures.

The *fipsmode* option restricts SecureZIP to using only algorithms that comply with the FIPS 140 standard to perform cryptographic operations.

With *fipsmode* on, SecureZIP exclusively uses FIPS-validated algorithms not only to encrypt but also to decrypt. If you try to decrypt a file that is encrypted using an algorithm that is not FIPS-validated, SecureZIP responds with an error or warning and does not decrypt it.

When applying or authenticating signatures, SecureZIP again uses only FIPS-validated hashing algorithms when the *fipsmode* option is on. If a signature was created using a hashing algorithm that is not FIPS-validated, SecureZIP shows a warning even if the signature is otherwise valid.

The *fipsmode* option is not compatible with the *204* option (which cannot create archives with strong encryption).

For the *fipsmode* option to work—that is, to actually result in FIPS-mode processing—a FIPS-validated cryptographic module must be installed on your system. On UNIX, SecureZIP supplies such a module itself. On Windows, however, it is the system administrator's responsibility to ensure that a version of the Microsoft CryptoAPI cryptographic module appropriate to the operating system is installed and that no non-FIPS-validated cryptographic providers (for example, a non-FIPS-validated smart card) are used.

For reference, see the list of FIPS-validated cryptographic modules grouped by vendor at the following NIST Web site:

<http://csrc.nist.gov/cryptval/140-1/1401vend.htm>

The following table lists FIPS-validated encryption and hashing algorithms that can be set for various Windows operating systems.

| | <i>FIPS-validated hashing algorithms</i> |
|-------------|--|
| XP SP3 | SHA-1, SHA-256, SHA-384, SHA-512 |
| 2003 Server | SHA-1, SHA-256, SHA-384, SHA-512 |
| Vista | SHA-1, SHA-256, SHA-384, SHA-512 |
| Windows 7 | SHA-1, SHA-256, SHA-384, SHA-512 |

Note: In response to NIST Special Publication 800-131A from the National Institute of Standards and Technology, the SHA-1 hashing algorithm is not supported in FIPS 140 mode.

In response to NIST Special Publication 800-131A from the National Institute of Standards and Technology, the 3DES-112 (also known as "two key" 3DES) algorithm is not supported in FIPS 140 mode.

When used with the *fipsmode* option, the commands *listcryptalgorithms* and *listhashalgorithms* list only available FIPS-validated algorithms. For example:

```
pkzipc -fipsmode -listcryptalgorithms
```

```
pkzipc -fipsmode -listhashalgorithms
```

The *fipsmode* option has two sub-options, *Enabled* and *Disabled*, used to configure the default state of the option or, on the command line, to override the configured default.

On Windows, SecureZIP sets the default state of the *fipsmode* option according to the Windows FIPS policy setting System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing. This setting is set by an administrator in the Local Security Policy or as part of Group Policy. It affects the behavior of Microsoft Internet Explorer and various areas of the operating system, depending on the version of Windows. If the setting is enabled, the default value of *fipsmode* is *Enabled*.

Note: The fastest version of the Advanced Encryption Standard (AES) is not FIPS-compatible. If your system is FIPS-enabled, you will not be able to use the FastAES sub-option with the *-cryptooptions* command. See “Advanced Encryption Options in Windows” in Chapter 5.

The following example turns on *fipsmode* for the current command line:

```
pkzipc -add -recipient="John Public" -fipsmode save.zip *.doc
```

The next example turns on *fipsmode* and uses the *sfx* option to create a graphical Windows self-extracting archive *mysfx.exe*. A self-extracting (SFX) archive created with *fipsmode* on extracts in FIPS mode, by default, too.

```
pkzipc -add -recipient="John Public" -fipsmode -sfx=win32_x86
mysfx *.doc
```

For more information on self-extracting archives, see “Working with Self-Extracting (PKSFX) Archives” later in this chapter.

The example below overrides a configured default setting of *fipsmode=enabled* to turn off *fipsmode* for the current command line:

```
pkzipc -extract -fipsmode=disabled wedding_plans.zip *.*
```

The following command line prefixes the *fipsmode* option with two hyphens (*--*) to turn off FIPS mode when extracting an SFX archive that was created with the *fipsmode* option on. Ordinarily, an SFX archive that was created with the *fipsmode* option on extracts in FIPS mode too. This example shows how to override the FIPS flag set internally in the SFX archive to allow files in the archive to be decrypted and authenticated without using only FIPS-validated algorithms:

```
mysfx.exe --fipsmode
```

Conversely, the *fipsmode* option can also be used with a single hyphen to apply FIPS-mode constraints on extraction to an SFX archive that was not created with the *fipsmode* option on.

```
mysfx.exe -fipsmode
```

Creating OpenPGP Files

Some organizations use encryption tools based on the OpenPGP standard, rather than X.509. OpenPGP uses the same Public Key Infrastructure principles for exchanging encrypted files, but uses a decentralized “Web of Trust” method of authenticating signatures. See “Working with OpenPGP Files” in Chapter 5 for more information.

When you create a file using `-archivetype=pgp`, as in the following example, SecureZIP creates a GNU TAR archive, copies the selected file(s) to the archive, and then encrypts the TAR archive using OpenPGP. This command-line takes all text files in the current directory, creates a PGP archive called `myfile.pgp`, encrypts it with 128-bit AES and makes it available to a recipient, Test:

```
pkzipc -add -archivetype=pgp -cryptalg=AES,128 -recipient="Test"
-cert="Test" myfile.pgp *.txt
```

Note: Always use the `-archivetype` command when working with OpenPGP files.

Attaching Digital Signatures

With SecureZIP, you can attach a digital signature to files in an archive, or to an archive itself. A digital signature assures people who receive the signed file that it is really from the person who signed it and has not been changed.

Note: PKZIP authenticates digital signatures on files signed by others, but you must have SecureZIP to attach digital signatures of your own.

SecureZIP allows you to digitally sign either individual files in an archive or the central directory of the archive, or both. The central directory contains a list of files in the archive. Signing the central directory enables a recipient to confirm that the archive as a whole has not changed. Both PKZIP and SecureZIP authenticate digital signatures on extraction.

Find more information on using digital certificates in Chapter 5.

Commands and Options for Signing Archives

certificate

Use the `certificate` option to specify a certificate to use to sign files. To specify a certificate, use one of the sub-options described in the following table.

Note: The `certificate`, `hash`, and `sign` options described below and the ability to use certificates to attach digital signatures are available only with SecureZIP.

| Sub-Option | To use | For example |
|-----------------|---|--|
| <Common name> | Specify, in quotes, the common name of the subject of the certificate (that is, the <code>cn</code> field in a string representation of a certificate); optionally, precede with: <code>cn=</code> SecureZIP searches for certificates by common name by default. | <code>-certificate=cn="John Public"</code> <code>-certificate="John Public"</code> |
| <Email address> | Specify the email address of the certificate (that is, the <code>e</code> field in a string representation of a certificate); optionally, precede with: <code>e=</code> | <code>-certificate=e=john.public@xyz.com</code> <code>-certificate=john.public@xyz.com</code> |

| Sub-Option | To use | For example |
|--------------|--|---|
| #<file name> | <p>Specify the name and location of a file containing the certificate to use.</p> <p>If the certificate's private key is not in the file with the certificate, use the keyfile option to point to the separate file that contains the private key. If necessary, use the keypassphrase option to specify a passphrase to read the private key.</p> | <pre>pkzipc -add - certificate=#mycert.pem -keyfile=mykey.key save.zip *.doc pkzipc -add - certificate=#mycert.p12 -keypassphrase="my passphrase" save.zip *.doc</pre> |

For example, if the common name of the subject is *John Q. Public*, you can specify that certificate as follows:

```
pkzipc -add -certificate="John Q. Public" test.zip
```

The command uses the *John Q. Public* certificate to sign files. By default, both the files in the archive and the archive itself are signed. Use the *sign* option to change what is signed. Use the *hash* option to change the hash method used for signing.

The following examples reference a certificate by email address:

```
pkzipc -add -certificate=john.public@nowhere.com test.zip
pkzipc -add -certificate=e=john.public@nowhere.com test.zip
```

The prefix "e=" when using an email address is optional. SecureZIP automatically looks for an email address if the string contains an "@" and a dot and looks like an email address.

Note that a certificate must contain an email address in order to be found by this method. Not all certificates embed an email address.

keyfile

You can reference a file that contains a certificate to use for signing with the #<filename> sub-option of *certificate*. If the private key is not included in the file with the certificate, use the *keyfile* option to specify the file that contains the private key. For example:

```
pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key save.zip
*.doc
```

The *keyfile* option specifies a file containing the private key for the certificate specified by the *certificate* option. The option is most useful with SSL server certificates, which often have the private key and certificate in separate files.

keypassphrase

A private key in a file by itself or in a file that contains a certificate may be encrypted and require a passphrase for PKZIP to decrypt it to use. Use the *keypassphrase* option to supply the passphrase. For example:

```
pkzipc -add -certificate=#mycert.p12 -keypassphrase="my
passphrase" save.zip *.doc

pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key -
keypassphrase="my passphrase" save.zip *.doc
```


The *keyphrase* option specifies the passphrase used to decrypt private key information. This can be the passphrase used for your certificate store (UNIX only), for a PKCS#12 file (specified with the *certificate* option), or a key file specified with the *keyfile* option.

hash

You can use the *hash* option with the *certificate* option to specify the hash method/algorithm to use for signing. The option has the sub-options shown in the following table.

| Sub-option | Description |
|---------------|--|
| <i>sha1</i> | Uses the SHA-1 hashing algorithm (default) (not FIPS-compatible; cannot be used with the <i>fipsmode</i> option) |
| <i>sha256</i> | Uses the SHA-256 hashing algorithm (<i>fipsmode</i> default) |
| <i>sha384</i> | Uses the SHA-384 hashing algorithm |
| <i>sha512</i> | Uses the SHA-512 hashing algorithm |
| <i>md5</i> | Uses the MD5 hashing algorithm (not FIPS-compatible; cannot be used with the <i>fipsmode</i> option) |

The SHA algorithms are all stronger than the MD5 algorithm. Among the SHA algorithms, the higher-numbered ones are stronger than the lower-numbered ones. See the *fipsmode* option for information on which algorithms are supported for FIPS processing on different versions of Windows.

Use the *listhashalgorithms* command to list hashing algorithms available on your system. If *fipsmode* is on, the *listhashalgorithms* list shows only FIPS-validated algorithms.

The *hash* option's default is configurable.

The following example specifies the SHA-256 algorithm and the "My Cert" certificate to use to sign files:

```
pkzipc -add -certificate="My Cert" -hash=sha256 test.zip *.*
```

sign

You can use the *sign* option with the *certificate* option to specify whether to sign the central directory of the archive itself, the archived files, or both.

Signing the files enables a user to verify that the files are the same files you signed; signing the archive itself enables a user to verify that the contents of the archive have not changed—that, for example, no files have been added or removed. By default, SecureZIP signs both.

The sub-options are listed in the following table.

| Sub-option | Description | Example |
|------------|--|----------|
| <i>cd</i> | Sign only the central directory of the archive, not the files in the | -sign=cd |

| <i>Sub-option</i> | <i>Description</i> | <i>Example</i> |
|-------------------------|---|----------------|
| | archive | |
| <i>files</i> | Sign only the files in the archive, not the archive itself | -sign=files |
| <i>all</i> (Default) | Sign both the archived files and the archive itself | -sign=all |
| <i>none</i> | Do not sign files. This sub-option is used to turn signing off if it has been configured. | -sign=none |

For example:

```
pkzipc -add -certificate="My Cert" -sign=cd test.zip *.*
```

You can also use *sign* to add a digital signature to an existing archive. See “Attaching a Signature to an Existing Archive” in Chapter 5 for more information.

listcertificates

Use the *listcertificates* command to list the certificates that are in a specified store on your system. Information for each certificate tells whether the certificate is *Valid*, *Expired*, *Not Trusted*, or *Revoked* (if known). If OpenPGP certificates are enabled and available on the system, these will be displayed.

Specify the store using one of the sub-options in the following table. Personal certificates in the MY store are listed by default if no sub-option is used.

| <i>Sub-option</i> | <i>Description</i> | <i>Example</i> |
|--------------------|--|---|
| <i>my</i> | Lists certificates in the MY store. This store contains your personal certificates with private keys. | pkzipc -listcertificates or pkzipc -listcert=my |
| <i>addressbook</i> | Lists certificates in the AddressBook store. This store contains public certificates and public keys belonging to other people. | pkzipc -listcert=addressbook |
| <i>ca</i> | Lists certificates in the CA store. These are intermediate certificates in a trust chain, created by a certificate authority to validate other certificates. | pkzipc -listcert=ca |
| <i>root</i> | Lists certificates in the Root store. These are certificates at the beginning of a trust chain, which are trusted by the system. | pkzipc -listcert=root |

For example, the following command line lists certificates in the MY store:

```
pkzipc -listcertificates
```

The command line produces output like the following. In this case, the MY store contains four certificates, three of which have the same name, *John Doe*.

```
John Doe: Valid
```

```
John Doe: Expired  
John Doe: Expired  
users, John Doe: Valid
```

Setting a Default Certificate

If you only use one digital certificate to sign your archives, you can skip the *certificate* command in your scripts. Do this by defining that certificate as your default.

To define the “John Q. Public” certificate as your default, use the following command:

```
pkzipc -config -certificate="John Q. Public"
```

Once you have a certificate configured, each time you use *sign*, whether as a command or option, SecureZIP will attach this certificate to your archive.

You can still use the *certificate* option to attach a different certificate than your default.

See Chapter 7 for more information on setting and changing defaults.

Time Stamping Your Signed ZIP Archive

When you need to establish not only *who* is responsible for a file or set of files, but also *when* it was created, digital timestamping is a critical service. As you know, dates are critical for establishing original intellectual property rights, including copyright and patents. While all files carry a creation date as part of its default metadata, it is not very hard to manipulate this date before you create and sign the archive in question. The goal is to create a timestamp that cannot be changed, even by the owner of the file. Using a Time Stamp Authority outside of your computing environment takes the guesswork out of confirming the validity of a document.

The Internet Engineering Task Force governs digital timestamps through two standards: RFC 3161 establishes the method by which a client can connect to a secure computer that will stamp the document with its current date and time. This secure computer is called the Time Stamp Authority (TSA) or Time Stamp Server (TSS). RFC 4998, among many other things, defines what happens when a time stamp authority’s certificates expire, or are otherwise compromised.

With SecureZIP Command Line’s support for digital timestamping, you can add a timestamp to any signed archive.

Note: Digital timestamping, along with other features related to digital signatures are available only with SecureZIP.

SecureZIP only supports digital timestamping for ZIP archives.

Before beginning the process, you need to know the URL of your Time Stamp Authority. The TSA server may be on your network or on a public server.

To sign a new archive containing all documents in the current directory, and add a digital time stamp, type:

```
pkzipc -add -sign=timestamp -ts=<TSA_URL> test.zip *.docwhere <TSA_URL> is the location of your Time Stamp Authority’s service.
```

SecureZIP will calculate a hash based on the archive’s data and send that to the TSA. The TSA adds a timestamp to the hash and calculates the hash of this combination of the original hash with the timestamp. This second hash is then digitally signed with the TSA’s private key. All this information is then sent back to you. SecureZIP then adds the timestamp to the archives central directory signatures.

Updating and Renewing Time-Stamped Archives

The IETF standards permit multiple timestamps on a file, allowing for time-stamped archives to be updated and refreshed. In this way, you can establish a record of creation and updates. SecureZIP automatically handles updating the timestamp when you update the archive. Because the archive has changed, a new timestamp will be generated, but the original file signatures will be preserved by nesting the two signatures.

You must renew your time-stamped archives before the TSA's certificate expires.

Use the *sign* command:

```
pkzipc -sign=timestamp -ts=http://<TSA URL> test.zip
```

When renewing the timestamp, the original items and their order in the archive will be preserved normally.

Note: Renewing timestamped archives spanned across different media is not supported.

Writing an Archive to STDOUT and Special Files

Ordinarily, when you use the *add* command to archive files, you write the resulting archive to a physical file that you specify in the command line. For example, the following command line archives text files to the archive `myfiles.zip`:

```
pkzipc -add myfiles.zip *.txt
```

An archive can also be written, as a data stream, to some other destinations besides a physical file, notably, to STDOUT, a named pipe, a UNIX domain socket, or a device file.

Note: When PKZIP compresses and encrypts data to write an archive to a data stream, the data goes to the stream without ever appearing on disk in unencrypted form. PKZIP does create a temporary file to get the size of the data to put in local headers, which must be written before file data. But the data is already compressed and encrypted when it's placed in the temporary file. No security vulnerability is created.

Writing an Archive to STDOUT

You can write an archive to *standard output*, or STDOUT, instead of to a physical file. Data written to STDOUT appears on your computer screen but is not saved to disk (unless you do something extra to save it). It can also be piped to another program or be redirected to (for instance) a file.

To have PKZIP write the output of the *add* command to STDOUT, use a hyphen "-" in place of the name of an archive file. You must also use the *noarchiveextension* option to prevent PKZIP from outputting to a file named `-.zip` instead of to STDOUT. And finally, you should include the *silent* option to suppress the informational messages that PKZIP normally outputs so that these are not inserted in the archive data stream. For example:

```
pkzipc -add -noarchiveextension -silent=normal - *.txt
```

PKZIP creates ZIP-format archives by default. To write a different type of archive to STDOUT, use the *archivetype* option to specify the type. For example, the following command line tells PKZIP to write a TAR-format archive to STDOUT:

```
pkzipc -add -archivetype=tar -noarchiveextension -silent=normal - *.txt
```

The command line below sends output to STDOUT and then redirects that output to archive `myfile.zip`.

```
pkzipc -add -noarchiveextension -silent=normal - *.txt >
myfile.zip
```

When redirecting STDOUT to a file, you can use the *exclude* option to make sure that PKZIP does not include the file to receive the output in the set of files to be zipped. Unlike when writing directly to a specified archive file, PKZIP cannot infer from the command line that it should skip a file to which you redirect output. The *exclude* option explicitly tells PKZIP to skip specified files.

For example, the following command line archives all files in a directory and redirects output to a file in the same directory. The *exclude* option tells PKZIP not to add that file.

```
pkzipc -add -noarchiveextension -silent=normal -exclude=myfile.zip
- *.* > myfile.zip
```

You can use a hyphen “-” in place of the name of an archive file when you extract, as well. Used in a command line with the *extract* command, the hyphen tells PKZIP to extract files from STDIN (*standard input*).

For example, the following command line extracts files from STDIN instead of from a named archive.

```
pkzipc -extract -noarchiveextension -silent=input -
```

When extracting from STDIN, set *silent* to the *input* sub-option, as in the command line above, to suppress any PKZIP requests for input (a passphrase, for example). If input is needed, the extraction fails with an error.

The *noarchiveextension* option is needed so that PKZIP does not try to extract from a file named `-.zip`. If the archive is not a ZIP archive, use the *archivetype* option to specify its type. For example, the following command line tells PKZIP that the file is a BZIP2 archive:

```
pkzipc -extract -archivetype=bzip2 -noarchiveextension -
silent=input -
```

You can combine writing to STDOUT and extracting from STDIN to securely transfer files between two systems. For example, the following (UNIX) command line compresses and encrypts the files to be transferred and adds them to a ZIP archive. The archive is written to STDOUT instead of to a file. The command line pipes the output to the *rsh* (*remote shell*) system command, which runs PKZIP on the remote system to extract the files from STDIN.

```
pkzipc -add -noarchiveextension -cryptalgorithm=aes,256 -
recipient=Jon -silent
- | (rsh user@remote_system pkzipc -extract -noarchiveextension -
silent=input - )
```

Writing an Archive to a Named Pipe

An archive can be written to a named pipe instead of to a physical file.

The named pipe, socket, or device must already exist. You can then write an archive to it with a command line like the following. Use the name of the pipe in the command line in place of the name of an archive file.

```
pkzipc -add -noarchiveextension <name of pipe> <files to zip>
```

As when writing to STDOUT, you must use the *noarchiveextension* option to prevent PKZIP from outputting to a `.zip` file—in this case, one named for the pipe.

PKZIP creates ZIP-format archives by default. To write a different type of archive, use the *archivetype* option to specify the type. For example, the following command line tells PKZIP to write a TAR-format archive:

```
pkzipc -add -archivetype=tar -noarchiveextension <name of pipe>
<files to zip>
```

You must use the full UNC path when referring to a named pipe on Windows. For example:

```
pkzipc -add -noarchiveextension \\.\pipe\mypipe *.doc
```

In the preceding example, the dot in the path

```
\\.\pipe\mypipe
```

references the current machine. To reference a pipe on a different machine—named *boulder*—specify the machine.

```
\\boulder\pipe\mypipe
```

You can use either a name or an IP address to specify a machine.

Compressing Files in Subdirectories

recurse

PKZIP does not automatically compress files that appear in subdirectories, unless you specify those directories, or use the *recurse* option with the *add* command. With the *recurse* option, all specified files in a directory structure, including files located in subdirectories will be compressed.

If you have a directory called *tut* with a nested subdirectory called *test*, to compress all of the files in the *tut* directory and all files in the *tut/test* directory, you would type the following in the *tut* directory:

```
pkzipc -add -recurse test.zip *
```

All files in the *tut* directory as well as those files in subdirectories of the *tut* directory are compressed. However, directory path information is not stored within the .ZIP file. If you want to store directory information within your .ZIP file (in addition to compressing all the files in those directories), use the *path* option with the *recurse* option or simply use the *directories* option.

Compressing Open Files

OpenFile

In Windows, PKZIP does not automatically include files that are open in other applications in archives, as there is a small chance there could be differences between the file on screen (or in memory) and the file saved to disk. Use the *OpenFile* option to include open files in your archive.

The *OpenFile* option has sub-options that allow you to set notification and inclusion for open files that match the pattern you want to archive. These sub-options are listed in the table below. By default, using the *OpenFile* option without a sub-option includes all matching open files in your archive.

| <i>Sub-Option</i> | <i>Description</i> | <i>For example</i> |
|-------------------|---|--|
| <i>never</i> | PKZIP does not include any open files. A warning will appear if a matching file is open. | <i>pkzipc -add -OpenFile=never test.zip *.bmp</i> |
| <i>all</i> | PKZIP includes all matching open files without prompting first. A message noting each open file is included in the standard output. | <i>pkzipc -add -OpenFile test.zip *.bmp</i> |
| <i>prompt</i> | PKZIP notifies you when a matching file is open, and asks whether to add the open file or skip it. | <i>pkzipc -add -OpenFile=prompt test.zip *.bmp</i> |

Storing Directory Path Information

path

Normally, when PKZIP compresses files, only the files are stored within the .ZIP file, not the paths of those files. However, you can instruct PKZIP to store the directory path information of a file within the .ZIP file. This enables you to restore the directory structure when you extract the files.

For example, if a file you are compressing appears in the doc/temp directory, you can store the file within the .ZIP file as:

```
doc/temp/<file name>
```

To do this, use the *path* option with the *add* command. For example, the following command line adds all .TXT files in the specified directories and saves the specified path information:

```
pkzipc -add -path test.zip doc/temp/*.txt
```

If path information is saved, you can use the *directories* option with the *extract* command to extract files to the saved paths. PKZIP creates the directories on the saved path if they do not already exist.

Note that the *path* option gets files only from the specified directory. To get files in subdirectories of that directory as well, use the *directories* option instead of *path*. Or use *path* together with *recurse*.

Additional Methods for Storing Directory Path Information

The *path* option has sub-options that enable you to specify the path information stored. These sub-options are listed in the table below. By default, using the *path* option without a sub-option stores relative path information for all files added.

| <i>Sub-option</i> | <i>To</i> | <i>For example</i> |
|-------------------|--|--|
| <i>current</i> | Store the directory path relative to the current location. | <code>pkzipc -add -path=current docs.zip docs/*</code> In this example, only directory information under the docs directory will be stored. Parent directory information will not be stored. |
| <i>root, full</i> | Store the full path, starting from the root directory down. | <code>pkzipc -add -path=root docs.zip docs/*</code> In this example, the entire directory path, starting from "root" directory will be stored. |
| <i>specify</i> | Stores path information for subdirectories under the specified directories | <code>pkzipc -add -path=specify docs.zip temp/docs/*</code> Stores path information for subdirectories under temp/docs. |
| <i>relative</i> | Store the directory path relative to the current working directory of the drive specified. | <code>pkzipc -add -directories=relative docs.zip c:*.doc z:*.doc</code> In this example the path information for those directories recursed under the current working directory (for both the C: and Z: drives) will be stored. |
| <i>none</i> | Turn off the path option. (Used to override configuration file). | <code>pkzipc -add -path=none docs.zip /temp/docs/*</code> In this example, only the file names are stored. |

Storing and Recreating Directory Path Information

directories

The *directories* option works with both the *add* and *extract* commands.

- With the *add* command, the *directories* option is equivalent to using the *recurse* and *path* options together. It instructs PKZIP to search subdirectories for files and to save the files and their directory path information in the .ZIP file.
- With the *extract* command, the *directories* option extracts any directory tree structure saved with files.

The following example uses the *directories* option with the *add* command to add any files called `whatsnew.htm` in the current directory or in any subdirectory of the current directory:

```
pkzipc -add -directories testdir.zip whatsnew.htm
```

Or abbreviated:

```
pkzipc -add -dir testdir.zip whatsnew.htm
```

Screen output lists any matching files found in subdirectories:

```
Creating .ZIP: testdir.zip
```

```
Adding File: Win/PK/Whatsnew.htm Deflating (67.0%), done.
Adding File: Win/SZ/Whatsnew.htm Deflating (66.7%), done.
```

The following example gets all `.htm` files in the current directory or its subdirectories:

```
pkzipc -add -dir testdir.zip *.htm
```


To tell PKZIP to start looking for matches from a subdirectory of the current directory, specify the path to the subdirectory. The following example gets all `whatsnew.htm` files in `mysub\` or any of its subdirectories:

```
pkzipc -add -directories testdir.zip mysub\whatsnew.htm
```

The example below gets all `.htm` files in `mysub\` or any of its subdirectories:

```
pkzipc -add -directories testdir.zip mysub\*.htm
```

If you have multiple `mysub\` subdirectories under the current directory, you can get files from just those subdirectories by using a wildcard for the subdirectory from which to start the search:

```
pkzipc -add -directories testdir.zip *\mysub\whatsnew.htm
```

The command line below is similar, but it limits the search for `mysub\` subdirectories to just those under the `nextsub\` subdirectory:

```
pkzipc -add -directories testdir.zip nextsub\*\mysub\whatsnew.htm
```

Even if the command line includes the *directories* option, you can turn off the searching of subdirectories for matching files by specifying a full path beginning with a backslash (for the root directory) or (on Windows) a drive letter (for example, `C:`) in the pattern. The pattern must also not include any wildcard characters (`*` or `?`).

For example, the following command line adds only the specified file; it does not add matching files from subdirectories of `MyFiles`:

```
pkzipc -add -directories testdir.zip C:\MyFiles\whatsnew.htm
```

For information on extracting files saved with directory information, see the section “Retaining Directory Structure while Extracting” in Chapter 4.

As with the *path* option, PKZIP provides several choices for saving directory path information. The following table lists the sub-options you can use with *directories* option:

| <i>Sub-option</i> | <i>To</i> | <i>For example</i> |
|---------------------|--|---|
| <i>current</i> | Store the directory path relative to the current location. | <pre>pkzipc -add -directories=current docs.zip docs/*</pre> In this example, only directory information under the docs directory will be stored. Parent directory information will not be stored. |
| <i>root or full</i> | Store the full path, starting from the root directory down. | <pre>pkzipc -add -directories=root docs.zip docs/*</pre> In this example, the entire directory path, starting from "root" directory will be stored. |
| <i>specify</i> | Store path information for subdirectories under the specified directories | <pre>pkzipc -add -directories=specify docs.zip temp/docs/*</pre> Stores path information for subdirectories under temp/docs. |
| <i>relative</i> | Store the directory path relative to the current working directory of the drive specified. | <pre>pkzipc -add -directories=relative docs.zip c:*.doc z:*.doc</pre> In this example, the path information for those directories recursed under the current working directory (for both the C: and Z: drives) will be stored. |

| <i>Sub-option</i> | <i>To</i> | <i>For example</i> |
|-------------------|--|--|
| <i>none</i> | Turn off the path option. (Used to override configuration file). | <code>pkzipc -add -directories=none docs.zip /temp/docs/*</code> In this example, only the file names are stored. |

Setting the Compression Level

Native ZIP compression (which uses the Deflate compression algorithm) and the *bzip2* and *deflate64* compression options each support a range of compression levels from 0 (no compression) to 9 (maximum). By default, each of these options uses level 5, or *normal*, compression. Normal compression strikes a middle balance between compression and performance. In general, greater compression takes more time.

You can use the *level* option to specify a compression level from 0 to 9 when you create or update a ZIP file using one of the compression methods named above.

Alternatively, you can use the options *normal*, *store*, *speed*, *fast*, and *maximum* to specify a desired balance between speed and degree of compression. See “Specifying a Compression Level by Name” later in this chapter.

With the *dclimplode* option, you set the compression level in a different way, namely, by specifying the dictionary type and size as sub-options.

Specifying a Compression Level from 0-9

level

The *level* option enables you to specify a level or degree of compression to use when creating or updating a ZIP archive with the Deflate64, BZIP2, or default Deflate compression methods. (See the *deflate64* and *bzip2* options to learn about using these compression methods.)

To set a compression level with the *level* option, specify a numeric value for the option from 0 to 9. A value of 0 specifies zero compression.

The following command line specifies a compression level of 2 and uses the native Deflate compression method:

```
pkzipc -add -level=2 test.zip *.doc
```

The following command line specifies level 2 compression and the BZIP2 compression method to create or update a ZIP archive:

```
pkzipc -add -bzip2 -level=2 test.zip myfile.doc
```

Level 5 is the default compression level for *level*. You can use the *configuration* command to set a different default. For example, the following command line sets the default value for *level* to 9:

```
pkzipc -config -level=9
```

For information on changing default settings, see Chapter 7.

Specifying a Compression Level by Name

store, speed, fast, normal, maximum

As an alternative to setting numeric compression levels with *level*, you can use the options *normal*, *store*, *speed*, *fast*, and *maximum*.

These options enable you to use non-numeric names to specify a desired balance between speed and degree of compression. For example, the following command line specifies the *fast* compression option:

```
pkzipc -add -fast test.zip *.doc
```

The non-numeric compression level options are described in the following table:

| <i>Option</i> | <i>Description</i> | <i>Example</i> |
|----------------------------|--|--|
| <i>speed</i> | Provides the fastest performance and the least compression: some files are compressed with the Deflate method, using level 1 compression; others* are stored (level 0) uncompressed. | <pre>pkzipc -add -speed test.zip *.doc pkzipc -add -bzip2 -speed test.zip *.doc</pre> |
| <i>fast</i> | Provides the second fastest compression: some files are compressed with the Deflate method, using level 2 compression; others* are stored (level 0) uncompressed | <pre>pkzipc -add -fast test.zip *.doc</pre> |
| <i>maximum</i> | Provides the highest level of compression (level 9) | <pre>pkzipc -add -max test.zip *.doc</pre> |
| <i>store</i> | Provides zero compression: just stores files inside the archive (level 0) | <pre>pkzipc -add -store test.zip *.doc</pre> |
| <i>normal</i> (Default) | Provides a middle balance of compression and speed (level 5) | <pre>pkzipc -add -norm test.zip *.doc</pre> You would only need to use this option if you changed the default compression level. See Chapter 7 for information on setting defaults. |

* Types of files that the *speed* and *fast* options store uncompressed are listed below. The other named options (except *store*) compress files of these types. You can also use the *level* option to compress files of these types.

| | |
|---------|--------|
| *.bz2 | *.jpeg |
| *.bzip2 | *.jpg |
| *.cab | *.mp3 |
| *.gz | *.mpeg |
| *.gzip | *.mpg |
| *.rar | *.sxw |
| *.gif | |

Compressing Files with a List File

Instead of specifying a specific file or file pattern in your command line, you can point PKZIP to a list file that lists all the files or file patterns that you want to operate on. A list file is an ASCII text file that contains file names or file patterns and path information. A list file can be an ideal solution for users who archive specific file sets on a regular basis. Using a list file saves time in that you do not need to type file names and paths each time you wish to compress these files with PKZIP. A list file may contain wildcard specifications (*,?) as well as exact file names and paths.

A list file in a DOS based environment might look similar to the following:

```
*. exe
*. doc
\tut\*. doc
\tut\?????. *
pkzip. html
```

You reference a list file in the command line by prefixing its name with the list character—“@” by default. See the *listchar* option if you want to use a different character.

The following example adds the files listed in list file *lst.txt* to the archive *test.zip*:

```
pkzipc -add test.zip @lst.txt
```

You can also use a list file to specify files to exclude from an archive, based on some criteria, using the *exclude* option. The *exclude* option is discussed in Chapter 1. For more information on the *listchar* option, see “Changing the List Character for List Files” in Chapter 8.

Note: The way you list files to extract is slightly different from the way you list files to add to an archive. See “Extracting Files with a List File” in Chapter 4 for more information.

Getting a List of Files from Standard Input

Use a hyphen (-) prefixed with the list character (“@” by default) to identify a set of files in standard input as a list. For example, in the following command line, PKZIP treats a list of files output from *some program* as a list file and compresses the files into *test.zip*:

```
<some program> | pkzipc -add test.zip @-
```

The special, dynamically constructed list can also be used with the include and exclude options. For example:

```
<some program> | pkzipc -add test.zip -include=@-  
<some program> | pkzipc -add test.zip -exclude=@- *.doc
```

Compressing Files with the Deflate64 Method

deflate64

The *deflate64* option enables you to use the Deflate64 compression method to compress files and create ZIP archives. The Deflate64 method can produce greater compression than the Deflate method that PKZIP uses by default because Deflate64 uses a larger dictionary window (64K compared to 32K).

Not all ZIP-compatible programs from other vendors can extract files compressed with the Deflate64 method.

You can use the *level* option with *deflate64* to specify a level of compression from 0 to 9 (0 is zero compression).

The following command line uses the Deflate64 method with the *level* option set for maximum compression:

```
pkzipc -add -deflate64 -level=9 mydocs.zip *.doc
```

Compressing Files with the BZIP2 Method

bzip2

BZIP2 is an open-source compression algorithm that requires more memory and processing power than standard ZIP compression but provides greater compression. PKZIP can use BZIP2 compression to create either ZIP or BZIP2-format archives (.bz2 files). A BZIP2 archive, unlike a ZIP archive, can contain only a single file.

Files compressed with the BZIP2 method can be extracted with most versions of PKZIP, 4.6 and later, but other ZIP-compatible programs may not be able to extract files compressed with BZIP2.

You can use the *level* option with *bzip2* to specify a level of compression from 0 to 9 (0 is zero compression).

The following command line uses the BZIP2 method to create a ZIP file. The *level* option specifies maximum compression:

```
pkzipc -add -bzip2 -level=9 mydocs.zip *.doc
```

Compressing Files with the LZMA Method

lzma

The LZMA compression algorithm often produces a higher compression ratio than Bzip2 but uses a lot of memory—as much as 16 MB—and takes more time than Deflate.

Files compressed with the LZMA method can be extracted with PKZIP versions 12.3 and later, but other ZIP-compatible programs may not be able to extract such files.

Compressing Files Compatible with the Data Compression Library

dclimplode

The *dclimplode* option enables you to use the same compression algorithms used by the PKWARE Data Compression Library. Files compressed with this method can be extracted by most versions of PKZIP 2.5x and later, though not by other .ZIP-compatible programs.

When using the Implode compression method, you must specify dictionary type (ASCII or BINARY) and dictionary size (1024, 2048, or 4096). In general, the larger the dictionary, the greater the compression. Use the BINARY dictionary when compressing binary files (for example, executable programs) or when the type of the file is unknown. Use the ASCII dictionary with ASCII (text) files.

For example, to use the DCL Implode method to compress all text files in a directory, type the following:

```
pkzipc -add -dclimplode=ascii,4096 text.zip *.txt
```

Compressing Files with the PPMd Method

ppmd

The *ppmd* option achieves especially good compression for natural language text but can use a lot of memory (~16 MB) and takes more time than Deflate.

Files compressed with the PPMd method can be extracted with PKZIP versions 12.3 and later, but other ZIP-compatible programs may not be able to extract such files.

Compressing Files to a Specified Type of Archive

archivetype

The *archivetype* option explicitly tells PKZIP the type of archive to create or extract. Use the option when PKZIP cannot figure out the correct archive type from the archive's file name. For some examples, see "Writing an Archive to STDOUT."

PKZIP creates ZIP archives by default: When you use the *add* command to create a new archive, PKZIP creates a ZIP archive if you do not specify a file name extension that PKZIP recognizes as associated with a particular archive type.

For example, the following command creates a ZIP archive called *myfile.foo.zip*:

```
pkzipc -add myfile.foo
```

Similarly, if the command line does not tell PKZIP the type of archive to extract from, PKZIP tries to extract files from a ZIP-format file.

With the *archivetype* option, you can explicitly tell PKZIP the type of archive to work with.

For example, the following command line creates an archive *myfile.foo.bz2* of the BZIP2 archive type. The file name extension *bz2* associated with the BZIP2 archive type is added to the file name:

```
pkzipc -add -archivetype=bzip2 myfile.foo
```

A simpler way to create a BZIP2 archive called *myfile.foo.bz2* is to specify the file name extension as part of the file name. In this case, you do not need the *archivetype* option:

```
pkzipc -add myfile.foo.bz2
```

Note: You cannot create an OpenPGP-based archive by using the *.pgp* extension. Always use *archivetype=pgp* when working with OpenPGP files.

When you specify the archive type with *archivetype*, you can include the *noarchiveextension* option to tell PKZIP not to add an extension to the file name. For example, the following command suppresses the *bz2* extension that would normally be appended and creates a BZIP2 archive named *myfile.foo*:

```
pkzipc -add -archivetype=bzip2 -noarchiveextension myfile.foo
```

Compressing Files to Diskette

span

With PKZIP, you can save your .ZIP file or self-extracting file to one or more diskettes when you create it (instead of saving it on your hard disk drive). You can also create a *split* archive that is saved as multiple files on your hard disk. You can also have PKZIP format or wipe your removable media before writing to it.

Creating a Spanned Archive

You can save a ZIP file to multiple diskettes if it is too large to fit on a single one. This is called disk *spanning*. PKZIP prompts you to insert diskettes (or other media) as they are needed.

Depending on the size of the ZIP file, it may be necessary for PKZIP to save the file on multiple diskettes. This process is called "spanning".

To create a spanned archive:

1. Insert a diskette (or other appropriate medium) into its drive.
2. Type your PKZIP command, and press ENTER. Make sure to specify the drive letter or path that corresponds to your destination drive. A sample command line appears below:

```
pkzipc -add -span a:\test.zip *.doc
```

Note: Ordinarily, PKZIP recognizes removable media as such and spans them as necessary automatically, even if you do not specify the *span* option. However, if PKZIP is unable to detect that you are creating your ZIP file on removable media, use the *span* option to tell PKZIP to span.

Creating a Split Archive

The *span* option is also used to create a *split* archive. A split archive is an archive created in segments, all of which are written to your hard disk as separate files.

To create a split archive on your computer disk, specify a size in bytes, or use a predefined size from the following table:

| <i>Predefined size</i> | <i>Comment</i> |
|------------------------|------------------------------------|
| 360 | 360KB floppy disk (362496 bytes) |
| 720 | 720KB floppy disk (730112 bytes) |
| 1.2 | 1.2MB floppy disk (1213952 bytes) |
| 1.44 | 1.44MB floppy disk (1457664 bytes) |
| 2.88 | 2.88MB floppy disk (2915328 bytes) |
| 95.7 | 100MB ZIP disk (100431872 bytes) |
| 650 | 650MB CD-ROM (681574400 bytes) |
| 700 | 700MB CD-ROM (734003200 bytes) |

For example, to create a split archive of size 1.44 Mb to your local system, type the following command:

```
pkzipc -add -span=1.44 c:\test.zip *.doc
```

To have PKZIP format or wipe removable media before writing to it, use the *span* command with *format* or *wipe*. For example, the following command line formats the media prior to creating a ZIP archive:

```
pkzipc -add -span=format a:\test.zip *.doc
```

Preserving International Characters in File Names

utf8

The *utf8* option enables UTF-8 characters in file names and file comments to be correctly displayed when an archive's contents are viewed or extracted in compatible non-UTF-8 locales.

For example, with the *utf8* option, you can archive files in a Japanese locale using the EUC character set (and the *utf8* option) and then correctly view or extract the files in a Japanese locale using the Shift-JIS character set.

The option can be used with these commands/options (*comment* can be either a command or an option):

- *Add*
- *Comment*

If a command line containing the *utf8* option modifies an archive in any way, UTF-8 characters are used in the names of all files in the archive.

Comments will always follow the format of the file name it is attached to. Applying `--utf8` to a comment on a file with UTF-8 character formatting will not remove UTF-8 characters from the comment.

In general, use the `utf8` option when you add to an archive files that contain international (that is, non-English) characters in file names and file comments. For example:

```
pkzipc -add test.zip -utf8 *.*
```

PKZIP displays the following message to highlight that the option is used:

```
Using UTF-8 file names and comments
```

PKZIP uses the `utf8` option automatically when run on UNIX in a UTF-8 locale (such as `ja_JP.UTF-8`); you do not need to use it explicitly.

The `utf8` option is incompatible with the `204` option: an error results if the two options are used together. (PKZIP does not turn on the `utf8` option automatically on UNIX if the `204` option is used.)

PKZIP/SecureZIP Server version 8.6 or SecureZIP for Windows version 11 is required to extract files added with the `utf8` option, so use the option only with archives that you expect to be extracted with these (or later) versions of these programs.

Creating Multiple, Respective Archives

archiveeach

With the `archiveeach` option, you can create a separate archive for each of multiple files specified in a single command line.

```
pkzipc -add -archiveeach *.*
```

With `archiveeach`, you do not specify names for new archives. PKZIP names each new archive after the file it contains, with an archive-type file name extension (`ZIP` by default) appended to the end. For example, a ZIP archive created for file `mydata.xls` is named `mydata.xls.zip`. An archive created for file `mydata.zip` is named `mydata.zip.zip`.

If an archive with the same name already exists in the target location, PKZIP appends a number to the archived file name before appending the `.zip` (or other file name extension). For example: `mydata.xls2.zip`.

To specify a particular archive type, use the `archivetype` option with the `archiveeach` option. The `archiveeach` option can also be used with the `encode` option, to convert the archive initially created to a different type. By using `archivetype` and `encode` together with `archiveeach`, you can, for example, create multiple `.tar.gz` files:

```
pkzipc -add -archiveeach -archivetype=tar -encode=gz C:\data\*.*
```

You can specify a destination for the new archives in a sub-option to `archiveeach`:

```
pkzipc -add -archiveeach=C:\newzips C:\myfiles\*.*
```

You can use the *substitution* option to have PKZIP add a timestamp to the name of a new destination directory created for the archives. See “Inserting a Timestamp in the Archive File Name” in Chapter 6.

Storing File Information

PKZIP allows you to store specific file attribute/information within your .ZIP file. You can:

- Store file attributes, including hidden, system, archive, and read-only.
- Store extended file attribute information.
- Remove (mask) file attributes.

Refer to the sections that follow for more information.

Compressing Files with Specified Attributes

attributes

PKZIP allows you to compress files based on the attributes that they possess. These attributes are usually assigned either by the creator of a file, a system administrator, or by the operating system. The following are attributes you can store:

- Hidden
- System
- Read-only
- Archive

The attributes set by default for compression are *archive* and *read-only*. With this setting, if you do not use the *attributes* option on your command line, PKZIP compresses all files except any having the attributes *hidden* or *system*.

To specify a file attribute, you must include it with the *attributes* option in your command line. Each attribute is a value for the *attributes* option. You can:

- Specify which file attributes to compress
- Override configured default values
- Turn off the *attributes* option

The table below lists all of the available sub-options for storing file attribute information:

| <i>Sub-Option</i> | <i>To</i> | <i>For example</i> |
|-------------------|---|---|
| <i>hidden</i> | Compress files including those that contain the "hidden" file attribute. | <code>pkzipc -add - attributes=hid test.zip</code> |
| <i>system</i> | Compress files including those that contain the "system" file attribute. | <code>pkzipc -add - attributes=sys test.zip</code> |
| <i>readonly</i> | Compress files including those that contain the "read-only" file attribute. | <code>pkzipc -add - attributes=read test.zip</code> |

| <i>Sub-Option</i> | <i>To</i> | <i>For example</i> |
|-------------------|---|--|
| <i>archive</i> | Compress files including those that contain the "archive" file attribute. | <code>pkzipc -add -attribute=archive test.zip</code> |
| <i>all</i> | Compress files including those that contain the hidden, system, or read-only file attribute. | <code>pkzipc -add -attributes=all test.zip</code> |
| <i>none</i> | Turn off the attributes option in the configuration file or compress files that do not have any attributes set. | <code>pkzipc -config -attributes=none</code> |

You may use a hyphen (-) before an *attributes* sub-option on your command line to exclude files with a specific attribute from being added regardless of the default attributes configuration setting. If, for example, the default attributes configuration setting was set to "all", you could enter the following command line to exclude hidden files from being added to the test.zip file.

```
pkzipc -add -attributes=-hidden test.zip
```

Extended Attribute Storage

noextended

When PKZIP adds files to an archive, PKZIP stores the standard FAT file system attributes (Read-Only, Archive, System, Hidden, Directory). By default, various extended attributes are stored as well. These include NTFS times on Windows. The extended attribute timestamps are more accurate than the DOS modification time, but you can slightly reduce the size of an archive by omitting this extended attribute information.

To exclude extended attribute information, use the *noextended* option, as in the following example:

```
pkzipc -add -noextended test.zip readme.doc
```

Note: The *noextended* option does not affect storage of the offline, temporary, and system attributes.

Extended Attributes and the OS

Extended attributes are automatically added to .ZIP archives when they are created. PKZIP does not display a message indicating that it is saving extended attributes.

PKZIP stores the following extended attributes:

- Create time
- Last modification time
- Last access time

Caution: Typically, PKZIP automatically extracts extended attributes with archived files and/or directories. Whether any existing files, directories and extended attributes on your system are replaced with those stored in the extracted archive depends on your file system privileges and the options and sub-options you use.

Extended Attributes and 204g Compatibility

204

By default, PKZIP does not enable PKZIP for DOS 2.04g compatibility. When 204g compatibility is enabled, extended attribute data is stored in both the Local header and Central header records. This will result in a slightly larger .ZIP file size, but improves the chance that extended attribute information can be recovered if the .ZIP file should become damaged. It also ensures the extended attribute information is always retained if the file is generated with a version of PKZIP other than 2.04g. This option is ignored when extracting. The *204* option also limits the number of files that can be added to a .ZIP archive to 16,383. To enable 204g compatibility, use the *204* option as in the following example:

```
pkzipc -add -204 test.zip *
```

Including Additional Information in a ZIP File

With PKZIP, you can include additional information in your .ZIP file, such as a "comment", to identify that .ZIP file.

You can include a:

- Text comment.
- Header comment.
- Date for the .ZIP file (other than the creation date).

Refer to the sections that follow for more information.

Including a Text Comment

comment

With PKZIP, you can include a comment for the individual files within a .ZIP file. There are several options for adding comments to your .ZIP files. To include a comment, use the *comment* option alone or with the *add* command. When you run the command, PKZIP prompts you to enter the comment.

The table below lists the available sub-options for adding comments to your .ZIP archives:

| <i>Sub-Option</i> | <i>To</i> | <i>For example</i> |
|-------------------|---|--|
| <i>all</i> | Comment all of the files and any new files added. | <i>pkzipc -add -comment=all test.zip *</i> |
| <i>unchanged</i> | Comment only files existing in the ZIP file that are not either updated or being added. | <i>pkzipc -add -comment=unchanged test.zip *</i> |
| <i>add</i> | Comment all files added. | <i>pkzipc -add -comment=add test.zip *</i> |
| <i>none</i> | Disable the comment option. | <i>pkzipc -add -comment=none test.zip *</i> |

| <i>Sub-Option</i> | <i>To</i> | <i>For example</i> |
|-------------------|--|--|
| <i>freshen</i> | Comment all of the files updated in the ZIP file. | <code>pkzipc -add -comment=freshen test.zip *</code> |
| <i>update</i> | Comment all files added and updated in the zip file. | <code>pkzipc -add -comment=update test.zip *</code> |

Note: Comment length is limited to 59 characters.

Including a Header Comment

header

With PKZIP, you can include a general comment for a .ZIP file. This is called a "header" comment because it appears in the header portion of a .ZIP file. This differs from the *comment* option in that the "header" comment applies to the entire .ZIP file, not to individual files within the .ZIP file.

Headers for .ZIP files are limited to 16K in size. PKZIP truncates headers larger than 16K.

To include a header comment, use the *header* option with the *add* command. PKZIP provides several ways to specify the comment. You can enter the comment with the *header* option, or you can specify a file that contains the comment.

To include the comment in the command line, specify the comment as a value for the *header* option. Enclose the comment text in quotes if the text includes spaces. For example:

```
pkzipc -add -header="This is the comment" test.zip *
```

If you include the *header* option alone, without a value, PKZIP prompts you for text to use, as follows:

```
Zip Header ?
```

Type your header comment and press ENTER.

To use header text from a file, specify the file name (and path, if necessary) as a value for the *header* option. Prefix the file name with the list character (@). Put the file name in quotes if it contains spaces. For example:

With this method, you type the *header=@filename.ext* option. If there are no spaces in the file name, it is not necessary to use quotation marks. For example:

```
pkzipc -add -header=@header.txt test.zip *
pkzipc -add -header=@"my header.txt" test.zip *
```

Specifying the Date of a .ZIP File

archivedate

When you create an archive file, PKZIP gives it the current date by default. You can specify a different date for the file by using the *archivedate* option with the *add* command.

Note: The *archivedate* option replaces the older *zipdate* option, which is now deprecated.

PKZIP provides several methods for applying a date to an archive file. The table below lists the available sub-options for applying date information to your archives:

| <i>Sub-Option</i> | <i>To use</i> | <i>For example</i> |
|--------------------------|--|---|
| <i>retain</i> | The date that the file was created. | <i>pkzipc -add -archivedate=retain test.zip *</i> |
| <i>none</i> (Default) | The current date. | <i>pkzipc -add -archivedate=none test.zip *</i> |
| <i>oldest</i> | The date of the oldest file within the archive file. | <i>pkzipc -add -archivedate=oldest test.zip *</i> |
| <i>newest</i> | The date of the newest file within the archive file. | <i>pkzipc -add -archivedate=newest test.zip *</i> |

Removing File Attributes

mask

If you use the *attributes* option to have PKZIP process files that have attributes, such as *hidden* or *system*, specified with the *attributes* option, you can use the *mask* option to strip those attributes from the files when they are archived or extracted.

You can only use the *mask* option with attributes specified with the *attributes* option. Attributes can be specified with this option either on the command line or as configured defaults.

The table below lists all of the available sub-options for masking file attribute information:

| <i>Sub-Option</i> | <i>To</i> | <i>For example</i> |
|-------------------|---|--|
| <i>hidden</i> | Remove the hidden file attribute from files. | <i>pkzipc -add -mask=hidden test.zip *</i> |
| <i>system</i> | Remove the system file attribute from files. | <i>pkzipc -add -mask=system test.zip *</i> |
| <i>readonly</i> | Remove the read-only file attribute from files. | <i>pkzipc -add -mask=readonly test.zip *</i> |
| <i>archive</i> | Remove the archive attribute from the file. | <i>pkzipc -add -mask=archive test.zip *</i> |
| <i>none</i> | Turn off file masking. | <i>pkzipc -add -mask=none test.zip *</i> |
| <i>all</i> | Remove all attributes from files. | <i>pkzipc -add -mask=all test.zip *</i> |

The *mask* sub-options can be used on the command line either individually or in a comma-separated list.

You may use a dash (-) before a *mask* sub-option on your command line to preserve a file attribute being added or extracted with a file, regardless of the default *mask*

configuration setting. For example, if the default *mask* configuration is set to *all*, you can enter the following command line to preserve the *hidden* attribute associated with any of the files to be added:

```
pkzipc -add -mask=-hidden test.zip
```

Sorting Files Within a .ZIP File

sort

With PKZIP, you can sort the files in an archive in several ways. If you do not change the sort order, the files are automatically sorted in the order in which they were compressed into the archive. This is called the "natural" order.

The *sort* option works with *add*, *extract*, *test*, and *view*. The value you include with *sort* depends on the command you select.

| <i>Sub-Option</i> | <i>To sort by</i> | <i>For example</i> |
|-------------------|---|--|
| <i>date</i> | File date. | <i>pkzipc -add -sort=date temp.zip</i> |
| <i>size</i> | Original uncompressed size of the file ("length" in display). | <i>pkzipc -add -sort=size temp.zip</i> |
| <i>extension</i> | File extension. | <i>pkzipc -add -sort=ext temp.zip</i> |
| <i>name</i> | Sorts files and folders by name in a single series. (Contrast with <i>-sort=none</i> .) | <i>pkzipc -add -sort=name temp.zip</i> |
| <i>none</i> | Groups folders first, sorted by name, and then groups files, sorted by name. (The default.) | <i>pkzipc -view -sort=none temp.zip</i> |
| <i>natural</i> | Preserves the order in which files were added to an archive. | <i>pkzipc -view -sort=natural temp.zip</i> |
| <i>ratio</i> | Ratio of uncompressed size to compressed size. | <i>pkzipc -view -sort=ratio temp.zip</i> Note: The ratio sub-option will not work with the add command. |
| <i>crc</i> | CRC (Cyclic Redundancy Check) number. | <i>pkzipc -view -sort=crc temp.zip</i> Note: The crc sub-option will not work with the add command. |
| <i>comment</i> | File comment. | <i>pkzipc -view -sort=comment temp.zip</i> Note: The comment sub-option will not work with the add command. |

The *name* sub-option sorts entire path names; it does not sort file names directly if folder information is present.

For example, the *name* sub-option sorts the two files *abacus.txt* and *zebra.txt* as follows if they are added to an archive without including any path or folder information:

```
abacus.txt
zebra.txt
```

However, if the files are added with folder information, the name of the outermost folder in the path determines their order of appearance. This is because *name* sorts the entire path name whether or not it includes folder names. For example:

```
all\junk\zebra.txt
everything\important\abacus.txt
```

By contrast, the *none* sub-option groups path names that contain folder names and sorts this group in a separate series from file names that do not include folder information. The names below are sorted by *none*:

```
all\junk\zebra.txt
everything\important\abacus.txt
anotherfile.txt
lonelyfile.doc
somepix.gif
```

If no *sort* option is specified, files are sorted as if *sort=none* was specified (unless you have changed configuration defaults).

If you specify the *sort* option on your command line but do not specify a sub-option value, the *name* sub-option is applied.

Note: Using the *sort* option with the *add* command only works on new archive files. It does not work with an archive that is being updated.

Moving Files to a .ZIP File

move

Normally, when you compress files, you end up with two copies of each file: the original file and the compressed file. With PKZIP, you can choose to remove the original file "after" you compress it into the .ZIP file.

If you want to move only specific files, you must compress them separately since you can only move all or none of the files that you are compressing.

To move files, use the *move* option with the *add* command, as shown below:

```
pkzipc -add -move test.zip *.doc
```

This sample command line tells PKZIP to compress and add to archive *test.zip* all files that end in *.doc* and then to delete the original files.

CAUTION: Like any operation that deletes files, the *move* option should be used with care.

Shredding Deleted Files

shred

A deleted file still remains on your disk and can often be fully or partly recovered. So can the temporary files that PKZIP creates when updating an archive. To erase these

files to prevent information from being retrieved from them, use the *shred* option with the *add* command. Shredding a file overwrites the file's data so that it cannot be read.

Shredding overwrites these files:

- Deleted originals that have been moved into an archive with the *move* option
- Temporary files that contain the previous version of an archive that has just been updated

Note that overwriting files with the *shred* option takes some additional time.

Shredding can overwrite files only if the file system applies the overwriting to the same physical disk sectors that the file to be overwritten used. Most UNIX and Linux file systems do not do this. For this reason, shredding works most reliably on Windows.

Shredding has a couple of other constraints:

- Files on the Windows NTFS file system that have been encrypted or compressed by NTFS itself have a special NTFS attribute. PKZIP cannot shred these files.
- The system temporary folder must be local; it cannot be on a removable or network drive for shredding to work. PKZIP can delete files that are on a removable or network drive but cannot shred them.

The *shred* option has these sub-options:

| <i>Sub-Option</i> | <i>Description</i> |
|-------------------|---|
| <i>None</i> | Turns shredding off if it is configured on |
| <i>Random</i> | Overwrites files once with random data (the default) |
| <i>Dod5220</i> | Overwrites files three times, to the DOD 5220.22-M specification |
| <i>NSA</i> | Overwrites files seven times, to the NSA standard. (Takes much longer.) |

For example:

```
pkzipc -add -move -cryptalgorithm -passphrase -shred=NSA
secret.zip *.*
```

Working with Self-Extracting (PKSFX) Archives

sfx

If you have the PKZIP Self-Extractor add-on, you can use PKZIP to create PKSFX archives. A PKSFX archive is self-extracting: it has an `.exe` file name extension (instead of `.zip`, for instance), and it can be extracted just by executing it, even by someone who does not have PKZIP or another ZIP utility. (PKSFX archives are also called self-extractors or SFX files, for short.)

Note: You must have PKZIP Enterprise or SecureZIP to create a PKSFX archive.

You can create self-extractors of two general types:

- A native command line self-extractor for use in the command line environment of the operating system on which PKZIP is running. The native command line self-extractor extracts without using any graphical user-interface features such as dialog boxes.
- A graphical 32-bit Windows self-extractor for use in the graphical Windows environment. When run, a graphical Windows self-extractor opens a dialog that contains controls to view progress or set options for extracting files.

To create a self-extracting archive, use the *sfx* option with the *add* command. For example, the following line creates a native command line self-extractor `mysfx.exe`:

```
pkzipc -add -sfx mysfx *.doc
```

When used without a sub-option, the *sfx* option creates a native command line self-extractor by default.

Use the *listsfxtypes* command to list *sfx* sub-options for the types of self-extractors available to you. The exact types vary with your system and license. For example, the following command

```
pkzipc -listsfxtypes
```

may produce a display like this:

The SFX sub-option choices are:

```
WIN32_X86_C1230 - V12.30 Command Line SFX for Windows on X86
WIN32_X86_G1230 - V12.30 Windows SFX for Windows on X86
```

In the list above, *win32_x86_c...* designates the native Windows command line self-extractor, and *win32_x86_g...* designates the graphical Windows self-extractor. The digits at the end give the version number.

To create a graphical Windows self-extractor, use the *sfx* option with the *win32_x86_g1230* sub-option. For example:

```
pkzipc -add -sfx=win32_x86_g1230 mysfx *.doc
```

You only need to enter enough of the name of an SFX type to uniquely identify it; you can leave off the version number at the end:

```
pkzipc -add -sfx=win32_x86_g mysfx *.doc
```

You can also use *sfx* as a command to convert an existing, ordinary ZIP file to a self-extractor. To do so, use the *sfx* command by itself on the command line, without the *add* command, and specify the ZIP file to convert. For example:

```
pkzipc -sfx=win32_x86_g1230 myfiles.zip
```

Notes:

- You cannot use the *sfx* option with the *cd* option to create or convert an archive with encrypted file names
- The *sfx* command can only convert ZIP archives that are physical files. It cannot convert ZIP archives that are special files (named pipes, sockets) or are presented from STDIN.

Converting a Standard Archive to a Self-Extractor

To convert a standard ZIP file to a self-extracting archive, use the *sfx* command, without the *add* command.

For example, the following command line converts standard archive `test.zip` to self-extractor `test.exe`. PKZIP replaces `zip` in the file name with `exe`.

```
pkzipc -sfx test.zip
```

Converting to a Self-Extractor with a Different Name

Ordinarily, when you use the `sfx` command to convert a standard archive to a self-extracting archive, the archive keeps its original name except for the extension, which PKZIP changes from `zip` to `exe`. To give an archive a different name, use the `namesfx` option to specify a new name when you convert the archive:

```
pkzipc -sfx -namesfx=test123.exe test.zip
```

If you omit the `.exe` in the new name, PKZIP supplies it.

Note: You cannot use the `sfx` option with the `cd` option to create or convert an archive with encrypted file names.

Options for Creating Self-Extractors

You can use the following options together with the `sfx` command/option to customize a self-extractor in various ways when you create it. The options are described in the following sections. Default values for all the options can be configured with the `configuration` command.

As indicated in the table below, some of the options require a GUI (that is, a graphical) self-extractor and do not work with command line self-extractors.

| Option | Works only with GUI Self-Extractors |
|----------------|-------------------------------------|
| SFXDestination | X |
| SFXDirectories | X |
| SFXLogfile | |
| SFXOverwrite | X |
| SFXUIType | X |
| RunAfter | |

SFXDestination

The `SFXDestination` option specifies a default target folder for extracted files. For example:

```
pkzipc -add -sfx=win32_x86_g -sfxdestination="My
Documents\newstuff" mysfx *.doc
```

If no drive letter is listed in the path, the self-extractor chooses the drive that contains the temporary folder and appends the path to the temporary folder.

If the specified destination folder or path does not exist, the self-extractor prompts the user whether to create it.

The `SFXDestination` option works only with a GUI self-extractor.

SFXDirectories

The *SFXDirectories* option causes the self-extractor to restore saved directory paths on extraction. To recurse subdirectories and save path information (relative to the current directory) when you add files to a self-extractor, use the *directories* option.

For example, the following command line archives the `docs` folder and all its files and subfolders. The `docs` folder and the saved subfolders are restored on extraction.

```
pkzipc -add -sfx=win32_x86_g -sfxdirectories -directories mysfx
"docs\*.*"
```

The *SFXDirectories* option works only with a GUI self-extractor.

SFXLogfile

The *SFXLogfile* option creates an ASCII text SFX error log named `pkerrlog.txt` in the destination directory on extraction.

```
pkzipc -add -sfx -sfxlogfile test.exe *.doc
```

SFXOverwrite

The *SFXOverwrite* option specifies when the self-extractor overwrites files that have the same name as a file being extracted. The option has the sub-options listed in the table below.

| <i>Sub-option</i> | <i>Description</i> |
|-------------------|---|
| <i>prompt</i> | (Default) The user is asked whether to overwrite files |
| <i>always</i> | Files that have the same name in the destination folders are overwritten without prompting |
| <i>update</i> | Only files that do not already exist or are newer than same-named files |
| <i>freshen</i> | Only newer versions of files that already exist in the destination folders are extracted; the older files are overwritten without prompting |
| <i>never</i> | Files are never overwritten |

For example:

```
pkzipc -add -sfx=win32_x86_g -sfxoverwrite=freshen mysfx *.doc
```

The *SFXOverwrite* option works only with a GUI self-extractor.

SFXUIType

The *SFXUIType* option specifies the type of graphical interface that the self-extractor presents to the user. This option only affects GUI self-extractors. (Command line self-extractors do not present a GUI.) The option has the sub-options listed in the table below.

| <i>Sub-option</i> | <i>Description</i> |
|-------------------|--|
| <i>AutoSFX</i> | Presents a dialog that displays a bar to show progress extracting, and a Cancel button |

| <i>Sub-option</i> | <i>Description</i> |
|-------------------|---|
| <i>EasySFX</i> | (Default) Presents a dialog that enables the user to select a destination folder and to turn off any <i>runafter</i> option set. (See “Run Programs with the Self-Extractor,” below.) |
| <i>RegularSFX</i> | Presents a dialog that enables the user to change the destination folder and other options before the archive is extracted |

For example:

```
pkzipc -add -sfx=win32_x86_g -sfxuftype=regularsfx mysfx *.doc
```

Run Programs with the Self-Extractor

Use the *runafter* option with the *sfx* option to create a self-extracting archive that runs a program after the self-extractor is run. This option enables you to create a self-extractor that runs a script or opens a file after the contents of the self-extractor are extracted.

The *runafter* option does not work with the following types of self-extractors:

- DOSJR_X86_C250 - 2.04g compatible SFX Junior for DOS
- DOS_X86_C250 - 2.04g compatible SFX for DOS

Use the *listsfxtypes* command to list the types of self-extractors available to you:

```
pkzipc -listsfxtypes
```

Here are examples showing uses of the *runafter* option.

Create a self-extractor to open a `readme.txt` file after extraction:

```
pkzipc -add -sfx -runafter="notepad.exe readme.txt" test.exe *
```

Create a self-extractor to open a file by means of its associated application:

```
pkzipc -add -sfx -runafter="{readme.txt}" test.exe *
```

Create a self-extractor to run an install script:

```
pkzipc -add -sfx -runafter="{install}install.inf" test.exe *
```

Create a self-extractor to run an install script, with the full path prepended (%0):

```
pkzipc -add -sfx -runafter="{install}%0install.inf" test.exe *
```

Extraction Options for the Native Self-Extractor

To extract files from a self-extracting archive, you run the archive. For example, to extract files from self-extractor `test.exe`, use the following command line:

```
test.exe
```

Note: When extracting encrypted files on UNIX systems from a self-extracting archive, you may encounter a “Recipient not found” error message. This results from a change in the certificates database file in SecureZIP Server version 14.0. You should be able to extract the file as an ordinary ZIP archive using the *noarchiveextension* command:

```
pkzipc -extract -noarchiveextension test.exe
```

When you run a native command line self-extractor, you can use the command line options listed below. The options can be used only with a native self-extractor; they cannot be used with a Windows graphical self-extractor:

| | | |
|-------------|------------|-----------|
| after | locale | silent |
| before | lowercase | smaller |
| console | mask | sort |
| directories | more | test |
| exclude | newer | times |
| extract | noextended | translate |
| help | older | version |
| include | overwrite | warning |
| larger | passphrase | |
| license | print | |

For example, the following command line excludes all text (.txt) files from the set of files to be extracted:

```
test.exe -exclude="*.txt"
```

4

Extracting Files

This chapter describes the options PKZIP offers for extracting files from archives. These options give you various ways to choose what files to extract and where to extract them to and help you manage every aspect of extracting files.

Default Values for Commands and Options

Commands and options that have sub-options generally have a default value. This is the sub-option value that is used if none is explicitly specified on the command line. For example, the default behavior for the *extract* command is to unzip or uncompress all files in an archive. This behavior is set with the *all* sub-option of the *extract* command.

See Chapter 7 for information on configuring default sub-option values for commands and options.

Extracting New and Existing Files

When you extract files from a .ZIP file, you can select those files you wish to extract and those you do not. If the directory into which you extract the files contains files that have the same name as those being extracted, you have to decide if you want to overwrite those files.

PKZIP provides several ways to choose which files to extract. You can extract:

- All files in an archive (the *all* sub-option)
- Files that are not in the target extract directory plus files that are more recent versions of files that are in the extract directory (the *update* sub-option)
- Only files that are more recent versions of—that is, have the same names as—files that are already in the extract directory (the *freshen* sub-option)

Extracting All Files from an Archive

extract=all

To extract all files from an archive file, type *pkzipc -extract* and the name of your archive file, as shown below:

```
pkzipc -extract test.zip
```

In this example, all files in the archive are extracted into the current directory.

By default, *extract* uses the *all* sub-option; you do not need to specify this sub-option unless you have changed the default for *extract* to some other sub-option.

The following example explicitly specifies the sub-option. This command does the same thing as the first example but also overrides any changed default setting. The override applies only to this instance of the command; it does not reset the default you have defined.

```
pkzipc -extract=all test.zip
```

Extracting Newer Versions of Existing Files and New Files

extract=update

Update extracts to the target extract directory only files that are not already in the directory or are newer versions of files that are already there. Archive files that are older versions of files already in the directory are not extracted.

```
pkzipc -extract=update test.zip
```

Extracting Only Newer Versions of Files

extract=freshen

Freshen extracts only files that are newer versions of files that already exist in the target extract directory. It does not add any files to the directory that are not already there in an earlier version.

```
pkzipc -extract=freshen test.zip
```

Checking for Viruses when Extracting

avscan, avargs

PKZIP can use your anti-virus program to scan for viruses when you extract files.

The *avscan* option controls whether extracted files are scanned for viruses and specifies the anti-virus program to run to do scans.

When you extract with the *avscan* virus scanning option turned on, PKZIP first extracts the specified files and then runs the anti-virus program to recursively scan all files in the specified destination directory and its subdirectories. PKZIP relays to you any messages returned by the virus scanning program.

If your virus scanner is set up to scan files dynamically as they are read or written, you do not need launch a virus scan from PKZIP. Your virus scanner will automatically scan the files as they are extracted.

How your anti-virus program deals with files infected by a virus is determined by the way the program is configured and by the arguments, if any, included in the PKZIP command line used to run the scanner. The contents of the command line used to run the scanner and the arguments that may be available for it depend on your anti-virus program.

Use the PKZIP *avargs* option to specify any anti-virus command line arguments. To tell the anti-virus program what directory to scan, include the variable %e. PKZIP replaces this variable with the full path to the extraction directory before passing the command line to the anti-virus program.

The following example shows *avscan* used to run a virus-scanning program. The variable %e and arguments for the virus-scanning program's command line are given in the *avargs* option.

```
pkzipc -extract -avscan=f-prot.exe -avargs="%e /silent /nomem
/noboot" myfiles.zip
```

In *avscan*, specify the full path to the anti-virus program if the executable is not on the search path.

PKZIP assumes that the anti-virus program will not launch any graphical interfaces that require user interaction and that the program will automatically clean up any viruses that it finds.

Most virus scanning programs return a value of 0 when a scan completes successfully and finds no viruses. If a program returns any other value as the result of a scan, PKZIP issues a warning that some of the extracted files may not have passed the scan.

Both *avscan* and *avargs* can be configured for use by default. Configuring *avscan* causes PKZIP to do virus scans by default whenever files are extracted, using the specified anti-virus program executable and whatever anti-virus command line arguments, if any, are given in *avargs*.

Extracting from an Archive Embedded in an Archive

embedded

An archive can contain other archive files. For example, a ZIP file can contain other ZIP archives, or a GZIP archive might contain a TAR archive. Such contained archives are said to be *embedded* in the archive that contains them.

If PKZIP encounters a lone embedded archive file in another archive whose contents PKZIP is extracting, PKZIP prompts you whether you would like to extract the contents of the embedded archive or just the archive itself. For example, if PKZIP is extracting the contents of *outerarchive.zip*, and *outerarchive.zip* contains *innerarchive.zip*, PKZIP asks you whether you want to extract *the files* in *innerarchive.zip* or just *innerarchive.zip* itself.

The *embedded* option can be used with *extract* to tell PKZIP to omit the prompt and just go ahead and extract the files contained in any lone archive file embedded in an archive of the specified type. You must specify the type of the outer, container archive for which you want to extract files from embedded archives.

For example:

```
pkzipc -extract -embedded=zip outerarchive.zip
```

In the example, if *outerarchive.zip* contains a single embedded archive (it may also contain non-archive files), PKZIP extracts the files from the embedded archive instead of extracting the embedded archive itself, and does not prompt.

The *embedded* option can be configured to operate by default. For example, the following command line configures *embedded* so that files are routinely extracted from single archives (such as `.tar` archives) embedded in `.gz` files:

```
pkzipc -config -embedded=gz
```

Put a hyphen in front of the *embedded* sub-option to tell PKZIP *not* to prompt or extract the contents of an embedded archive in an archive of a specified type. A command line containing a hyphenated sub-option overrides a configured setting. For example, the following command line extracts only an embedded archive, not its files:

```
pkzipc -extract -embedded=-gz outerarchive.gz
```

Note that PKZIP extracts the contents of an embedded archive, with or without prompting, only if that archive is the *only* embedded archive in the outer archive file. If the outer archive file contains multiple embedded archives, the embedded archive files themselves are extracted.

Extracting Passphrase-Protected Files

To extract files from a passphrase-protected archive, use the *extract* command with the *passphrase* option.

- Type the passphrase (preceded by an equal sign) as part of your command. For example:

```
pkzipc -extract -passphrase=mysecret test.zip
```

If the passphrase is correct, the files are extracted (to the current directory, by default). If the passphrase is incorrect, PKZIP displays a warning message:

```
PKZIP: (W20) Warning! Incorrect passphrase for file: filename.ext
```

Re-type your command line with the correct passphrase.

- If you specify the *passphrase* option without a passphrase, PKZIP prompts for a passphrase. For example:

```
pkzipc -extract -passphrase test.zip
```

When you press ENTER, a prompt appears:

```
Passphrase?
```

Type the passphrase. The characters appear on the screen as asterisks, for security. Press ENTER. If you specified the correct passphrase, the files will be extracted to the current directory. If the passphrase you entered is incorrect, a warning message displays:

```
PKZIP: (W20) Warning! Incorrect passphrase for file: filename.ext
```

Retype your command line and when prompted enter the correct passphrase.

- If you do not specify the *passphrase* option when extracting an archive that contains passphrase-protected files, PKZIP warns that the encrypted files are being skipped, and the files are not extracted.

| |
|---|
| <p>Note: Many other ZIP utilities can decrypt archives encrypted with traditional ZIP encryption. When a ZIP utility that can read strongly encrypted ZIP files is not available, use ZIP Reader</p> |
|---|

Note: Passphrases are case sensitive.

Note: For greater security, enter passphrases at the prompt so that asterisks hide the characters you are entering. For information on using passphrases in scripts, see Appendix **Error! Reference source not found.**

Extracting an Archive on STDIN or a Special File

Ordinarily, when you use the *extract* command to extract files from an archive, you extract the files from a physical archive file. For example, the following command line extracts all *.txt* files from the archive *myfiles.zip*:

```
pkzipc -extract myfiles.zip *.txt
```

PKZIP can also extract files from an archive that is not a physical file but is presented from an input source such as STDIN or a named pipe.

Note: Some options are not supported when extracting from an archive that is not a physical file. In particular:

- Signatures (added with the *sign* option) on either files or the archive central directory are not processed.
- Because signatures are not processed, the *verifysigner* extraction option always fails. (This option requires verification that an archive was signed using a specified certificate.)
- File name encryption (*cd* option) is not supported.

Extracting from an Archive on STDIN

You can specify STDIN (*standard input*) instead of a physical file as the location or source of an archive from which to extract files. To do so, use a hyphen “-” in place of the name of an archive file. In a command line with the *extract* command (or the *test* or *view* command), the hyphen tells PKZIP to read the archive from STDIN. For example:

```
pkzipc -extract -noarchiveextension -silent=input -
```

The *noarchiveextension* option is needed so that PKZIP does not take the hyphen as a file name and try to extract from a file named *-.zip*. If the archive is not a ZIP archive, use the *archivetype* option to specify its type. For example, the following command line tells PKZIP that the file is a BZIP2 archive:

```
pkzipc -extract -archivetype=bzip2 -noarchiveextension -
silent=input -
```

The option *silent* is set to the *input* sub-option to suppress any PKZIP requests for input (a passphrase, for example). If input is needed, the extraction fails with an error.

See “Writing an Archive to STDOUT” in Chapter 0 for a way to create an archive that is presented through STDIN.

Extracting an Archive from a Named Pipe

You can specify a named pipe instead of a physical file as the location of an archive from which to extract files. The pipe or socket must first be created, perhaps by

another program, and an archive must be written to it. To extract, use the name of the pipe in the command line in place of the name of an archive file. For example:

```
pkzipc -extract -noarchiveextension <name of pipe>
```

As when extracting from STDIN, you must use the *noarchiveextension* option to prevent PKZIP from trying to extract from a .zip file—in this case, one named for the pipe or socket.

PKZIP tries to extract from ZIP-format archives by default. To extract from a different type of archive, use the *archivetype* option to specify the type. For example, the following command line tells PKZIP that the archive is a BZIP2-format file:

```
pkzipc -extract -archivetype=bzip2 -noarchiveextension <name of pipe or socket>
```

You must use the full UNC path when referring to a named pipe on Windows. For example:

```
pkzipc -extract -noarchiveextension \\.\pipe\mypipe *.doc
```

In the preceding example, the dot in the path

```
\\.\pipe\mypipe
```

references the current machine. To reference a pipe on a different machine—named *boulder*—specify the machine.

```
\\boulder\pipe\mypipe
```

You can use either a name or an IP address to specify a machine.

You must use the *noarchiveextension* option to prevent PKZIP from trying to extract from an archive file named .zip.

substitution

With the *substitution* option, you can extract the contents of an archive to a folder whose name and path are constructed on the fly from tokens embedded in the specification for the destination folder on the command line. PKZIP creates the actual name of the folder by substituting values for the tokens when the archive is extracted. Tokens are supplied that enable you to name the folder after the archive to be extracted to it, replicate the path to the archive, and embed timestamp elements.

With this option, you can use a single command line to extract multiple archives each to its own custom-named folder.

The table below lists the tokens for use with the *substitution* option when extracting.

| Token | Replaced by |
|---------------|---|
| {archivename} | Base name of archive, without the extension |
| {archiveext} | The file name extension of the archive |
| {archivepath} | The path of the archive, without the file name, preceded by a slash or backslash and excluding the drive letter or share path if the name is a UNC name |
| {id} | A job ID specified separately with the <i>jobid</i> 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); no leading 0 |
| {dd} | Day, 2-digit |
| {d} | Day, 1-digit (if possible); no leading 0 |
| {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); no leading 0 |
| {SS} | Second, 2-digit |
| {S} | Second, 1-digit (if possible); no leading 0 |
| {ampm} | a.m. or p.m. indicator to identify current 12-hour segment of the day |

The following command line shows a straightforward example of the *substitution* option. The command line extracts all ZIP files in the current directory, each to a subdirectory named after the ZIP archive extracted there. If two ZIP files, `myfiles.zip` and `myfiles2.zip`, are in the current directory, the command line extracts them to subfolders named `myfiles` and `myfiles2`, respectively.

```
pkzipc -extract -substitution *.zip {archivename}\
```

The example below uses the `{archivepath}` token to specify the archive path for the destination folder. The `{archivepath}` token includes a leading backslash (or slash). The command line extracts all ZIP files in folder `\home\thomas\` each to its own subfolder in `other\location\home\thomas\`. For example, it extracts `myfiles.zip` in folder `\home\thomas\` to subfolder `other\location\home\thomas\myfiles`.

```
pkzipc -extract -substitution \home\thomas\*.zip
\other\location{archivepath}\{archivename}\
```

If run from `C:\myproject`, the command line below extracts all ZIP files to `C:\myproject\test`. The dot in the specification for the target folder locates the start of the extraction path in the current folder. The drive letter is stripped.

```
pkzipc -extract -substitution D:\test\*.zip .{archivepath}\
```

If the date is July 31, 2008, and the directory `C:\app1\` contains `myfiles.zip` and `test2.zip`, the command line below extracts `test1.zip` to folder `test1-07312008` and `test2.zip` to folder `test2-07312008`:

```
pkzipc -extract -substitution C:\app1\*.zip {archivename}-
{mm}{dd}{yyyy}\
```

The following example shows how *{archivepath}* strips out a share path. If `\\server\share\path\to\zips` contains `test1.zip` and `test2.zip`, and the current directory is `d:\testme`, the command line extracts `test1.zip` to `d:\path\to\zips\test1` and extracts `test2.zip` to `d:\path\to\zips\test2`:

```
pkzipc -extract -substitution \\server\share\path\to\zips\*.zip
{archivepath}\{archivename}\
```

The example below uses the substitution option when extracting an archive from STDIN, represented by a hyphen (-) in the command line (see “Extracting an Archive on STDIN or a Special File”). If the date is July 31, 2008, an archive provided on STDIN is extracted to directory `\-07312008`. In this case, *{archivepath}* and *{archiveext}* are replaced with nothing, and *{archivename}* is replaced with a hyphen.

```
pkzipc -extract -substitution -noarchiveextension -
{archivepath}\{archivename}{archiveext}{mm}{dd}{yyyy}
```

The *substitution* option can also be used with the *add* command and a slightly different set of tokens to insert a timestamp in the name of a newly created or updated archive. See “Inserting a Timestamp in the Archive File Name” in Chapter 6.

Extracting Files in Lower Case

lowercase

The *lowercase* option allows you to extract files in lower case regardless of how the file name was originally archived. To force the file names to be extracted in lowercase, use the following example:

```
pkzipc -extract -lowercase test.zip
```

Preserving File Times

times

The *times* option allows you to preserve the access, creation and modification times of the extracted files. Specify the sub option *all* to preserve all times, use *access* to preserve the access times only, use *modify* to restore the time of last modification times or *create* to restore the creation times.

To preserve all the file times, use the following example:

```
pkzipc -extract -times=all test.zip
```

Retaining Directory Structure while Extracting

directories

If you stored directory path information within a .ZIP file, you can re-create those directory paths when you extract the files. For example, if you compressed a file called `apples.doc` in the `temp/fruit` directory, and you stored `temp/fruit` you can re-create `temp/fruit` in the location in which you extract the files.

To re-create directories, use the *directories* option with the *extract* command, as in the following example:

```
pkzipc -extract -directories test.zip
```

When you use this command, all directories that were stored in the .ZIP file will be retained during extraction. The directory path stored is appended to the directory in which you extract the files. For example, if your extract directory is /doc, and a directory path stored with the files is temp/fruit, the files would now be extracted to /doc/temp/fruit.

Retaining Zone Identifier Information for Downloaded Files

zoneidentifier

When you download a file from any other computer with Microsoft Internet Explorer, the browser attaches “security zone” information about the computer hosting the file. These zones are labeled Local Intranet, Trusted Sites, Internet, and Restricted Sites. As a result of this “zone identifier,” you may receive a warning about files received from the Internet from Windows before you open or activate the file, depending on your Internet Options settings.

By default, PKZIP does not retain this information when you extract files from an archive that contains this information. The *zoneidentifier* command allows you to preserve that information if you are extracting from an NTFS-formatted drive to another NTFS-formatted drive.

Note: Only NTFS volumes can preserve and process Zone Identifier information. Volumes created with FAT (the default file system for Windows 98 and earlier), or UNIX-based file systems will not preserve this alternate stream. This would include files saved in IE to temporary directories on non-NTFS systems.

To specify that all extracted files in MyDownloadedFiles.zip retain its Zone information, type:

```
pkzipc -extract -zoneidentifier MyDownloadedFiles.zip
```

To configure PKZIP to preserve the Zone information whenever possible, use this command:

```
pkzipc -config -zoneidentifier=enable
```

Sorting Files in the Extract Directory

sort

PKZIP allows you to specify the sort order of files that are compressed in a .ZIP file or extracted into a destination directory. For example, if you wish to extract files in a specified sort order (by date), you would type the following and press ENTER:

```
pkzipc -extract -sort=date test.zip
```

In this example, all files that exist in the test.zip file are extracted into the current directory sorted in ascending order by date. For more information on sort options, see Appendix A.

Extracting Files Only for Display

console

PKZIP gives you the option of displaying specific files contained in a .ZIP file to your computer monitor. For example, if you wish to view the contents of all of the .txt files contained in a .ZIP file, type the following and press ENTER:

```
pkzipc -console test.zip *.txt
```

In this example, all files with a .txt extension that exist in the test.zip are displayed on the monitor. Since many .ZIP files contain an information document (e.g., readme.txt), the *console* option is a good way to determine the contents of a .ZIP file without requiring you to extract a file or file(s) to your hard drive.

Note: You can also use the *console* and *silent* options to redirect files to pipe files directly to another program on Windows XP (and later) systems.

Extracting Files with a List File

You can use a list file to specify files to extract from an archive. In the list file, specify file and path name information to identify the target files. You can explicitly list individual files to extract, or you can use wild card characters (*, ?) to specify multiple files in a single entry. For example, entries like the four below are permitted:

```
Fred\My Documents\tmp\yparent\ychild\ychild1.txt
Documents and Settings\Fred\My Documents\tmp\yparent\*.txt
dparent?.txt
*.xls
```

How you identify files in an archive depends on the path information that was archived with them. In an archive, path information is treated as part of a file name for purposes of identification. So *d*.txt* does not just get all .txt files whose names start with *d* in the root folder of an archive; it gets all .txt files whose *pathname* starts with *d*. For example, it would get these files:

```
Documents and Settings\Fred\My Documents\tmp\yparent\*.txt
dparent?.txt
```

Do not use drive letters in a list file used to extract. Drive letters are not saved with other path information in an archive and are not allowed in extraction list file entries.

To specify a list file to use to extract, prefix the pathname of the list file with the @ character on the command line after the name of the archive. For example, the following line extracts using list file *mylist.txt*:

```
pkzipc -extract test.zip @tmp\mylist.txt
```

See the *listfile* option for information on using this option to create a list file. See the *view* option for information on viewing path information saved in an archive.

Authenticating Digital Signatures

When you extract files from an archive or test an archive with the *test* command, PKZIP authenticates any digital signatures attached to the files or the archive. A digital signature, like a pen-and-ink signature, warrants that the signed item really comes from the signer and has not been changed.

You can use the *test* command on an archive to check for a signature before extracting files. Testing tells you whether files are signed, authenticates any signatures, and gives you information about certificates used to sign files. PKZIP authenticates signatures automatically when extracting.

Use the *cr1* option to have PKZIP check an accessible certificate revocation list (CRL) to see if a certificate used for signing has been revoked. (See “Checking for Revoked Certificates” in Chapter 5.)

Signatures can be applied to particular files and/or to the central directory of an archive (that is, to the archive itself).

The following table lists warning messages that can appear when you test or extract signed files, causing PKZIP to authenticate signatures.

| <i>Message</i> | <i>Explanation</i> | <i>What to do?</i> |
|-----------------------------------|---|--|
| <i>Signature is invalid</i> | The file or archive has changed since it was signed. The archive may be corrupt. | You may want to try to obtain the file again (for example, download the file again from the Web site). Contact the archive creator as the file/archive has been compromised. If the file was downloaded from a Web site, you may want to contact a person at that company about the file. If a file has an invalid signature, then the file may have been modified. If the central directory has an invalid signature, then file(s) have been modified, added or deleted from the archive since the archive was signed. |
| <i>Certificate is not trusted</i> | The certificate used to sign is currently not to be trusted. | This message indicates that the certificate is not to be trusted, but there may be no problem with the archive. Contact the issuer of the certificate to validate the certificate/signature. |
| <i>Certificate is expired</i> | The certificate has expired (perhaps because the archive was signed a long time ago). | Contact the owner of the certificate. This message indicates that the certificate is not to be trusted, but there may be no problem with the file or archive. |
| <i>Certificate is revoked</i> | Indicates the issuer has revoked the certificate. | Contact the issuer or owner of the certificate. This message indicates that the certificate is not to be trusted, but there may be no problem with the file or archive. |
| <i>Certificate not found: XXX</i> | The certificate for the signature could not be found on your system. | Check to see if the certificate name was misspelled. Confirm that the certificate is on the system. |

Extracting Only Trusted Archives

verifysigner

With the *verifysigner* option, you can set PKZIP to extract an archive only if the archive is signed using one of a specified set of certificates. If the *verifysigner* option is used, PKZIP will extract an archive only if these two conditions are met:

- The archive central directory is signed using a certificate specified with the option
- PKZIP can find a copy of each certificate specified with the option, containing the public key, in the local store

For example, the following command line extracts only if the archive is signed by John Smith, and PKZIP can find the certificate used to sign:

```
pkzipc -extract -verifysigner="John Smith" important.zip
```

You can use the option multiple times in the same command line to specify more than one acceptable, trusted signer:

```
pkzipc -extract -verifysigner="John Smith" -verifysigner="Jane Doe" important.zip
```

The command line above extracts if the archive is signed by *either* John Smith or Jane Doe, but certificates for *both* John Smith and Jane Doe must be found.

The requirement that PKZIP be able to find a copy of a signer's certificate locally (or on a directory server) ensures that the signer is the person you think he is. If PKZIP only authenticated the signature without also checking its certificate, you would know that an archive really was signed by someone named John Smith, but you would not know if this John Smith is the same John Smith whose certificate you have.

Specifying Trusted Signers

You can specify a list of trusted certificates/signers either by specifying each certificate individually on the command line or by specifying a file that contains a list.

By default, PKZIP searches for certificates for listed recipients only in the system's local certificate stores.

Specifying Trusted Signers Individually

You can specify a trusted signer using any of the following criteria:

| Criterion | To use | For example |
|-------------|---|---|
| Common name | Specify, in quotes, the common name of the subject of the certificate (that is, the <i>cn</i> field in a string representation of a certificate); optionally, precede with: cn= By default, SecureZIP searches for certificates by common name unless another sub-option is used or the value appears to be an email address. | <i>-verifysigner=cn="John Public"</i> <i>-verifysigner="John Public"</i> |

| <i>Criterion</i> | <i>To use</i> | <i>For example</i> |
|----------------------|--|---|
| <i>Email address</i> | <p>Specify the email address of the certificate (that is, the <i>e</i> field in a string representation of a certificate); optionally, precede with:</p> <p style="text-align: center;">e=</p> <p>SecureZIP automatically looks for an email address if the string contains an @ and a dot and looks like an email address.</p> <p>Note that a certificate must contain an email address in order to be found by this method. Not all certificates embed an email address.</p> | <p>- <code>verifysigner=e=john.public@xyz.com</code></p> <p>- <code>verifysigner=john.public@xyz.com</code></p> |

Specifying a File That Lists Trusted Signers

PKZIP can extract a list of certificates from these kinds of files:

- An ordinary text file that lists the common name of each certificate on a line by itself

To use the *verifysigner* option to specify an ordinary text file list as a sub-option, prefix the file name with the listfile character (@, by default):

```
pkzipc -extract -verifysigner=@my_list_file.txt test.zip
```

- A PKCS#7 or PKCS#12 file: These kinds of files can contain one or more actual certificates. PKCS#7 files have the file name extensions `.p7b` and `.p7c` and do not contain private keys, only public ones. PKCS#12 files have the file name extensions `.pfx` and `.p12` and may contain private keys as well as public keys.

To use the *verifysigner* option to specify one of these types of file to define a list comprising the owners of the certificates in the file, prefix the file name with a hash (#) character:

```
pkzipc -extract -verifysigner=#my_cert_file.p7b test.zip
```

The *verifysigner* option can be configured for use by default.

5

Working with Digital Signatures

With SecureZIP, you can attach a digital signature to files in an archive, or to an archive itself. A digital signature assures people who receive the signed file that it is really from the person who signed it and has not been changed.

Note: PKZIP authenticates digital signatures on files signed by others, but you must have SecureZIP to attach digital signatures of your own.

SecureZIP allows you to digitally sign either individual files in an archive or the central directory of the archive, or both. The central directory contains a list of files in the archive. Signing the central directory enables a recipient to confirm that the archive as a whole has not changed. Both PKZIP and SecureZIP authenticate digital signatures on extraction.

SecureZIP signing functionality is based on the X.509 certificate standard and is compatible with standard authenticity functionality in other applications such as Microsoft's Internet Explorer. These certificates must be in 1024-bit (minimum) RSA format and must contain a private key.

SecureZIP also supports digital signatures under the OpenPGP (RFC 4880) certificate standard. PKZIP will authenticate certificates validated by PGP key rings on your system.

To use SecureZIP to sign files, you must have a digital certificate. Digital certificates are available from various certificate authorities. Visit the PKWARE Web site for information on obtaining a certificate:

<http://www.pkware.com>

This chapter describes the SecureZIP tools and commands that work with digital certificates under both X.509 and OpenPGP standards.

Public-Key Infrastructure and Digital Certificates

SecureZIP uses digital certificates in two important contexts:

- Confirming and authenticating a person's identity through a digital signature
- Encrypting and decrypting files through the use of recipient lists

To apply or authenticate digital signatures, or to encrypt or decrypt files for recipients, PKZIP needs to access keys in the certificates used. In this section, you'll learn some background and terminology that will help you understand how digital certificates work.

Public-Key Infrastructure (PKI)

Use of digital certificates for encryption and digital signing relies on a combination of supporting elements known as a *public-key infrastructure* (PKI). These elements include software applications such as SecureZIP that work with certificates and keys as well as underlying technologies and services.

The heart of PKI is a mechanism by which two cryptographic keys associated with a piece of data called a certificate are used for encryption/decryption and for digital signing and authentication. The keys look like long character strings but represent very large numbers. One of the keys is private and must be kept secure so that only its owner can use it. The other is a public key that may be freely distributed for anyone to use to encrypt data intended for the owner of the certificate or to authenticate signatures.

How the Keys Are Used

With encryption/decryption, a copy of the public key is used to encrypt data such that only the possessor of the private key can decrypt it. Thus anyone with the public key can encrypt for a recipient, and only the targeted recipient has the key with which to decrypt.

With digital signing and authentication, the owner of the certificate uses the private key to *sign* data, and anyone with access to a copy of the certificate containing the public key can authenticate the signature and be assured that the signed data really proceeds unchanged from the signer.

Authentication has one additional step. As an assurance that the signer is who he says he is—that the certificate with Bob's name on it is not fraudulent—the signer's certificate itself is signed by an issuing certificate authority (CA). The CA in effect vouches that Bob is who he says he is. The CA signature is authenticated using the public key of the CA certificate used. This CA certificate too may be signed, but at some point the *trust chain* stops with a self-signed *root* CA certificate that is simply trusted. The PKI provides for these several layers of end-user public key certificates, intermediate CA certificates, and root certificates, as well as for users' private keys.

X.509

X.509 is an International Telecommunication Union (ITU-T) standard for PKI. X.509 specifies, among other things, standard formats for public-key certificates. A public-key certificate consists of the public portion of an asymmetric cryptographic key (the public key), together with identity information, such as a person's name, all signed by a certificate authority. The CA essentially guarantees that the public key belongs to the named entity.

Digital Certificates

A *digital certificate* is a special message that contains a public key and identity information about the owner, usually including name and perhaps email address. An ordinary, end-user digital certificate is digitally signed by the CA that issued it to warrant that the CA issued the certificate and has received satisfactory documentation that the owner of the certificate is who he says he is. This warrant, from a trusted CA, enables the certificate to be used to support digital signing and authentication, and encryption of data uniquely for the owner of a certificate.

For example, Web servers frequently use digital certificates to authenticate the server to a user and create an encrypted communications session to protect transmitted secret information such as Personal Identification Numbers (PINs) and passphrases.

Similarly, an email message may be digitally signed, enabling the recipient of the message to authenticate its authorship and that it was not altered during transmission.

To use PKI technology in SecureZIP for encryption and to attach digital signatures, you must have a digital certificate.

Certificate Authority (CA)

A *certificate authority* (CA) is a company (usually) that, for a fee, will issue a public-key certificate. The CA signs the certificate to warrant that the CA issued the certificate and has received satisfactory documentation that the owner of the new certificate is who he says he is.

Private Key

A *private key* is used to decrypt data encrypted with the associated public key and to attach digital signatures.

A private key must be accessible solely by the owner of the certificate because it represents that person and provides access to encrypted data intended only for the owner.

SecureZIP may use a private key maintained in X.509 PKCS#12 format. To access such keys, a password must be entered for each SecureZIP request.

Public Key

A *public key* consists of the public portion of an asymmetric cryptographic key in a certificate that also contains identity information, such as the certificate owner's name.

The public key is used to authenticate digital signatures created with the private key and to encrypt files for the owner of the key's certificate.

Certificate Authority and Root Certificates

End entity certificates and their related keys are used for signing and authentication. They are created at the end of the trust hierarchy of certificate authorities. Each certificate is signed by its CA issuer and is identified in the "Issued By" field in the end certificate. In turn, a CA certificate can also be issued by a higher level CA. Such certificates are known as *intermediate* CA certificates. At the top of the issuing chain is a self-signed certificate known as the *root*.

SecureZIP uses public-key certificates in PKCS#7 format. The intermediate CA certificates are maintained independently from the ROOT certificates.

Using Digital Signatures

This section describes less common tasks relating to signing archives and files inside archives. You will also see special tasks for using and handling certificates in Windows and UNIX systems. See "Attaching Digital Signatures" in Chapter 3 and "Authenticating Digital Signatures" in Chapter 4 for information on these tasks.

Attaching a Signature to an Existing Archive

You can use *sign* as a command to sign an existing archive's files as well as its central directory.

Examples:

To digitally sign all files and central directory in save.zip using the "My Name" certificate:

```
pkzipc -certificate="My Name" -sign=all save.zip
```

To digitally sign *.doc in save.zip using the "My Name" certificate

```
pkzipc -certificate="My Name" -sign=files save.zip *.doc
```

To digitally sign the central directory of save.zip using the "My Name" certificate

```
pkzipc -certificate="My Name" -sign=cd save.zip
```

Note: If you intend to perform multiple operations on the archive, always put -sign last.

Applying Strict Checking to Certificates

strict

The *strict* option is for use when doing certificate-based encryption or attaching digital signatures. The option turns on *strict checking*: in other words, it checks to be sure that certificates are

- Valid
- Designated (on the certificate) to be used for the purpose for which they are about to be used in the current command line, namely, encryption or signing

A field on the certificate shows whether the certificate is designated for use only for a specified purpose. Strict checking excludes certificates that are either not designated for any purpose or are designated for the wrong one. For example, strict checking excludes a certificate from being used for encryption if it is designated for signing.

Note: Strict checking only applies to X.509 certificates.

The usage flags listed in this table can optionally be turned off before a strict check is performed:

| <i>Option</i> | <i>Description</i> |
|--------------------|--|
| <i>KeyUsage</i> | Check the purpose for which the certificate is designated (encryption or signing). |
| <i>TimeNesting</i> | Check whether the period of validity of the certificate does not extend past the dates when the issuer certificate is valid. For example, if the issuer certificate is valid from February 1, 2005, to January 31, 2008, the date range during which the selected certificate is supposed to be valid does not begin before February 1, 2005, or end after January 31, 2008. |
| <i>TimeValid</i> | Check whether the current date is within the valid range of dates for the certificate |

The following command line applies strict checking to the certificate to be used to encrypt for a recipient:

```
pkzipc -add -cryptalgorithm -recipient="John Q. Public" -strict
test.zip *.doc
```

If a certificate does not pass strict checking, it is not used, and PKZIP displays a warning like the following:

```
(W6) Warning! John Q. Public does not pass the strict certificate
checks, and will not be used.
```

When a certificate fails strict checking and is not used, other warnings may display as well. For example, if the certificate in the sample command line above fails strict checking, PKZIP also displays the following two warnings because a strong encryption method was specified (*cryptalgorithm*) but no certificate survived strict checking:

```
(W47) Warning! No recipients specified
```

```
(W63) Warning! You must specify -passphrase or -recipient to encrypt
files!
```

Checking for Revoked Certificates

crl

Digital certificates used to apply signatures and to do recipient-based encryption are issued by a certificate authority (CA).

Periodically, CAs publish lists of certificates that have been revoked for one reason or another. For example, an employer might request revocation of a certificate that belongs to an employee who has left the company. Or revocation might be requested for a certificate that has been lost or stolen with its private key.

A CA's list of revoked certificates is called a *certificate revocation list* (CRL). It consists of a file that contains serial numbers of certificates that have been revoked and the dates. The CRL is signed by the issuing CA.

The *crl* option tells PKZIP to check to see if a certificate that you propose to use for digital signing, encryption, or authentication appears in a CRL accessible to PKZIP. If it does, PKZIP displays a warning, (W42) *Certificate was revoked*.

Note: CAs periodically update CRLs. The fact that you can use the *crl* option and not receive a warning only guarantees that the certificate you accessed is not on a CRL that PKZIP checked. The certificate could still have been revoked subsequent to publication of your list.

The following sample command line checks any certificates used for signatures in an archive to be extracted:

```
pkzipc -extract -crl test.zip
```

You can configure the *crl* option so that it is used by default.

The following command line checks the certificate used to encrypt for a recipient:

```
pkzipc -add -recipient="John Q. Public" -crl test.zip *.doc
```

The command line below checks the certificate used to apply John Adams' signature to an archive:

```
pkzipc -add -certificate="John Adams" -crl test.zip
```

To have PKZIP refuse to use a revoked certificate for signing or encrypting, use the *strict* option. Unless you include the *strict* option, PKZIP merely warns if a certificate is revoked and uses it anyway for signing or encrypting.

The following sample command line checks the certificate used to encrypt for a recipient and uses the *strict* option to ensure that the certificate is used only if it is not known to be revoked:

```
pkzipc -add -recipient="John Q. Public" -crl -strict test.zip  
*.doc
```

Obtaining a CRL

Certificate authorities commonly make CRLs available for downloading on their Web sites. A CA is apt to provide different CRLs for different series or types of certificates. You must find the CRL for the type of certificate that you want to use it for.

For PKZIP to access a CRL, the CRL must be downloaded and imported into a certificate store that PKZIP checks for certificates. Such a downloaded and imported CRL is called a *static* CRL to distinguish it from a *dynamic* CRL that may be published on the Web. PKZIP does not access CRLs published on the Web.

In Windows, you can import a CRL by double-clicking the downloaded file.

Using Digital Certificates on Windows

Microsoft Windows sets up certificate stores on the local system, and you can use the Windows Control Panel to work with certificates, private and public keys.

Exporting Public Keys in Windows

If an archive is signed or contains signed files, certificates that have the public keys needed to authenticate the signatures are included in the archive. You can export these public key certificates to install on your system if you do not already have them. (A method that works on most Windows systems is to right-click the exported certificate file in Windows Explorer and choose Install certificate.) Once the certificate is installed, you can use its key with email that you send or receive from the owner. To export public keys for certificates used to sign files in the current archive:

1. Choose Export to open a Save As dialog.
2. Enter a name and location for the file.

Typically, this type of file will have the file name extension .p7c unless you specify a different one.

Note: A .p7c file can contain all the certificates in a certificate chain. Certificates are issued in chains: one certificate may be issued by another certificate further back in the chain. The chain starts with a root certificate issued by a trusted certificate authority.

Backing Up Private Keys in Windows

You can back up a private key to a .pfx file from the Windows Control Panel or Internet Explorer. The following steps describe how to do this in Windows 7 with Internet Explorer version 8. The specific process will differ, depending on your version of Internet Explorer.

1. Go to Start > Control Panel > Internet Options.
2. Select the Content page and click Certificates.
3. Select a certificate and choose the Export button to open the Certificate Export Wizard.
4. In the Export Wizard screen, click Next >.
5. The Export Private Key screen appears. Select "Yes, export the private key."

6. In the Certificate Export File Format screen select "Personal Information Exchange" and check the box "Include all certificates in the certificate path if possible" and check the box "Enable strong protection". Click Next >.
7. Type an export password twice. Click Next >.
8. Use the Browse button to identify the directory where the certificate is to be stored. Name the file. Click Next >.
9. Click Finish to complete the Certificate Manager Export Wizard.

Importing an Exported Certificate

To restore a previously exported certificate to your Windows system:

1. Go to Start > Control Panel > Internet Options.
2. Select the Content page and click Certificates.
3. Click Import... to open the Certificate Import Wizard.
4. In the Export Wizard screen, click Next >.
5. In the File to Import box, click Browse and locate the saved Certificate file. Use the dropdown menu on the right to change X.509 Certificate (*.cer;*.crt) to the .PFX extension, Click the file, click Open, and then click Next >.
6. Type the password in the Password Protection for Private Keys box to access the file, click to select both the Enable strong private key protection and the Mark the private key as exportable boxes, and then click Next >.
7. In the Certificate Store screen, choose whether to Automatically select the Certificate Store based on the type of certificate, or keep all certificates in a specific store (most likely Personal). Click Next .
8. Click Finish to complete the Certificate Import Wizard.
9. In the screen that appears, click Set Security Level.
10. Select "High" to activate password protection for your certificate. Click Next.
11. Enter the new password twice and click Finish.

See the second note in "Notes on Using Certificates in Windows" for more information on setting security levels.

Notes on Using Certificates in Windows

- PKZIP does not work directly with Netscape certificate stores. For PKZIP to access a certificate that you used Netscape to install, you must export the certificate from Netscape and then install it in the Windows certificate stores (usually by double-clicking on the certificate file in Windows Explorer).
- When you install a certificate on your system, the level of security configured can affect what you may see when compressing files with digital certificates. The level of security— medium or high—determines what type of notification you may see when your private key is accessed by an application. Since SecureZIP uses your private key to sign a file, you may receive additional prompts or dialogs when signing a file.

If you selected low security, SecureZIP will be allowed to access your private key as needed with no additional prompts or dialogs. If you use medium security (the default), you will receive an additional notification dialog each time you access the private key. If you use high security, you will be

prompted to enter the passphrase (the one entered when the certificate was installed on your computer) before the certificate can be used.

Advanced Encryption Options in Windows

cryptoptions

This option covers three special cases involving encryption under Windows. One sub-option enables Windows systems equipped with newer Intel processors that include an AES instruction set to take advantage of the increased encryption speed. The other two sub-options enable PKZIP to support certificate-based encryption compatible with most smart cards.

The *FastAES* sub-option tells PKZIP to use the fastest version of the Advanced Encryption Standard (AES) available on the system. This option is only available if FIPS Mode is disabled, as this option is not FIPS-compatible. See “Encrypting Using Only FIPS-Approved Algorithms” for more information on FIPS. By default, this sub-option is turned off.

On Windows, PKZIP can access certificates stored on smart cards to decrypt strongly encrypted files if the smart cards work with Windows’ facilities for managing digital certificates.

These two *cryptoptions* sub-options are both on by default. They can be turned off to provide compatible certificate-based encryption for two special cases:

- *smartcard* sub-option: Turn off to support certificate-based encryption for recipients using versions of PKZIP v6.0 or earlier.
- *win2000* sub-option: Turn off to provide pure AES certificate-based encryption

The *smartcard* sub-option enables smart cards to decrypt files encrypted for a recipient list. However, if the *smartcard* sub-option is set, versions of PKZIP prior to 6.1 cannot decrypt files encrypted for a recipient list. To enable users of these earlier versions of PKZIP to decrypt such files, turn off the *smartcard* sub-option. Note, though, that files encrypted with this sub-option off cannot be decrypted by smart cards.

The sub-option affects only recipient-list encryption (that is, encryption using the *recipient* option). All versions of PKZIP can decrypt passphrase-encrypted files regardless of how the *smartcard* sub-option is set.

The *win2000* sub-option enables recipients using smart cards or running on Windows NT or Windows 2000 to extract files encrypted with AES for a recipient list.

By default when using a certificate to encrypt data with AES, PKZIP uses 3DES to protect the key. This is necessary to enable recipients using smart cards or running on Windows NT or Windows 2000 to decrypt the files.

Turn off the *win2000* sub-option if you want to avoid any use of the 3DES encryption algorithm when doing AES encryption. Turning off the option causes PKZIP to use only AES but has the result that recipients using smart cards or running on Windows NT or Windows 2000 will likely be unable to extract files encrypted for a recipient list with AES.

Like the *smartcard* sub-option, the *win2000* sub-option affects only recipient-list encryption (that is, encryption using the *recipient* option). Users of Windows NT or Windows 2000 can decrypt files encrypted using AES with a passphrase even with the sub-option off. (Smart cards do not support passphrase-based encryption.)

All three sub-options are set independently of one another. Turning *smartcard* off does not affect *win2000*. Nor does turning on *FastAES* affect the other sub-options.

For example, the configuration display of initial defaults shows both of these sub-options turned on (see “Viewing Configuration Settings” in Chapter 7):

```
CryptOptions = Smartcard, Win2000
```

To configure one of the sub-options off, prefix it with a hyphen:

```
pkzipc -config -cryptooptions=-smartcard
```

or, to configure both off:

```
pkzipc -config -cryptooptions=-smartcard,-win2000
```

Either option can also be turned off just for the current command line, to override a configured default setting:

```
pkzipc -add -cryptooptions=-smartcard -recipient="John Q. Public"
test.zip
```

To turn one of the sub-options on, omit the hyphen prefix. For example, the following command line configures all sub-options on:

```
pkzipc -config -cryptooptions=smartcard,win2000,fastaes
```

Working with OpenPGP Files

Some organizations use encryption tools based on the OpenPGP standard, rather than X.509. OpenPGP uses the same basic Public Key Infrastructure principles for exchanging encrypted files, but uses a decentralized “Web of Trust” method of authenticating signatures.

PKZIP and SecureZIP extract and decrypt files that comply with the OpenPGP standard, RFC 4880. SecureZIP can also create OpenPGP-compliant files and sign files with OpenPGP certificates. In this section, you’ll learn more about the OpenPGP standard, and how to use PKZIP with OpenPGP.

Overview: OpenPGP vs. X.509

As described in “Public-Key Infrastructure and Digital Certificates” earlier in this chapter, the X.509 standard relies on a hierarchical “trust chain” model, where an individual digital signature is issued by an intermediate Certificate Authority (CA), which is assumed to have received enough documentation to determine that an individual is who he says he is. The intermediate CA’s certificate gets its certificate, in turn, from a Root CA. Each certificate says who issued it, and theoretically if you question the authenticity of a certificate, you can find the documentation presented to the original CA.

OpenPGP certificates are typically created by individuals, and authenticated by other individuals. In the real world, you have friends who can vouch that you are who you say you are. If you walk into a room full of strangers, your friend can introduce you to the people he knows. Since you trust that your friend is correctly identifying his friends and acquaintances, that trust extends to his friends too.

When you translate the above experience to the electronic, OpenPGP world, it works this way: You create an OpenPGP certificate to identify yourself. When a friend comes to visit, display the certificate. The friend can now sign your certificate (often called “key signing”) and certify that this certificate represents you. Now everyone who trusts the person who signed your key can also trust that your certificate is

authentic. A Web of Trust is developed as more people authenticate each certificate. Everyone in the Web of Trust can also exchange messages in the OpenPGP format.

Supported OpenPGP Algorithms

This table lists the supported OpenPGP algorithms used for encryption, signing, and hashing.

| <i>Algorithm</i> | <i>Type</i> |
|-----------------------|------------------------------------|
| RSA | Public-Key Signature or Encryption |
| Elgamal | Public-Key Encryption |
| DSA | Public-Key Signature |
| 3DES | Symmetric-Key |
| CAST5 | Symmetric-Key |
| AES (128-bit) | Symmetric-Key |
| AES (192-bit) | Symmetric-Key |
| AES (256-bit) | Symmetric-Key |
| Uncompressed | Data Compression |
| ZIP (RFC 1951) | Data Compression |
| BZIP2 | Data Compression |
| SHA-1 | Hash |
| SHA-256 | Hash |
| SHA-384 | Hash |
| SHA-512 | Hash |
| MD5 | Hash |

Setting Up OpenPGP Keyrings

To configure SecureZIP to identify your OpenPGP private and public key pairs for inclusion in a recipient list, type:

```
pkzipc -archivetype=pgp -config -recipient=enabled
```

SecureZIP will always search for public keyring(s) in the system's environment variables, starting with PK_OPENPGP_PUBLIC_RING. If this environment variable is not set in Windows, SecureZIP will then search these folders until it locates the keyring:

- <Documents>\PGP\
- <AppData>\gnupg\

To identify existing OpenPGP keys for use in creating and signing OpenPGP files, type:

```
pkzipc -archivetype=pgp -config -certificate=enabled
```

SecureZIP will search the same folders listed above for secret keyring(s).

Configuring Other OpenPGP Settings

You can configure both the default hash and encryption algorithm for OpenPGP files separately from the X.509 algorithms. To do this, you must always include the *archivetype=PGP* option. For example, to use SHA-256 as the default hash algorithm for OpenPGP files, use this command:

```
pkzipc -archivetype=pgp -config -hash=sha256
```

6

Miscellaneous Operations

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. Available choices are represented by the sub-options described in the following table.

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

If you use *add* or *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% | 06/01/2001 | 4:50a | 87b3c388 | -a-w | red.txt |
| 8369B | DeflatN | 3084B | 63.2% | 06/01/2001 | 4:50a | 87b3c388 | -a-w | tan.txt |

16KB 6168B 63.2% 2PKZIP also provides two 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.

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% | 06/01/2001 | 4:50a | red.txt |
| 8369B | DeflatN | 3084B | 63.2% | 06/01/2001 | 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, 2001  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, 2001  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
```

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.

| <i>Sub-Option</i> | <i>Description</i> |
|-------------------|---|
| <i>none</i> | Does not change line endings |
| <i>dos</i> | DOS/Windows (carriage return/newline) |
| <i>mac</i> | MacOS (carriage return) |
| <i>unix</i> | UNIX (newline) |
| <i>ebcdic,nl</i> | With ZDW files, substitute EBCDIC newline (0x15) |
| <i>ebcdic,lf</i> | With ZDW files, substitute EBCDIC linefeed (0x25) |

| <i>Sub-Option</i> | <i>Description</i> |
|---------------------|---|
| <i>ebcdic,crlf</i> | With ZDW files, substitute EBCDIC carriage return/linefeed (0x0D25) |
| <i>ebcdic,lfcrl</i> | With ZDW files, substitute EBCDIC linefeed/carriage return (0x250D) |
| <i>ebcdic,crnl</i> | 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" in Chapter 3 for information on using an independent Time Stamp Authority to securely establish when a file was created or modified. This feature is available only with SecureZIP.

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.)

| <i>Token</i> | <i>Replaced by</i> |
|---------------|---|
| { <i>id</i> } | A job ID specified separately with the <i>jobid</i> 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, 2006 12:06:29 a.m., the resulting archive will be named `Design Spec 2006-08-09-12-06-29am.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="C:\newzips {yyyy}-{mm}-{dd}-{h}-{MM}-{SS}{ampm}" C:\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 2006-08-09-12-06-29am.zip` if the date and time are August 09, 2006 12:06:29 a.m.

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

Printing the Contents of a ZIP File

print

PKZIP gives you the option of printing files contained in a .ZIP file to a selected printer. For example, if you wish to print all of the .txt files contained in a .ZIP file, type the following:

```
pkzipc -print=lpt1 test.zip *.txt
```

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

```
Extracting files from .ZIP: test.zip
    Inflating: readme.txt <to LPT1>
    Inflating: whatsnew.txt <to LPT1>
```

In this example, all files with a .txt extension that exist in the test.zip are printed to the LPT1 printer. If you do not specify a print device, the 'default' printer is used. Since many .ZIP files contain an information document (e.g., readme.txt), the *print* option is a good way to determine the contents of a .ZIP file without requiring you to extract a file or file(s) to your hard drive.

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.

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 particular 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 Appendix B 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 particular 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 Appendix B 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 actually 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=z:/public big_file.zip myfile.doc
```

Notes:

- You need to provide a path in addition to the drive letter only if you have a particular 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, each file that is added, you will be prompted for the file to be set to ASCII, BINARY or if you want PKZIP to 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
```

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
```

7

Changing Defaults for Commands and Options

You can use the *configuration* command to view current default settings for commands and options. You also use this command to change default values. Another command—*default*—restores default settings for all commands and options to their original values.

With the *altconfig* option, you can create and apply alternate configuration profiles for special purposes.

Viewing Configuration Settings

To use the *configuration* command to view current default values for all commands and options, enter the command by itself on the command line:

```
pkzipc -configuration
```

A list of current default settings displays:

```

204 = Disabled
ArchiveDate = None
Comment = None
CRL = Disabled
Extract = Extract All Files
Hash = SHA-1
KeyPassphrase = Disabled
ListChar = @
Lowercase = Disabled
MoveArchive = Disabled
NoExtended = Disabled
OpenFile = Never (skip)
Passphrase = Disabled
Shortname = None
Sort = None
Substitution = Disabled
Times = All
View = Normal

Add = Add All Files
CD = Normal
Comp Method = Deflate
Encode = Disabled, UUE
FIPSMODE = Disabled
Hash (OpenPGP) = SHA-1
Level = Normal
Local e = Enabled
More = Disabled
NoArchiveExtension = Disabled
NoFix = Disabled
OptionChar = -
Recurse = Disabled
Shred = None
Span = None, Auto-Detect
Test = All Files
UTF8 = Disabled
ZoneIdentifier = Disabled

ASCII = Disabled
AVArgs = Disabled
AVScan = Disabled
Binary = Disabled
Certificate = John Public
Certificate (OpenPGP) = Disabled
CryptAlgorithm = Traditional
CryptAlgorithm (OpenPGP) = AES (256-bit)
CryptOptions = Smartcard, Win2000, FastAES
Embedded = Disabled
Error = None

```

```

Header = Disabled
Recipient = Disabled
Recipient (OpenPGP) = Disabled
Sign = Disabled, Central directory and individual files
Silent = Copying
Strict = Disabled : KeyUsage, TimeValid, TimeNesting
TS = Disabled
Temp = Disabled
VerifySigner = Disabled
Warning = None

PKSFX Options
  Create Folders = Disabled    Display Messages = Disabled
  Overwrite = Prompt           Sfx = WIN32_X86_C1230
  SfxLogfile = Disabled       Type = EasySFX
  Destination =
  Title Bar = Disabled
  RunAfter = Disabled
  Program Group = Disabled
  Extensions = Disabled

Compression Options
  After = Disabled
  Attributes = Read-Only, Archive
  Before = Disabled
  Exclude = Disabled
  Include = Disabled
  Larger = Disabled, 0
  Mask = None
  Newer = Disabled
  Older = Disabled
  Overwrite = Always Overwrite
  Path = No Path Information
  Smaller = Disabled, 18, 446, 744, 073, 709, 551, 615
  Translate = None - No Conversion

Extraction Options
  After = Disabled
  Attributes = Read-Only, Hidden, System, Archive
  Before = Disabled
  Exclude = Disabled
  Include = Disabled
  Larger = Disabled, 0
  Mask = None
  Newer = Disabled
  Older = Disabled
  Overwrite = Prompt
  Path = Full Path
  Smaller = Disabled, 18, 446, 744, 073, 709, 551, 615
  Translate = None - No Conversion

```

In the display, the command/option is to the left of the equal sign, and the default setting is to the right. An option listed as *Disabled* is disabled by default. An option listed as *None* has a *None* sub-option that is its default value. A command or option that has any other value has that value as its default. In most of these cases, the value is a predefined sub-option.

The PKSFX options appear only if you have PKZIP Enterprise or SecureZIP.

How Default Settings Work

Configurable *options* that have a default value are applied, with their default value, even when they are not explicitly entered on the command line. This includes options that have a default value of *None*: these options are applied with the value of their *None* sub-option. Disabled options are not applied.

For example, `Comment = None` indicates that, by default, PKZIP does not prompt for comments to attach to files in an archive. If you want PKZIP to always prompt for comments on files, you can configure the default to a different value—for example, `Comment = All`. **All** is another sub-option of *comment*. With this default, PKZIP will routinely prompt even when the *comment* option is not used in the command line.

For a *command*, the default setting determines what the command does when the command is listed on the command line *without an explicit sub-option*.

A command must explicitly appear in the command line to be used. This is a difference between commands and options. A default value for a command determines what the command does when it is used by itself, without any specified sub-option.

For example, the *add* command can add all specified files to an archive (the *all* sub-option), or it can just add ones that are not in the archive already or are newer versions of files that are (the *update* sub-option). Initially, *add* has the default value of *all*, so a command line like the following adds all specified files indiscriminately:

```
pkzipc -add myfiles.zip *.*
```

To have this same command line add only new and newer files instead, you can use the *configuration* command to change the default behavior of *add* from *all* to *update* (see the section “Changing a Default Value,” below):

```
pkzipc -configuration -add=update
```

Some options also have a value—distinct from any configurable default value—that is used automatically if the option is used on the command line without an explicit sub-option. This value overrides any configured default value.

For example, the initial configurable default for the compression filter *path* option, which saves or restores path information, is *None*. The option has several other sub-options that can be set as the default value instead, but no matter which sub-option is the default, *path* has the value of *current* (one of the sub-options) when used without a sub-option, as in this command line:

```
pkzipc -add -path myarchive.zip *.txt *.doc
```

Appendix A lists the defaults and override values for all commands and options.

Filter Options

At the bottom of the listing of defaults are two sets of *filter options*, one for compression and one for extraction. These are called filter options because they filter out files that do not meet their criteria. Only files that are not filtered out are selected. For example, the *after* option filters out all files whose date falls before the date specified with the option.

Each of the filter options takes a different default value for compression and for extraction.

Changing a Default Value

To change a default setting in the configuration file, use the *configuration* command. You can abbreviate this command to: *config*.

To specify a value (sub-option) to use as the default value for a command/option:

- Type *pkzipc -config* and the name of the command/option followed by an equal sign and the sub-option value you want to set as the default.

For example, to change the default for the *add* command to *update* (instead of the original default, *all*), type the following:

```
pkzipc -config -add=update
```

To turn on and use by default an option that has either no sub-options or a sub-option that is used by default:

- Type *pkzipc -config* and the name of the option.

For example, to do virus scanning by default when extracting files, set the *avscan* option on by default:

```
pkzipc -config -avscan
```

To turn on the *silent* option and use its default sub-option:

```
pkzipc -config -silent
```

After you use the *configuration* command to change a default setting, an updated list of settings displays. You can suppress this list so that it is not displayed. To do so, use the *configuration* command with its *silent* sub-option.

For example, the following command line sets a default value for the *overwrite* option and suppresses display of the updated list of settings that the *configuration* command ordinarily prints to the screen:

```
pkzipc -config=silent -overwrite=never
```

Note that the *silent* sub-option of the *configuration* command is different from the *silent* option proper, which suppresses messages when adding or extracting.

See Appendix A for a list of PKZIP commands, options and sub-options, and information about which commands and options have configurable defaults.

Changing Defaults for Filter Options

Options listed as filter options in the display of default settings take separate defaults for compression and extraction. To specify a default for a filter option for one of these operations, include the related command (*add* or *extract*) on the command line.

For example:

```
pkzipc -config -add -newer=1d
```

If you specify a default for a filter option without including the related command, as in the following example, PKZIP asks whether you want to specify the default for compression, extraction, or both:

```
pkzipc -config -newer=1d
```

Changing Defaults for Compression Method

The *Comp Method* item in the screen of configuration settings shows the current default setting for compression method. To set a default compression method,

specify the compression method that you want to make the default. For example, the following command makes BZIP2 the default compression method:

```
pkzipc -config -bzip2
```

The options in the table below set compression method:

| <i>Compression Method Options</i> | <i>Description</i> |
|-----------------------------------|--|
| <i>deflate64</i> | Sets the compression method to Deflate64 |
| <i>bzip2</i> | Sets the compression method to BZIP2 |
| <i>dclimplode</i> | Sets the compression method to DCL Implode |
| <i>lzma</i> | Sets the compression method to LZMA |
| <i>ppmd</i> | Sets the compression method to PPMd |
| <i>store</i> | Sets the compression method to Store (that is, no compression) |

The options in the next table set both compression method and level:

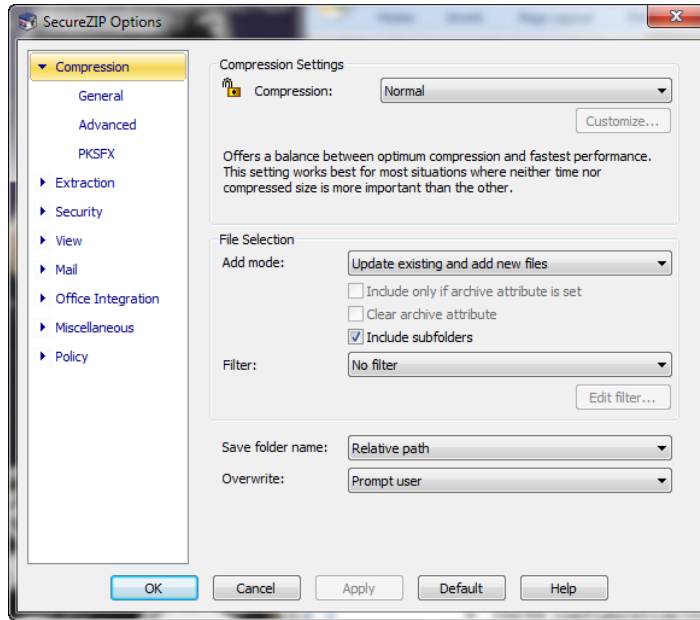
| <i>Option</i> | <i>Description</i> |
|----------------|--|
| <i>speed</i> | Sets the compression method to Deflate—the initial PKZIP default method—and the level of compression to 1 (the lowest) |
| <i>fast</i> | Sets the compression method to Deflate and the level of compression to 2 |
| <i>normal</i> | Sets the compression method to Deflate and the level of compression to 5. Normal is the initial default setting for compression method and level for PKZIP. |
| <i>maximum</i> | Sets the compression method to Deflate and the level of compression to 9 |
| <i>level=0</i> | When set to 0, the level option sets the compression method to Store (no compression) |

For example, the following command sets the default compression method to Deflate and the default compression level to 9:

```
pkzipc -config -maximum
```

Using the Options Dialog to Change Defaults

As an alternative to using the command line to change defaults, you can use the graphical Options dialogs if you have PKZIP or *SecureZIP* for Windows Desktop installed:



To display the graphical Options dialog:

- Use the *configuration* command with the *gui* sub-option:

```
pkzipc -config=gui
```

In the dialog, the *Help* button opens the online help for the Windows version of PKZIP or SecureZIP. There you can read how to set options in the dialog.

Settings that you make in the Options dialog when you use the *gui* sub-option apply only to the command line version of the product, not to the Windows version. Similarly, if you open the Options dialog from the Windows version, options that you set in the dialog apply only to the Windows version.

If you use the *gui* sub-option without having PKZIP for Windows installed, the sub-option is ignored, and the command works as if you had entered it with no sub-option.

Resetting to Original Defaults

Command or option default values that you have changed can be reset back to their original values. You can reset changed defaults either for individual commands and options that you specify, or wholesale, for all.

Resetting Individual Defaults

To reset an individual command or option to its original default value in the configuration file, use the *config* command and put two hyphens in front of the command or option that you want to reset.

For example, to reset the *add* value back to its original default without resetting any other default values that you may have modified, type the following and press ENTER:

```
pkzipc -config --add
```

Notice that there are *two* hyphens in front of the *add* command. The command changes the *update* value we set in a previous example back to *all*.

You can also use two hyphens without the *config* command to reset, or turn off, a configured default for an option just for the current command line. The example below turns off a configured default value (for instance, *AES,256*) for the *cryptalgorithm* option to do traditional encryption instead just for the current command line:

```
pkzipc -add --cryptalgorithm -passphrase wedding_plans.zip *.txt
```

Resetting All Defaults

To reset default values for all commands and options, use the *default* command. Type the following and press ENTER:

```
pkzipc -default
```

Using an Alternate Configuration File

altconfig

You can create alternate configuration profiles to use for special purposes. The *altconfig* option creates and loads alternate configuration profiles. With an alternate configuration profile, you can temporarily change multiple default command or option settings in a single pass just by loading the configuration profile that defines them.

Creating an Alternate Configuration File

To create an alternate configuration profile, use the *altconfig* option with the *configuration* command. This creates a copy of the current main configuration file with the file name and at the location specified by the *altconfig* option and updates default settings in the copy with any new settings specified in the command line. If an alternate configuration file of that name already exists at the specified location, the file is updated with the new default settings from the command line. Other settings in the file are left unchanged.

For example, the command line below creates or updates an alternate configuration file *secure.xml* in the root directory of drive C and specifies default values for the *cryptalgorithm*, *sign*, and *certificate* options:

```
pkzipc -config -altconfig=c:\secure.xml -cryptalg=aes,256 -
sign=all
-cert="John Public"
```

If you have PKZIP for Windows Desktop installed, you can use *config=gui* to configure defaults in the graphical Options dialogs. For example, the following command line opens the Options dialogs:

```
pkzipc -config=gui -altconfig=c:\secure.xml
```

If *secure.xml* exists, PKZIP displays its settings in the graphical Options dialogs. If the file does not already exist, PKZIP displays the settings of your main configuration file. In either case, saving settings from the Options dialog saves to *secure.xml*.

Using an Alternate Configuration File

To use the settings in an alternate configuration file, use the *altconfig* option to specify the file in a command line with which you want to use the alternate settings.

You can use the *altconfig* option with any command. For example, the following command line loads the alternate configuration file `secure.xml` to use its settings with the *add* command. The settings cause PKZIP to use the specified certificate to sign the archive central directory and all files added to `foo.zip` and to encrypt the files using the strong encryption algorithm AES 256.

```
pkzipc -add -altconfig=c:\secure.xml -pass foo.zip *.doc
```

Loading the settings from the alternate configuration file saves the trouble of specifying them all on the command line and does not require changing the main configuration file.

To view settings in an alternate configuration file, use the *configuration* command and specify the file with *altconfig*:

```
pkzipc -config -altconfig=c:\secure.xml
```

An alternate configuration file must already exist for you to use it in a command line with the *add* command or any other command besides *configuration*. The only time you can use the *altconfig* option to specify an alternate configuration file that does not already exist is when you use the option with the *configuration* command to create an alternate configuration file.

8

Command Characteristics

This chapter describes changes you can make to the PKZIP infrastructure. For example, you can specify different characters to use for the list character and the option character, and you can cause PKZIP to display dates and times using a different format from the one used by default on your system.

Ordinarily, the original values for the settings described in this chapter should be satisfactory. You should not change them without a good reason.

Changing Date and Time Environment Variables

locale

The *locale* option causes PKZIP to use your system's format for displaying dates and times. The option has two sub-options, *enable* and *disable*, to set it on or off. The option is configurable and is set on by default.

Formerly PKZIP used a date format of MMDDYY and a 12-hour time format of HH:MM. If you prefer PKZIP to use this format, you can revert to it by setting *locale* to *disable*.

If you have disabled the *locale* option by default, you can enable it for a particular command line by setting the option to *enable* in the command line. For example:

```
pkzipc -add -locale=enable test.zip *.doc
```

This command line causes PKZIP to use the system-defined settings regardless of the default settings.

Changing the List Character for List Files

listchar

PKZIP allows you to specify an ASCII file as a source list of the files to be archived. By default, you specify this ASCII file by pointing to it with the "@" character in your command line. However, if you have files that begin with an "@", you may experience problems when trying to add these files to a .ZIP archive. Fortunately, PKZIP allows you to change the default list character to avoid such problems. This is accomplished using the *listchar* option. For example, if you wish to define the "+" character in place of the "@" as your default list character, type the following and press ENTER:

```
pkzipc -config -listchar=+
```

If you wish to specify an alternate list character on the command line itself, could type a command line similar to the following and press ENTER:

```
pkzipc -add -listchar=+ test.zip +file1.txt
```

When used as a command line option, the *listchar* option only applies to the options that follow it on that particular command line. In our example the *listchar* option allows you to add files that begin with a "+" character (e.g., +file1.txt). For more information on using list files with PKZIP see "Compressing Files with a List File" in Chapter 0 and "Extracting Files with a List File" in Chapter 4.

Note: Avoid using metacharacters as list characters. Metacharacters have a special significance to the shell and as such their usage may cause unpredictable results. This would include the following characters:

```
; , & ( ) | < > # NEWLINE SPACE TAB
```

Changing the Command/Option Character

optionchar

The *optionchar* option specifies the character to use to identify commands and options as such in command lines. By default, PKZIP uses the hyphen "-" to flag commands and options in a command line. You can use *optionchar* to change this option character to a different character instead. For example, to make it easier to zip files whose names begin with a "-", you might change the option character to a "+".

You can change the option character either just for a single command line or indefinitely, to define a new default character. The following command changes the option character just for the immediate command:

```
pkzipc -optionchar=+ +add save.zip *.doc
```

In a Windows command line, you can also always use the "/" character to indicate a command or option in a particular command line.

```
pkzipc /add save.zip *.doc
```

You can also use *optionchar* with the *configuration* command to define a different option character to use by default. For example:

```
pkzipc +config -optionchar=+
```

Note that the newly defined option character is used immediately, in the same command line in which it is defined, by every command or option other than *optionchar* itself.

Note: Avoid using metacharacters as option characters. Metacharacters have a special significance to the shell and as such their usage may cause unpredictable results. This would include the following characters:

```
; , & ( ) | < > # NEWLINE SPACE TAB
```

A

Reference to Commands and Options

This appendix contains reference information on every PKZIP command and option. For each command/option, the following information is provided:

| <i>Column</i> | <i>Purpose</i> |
|-------------------------|--|
| <i>Name/Description</i> | Gives the name of the command/option and a brief description of what it does. If a default value can be configured for the command/option, the word "Configurable" appears. |
| <i>Value(s)</i> | Lists any sub-options or values associated with the command/option and specifies any initial default values |
| <i>Example usage</i> | Shows examples of the command/option used in a PKZIP command line |
| <i>Used with</i> | Identifies the item as a command or an option: a listing of <i>standalone</i> in this column means that the item is a command. For options, the column lists commands that the option can be used with. If <i>standalone</i> is included with a list of commands, the item can be used as an option with any of the listed commands or can be used by itself as a command. |

Information on each command/option follows:

| <i>Name/Description</i> | <i>Value(s)</i> | <i>Example usage</i> | <i>Used with</i> |
|---|---|--|------------------|
| 204 Turns on PKZIP for DOS 204g compatibility Configurable | No sub-options. ----- No default value. | <code>pkzipc -add -204 save.zip *</code> | add |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|--|
| <p>add</p> <p>Add files to an archive</p> <p>Configurable</p> | <p>all - Compress and add files that are new to the archive as well as files that the archive already contains a (maybe newer) copy of</p> <p>archive - Turn off archive attribute of all added files (prepares backup file set for incremental archiving).</p> <p>freshen - Add only files that the archive already contains an older copy of</p> <p>update - Freshen files that are in the archive already and add any new ones</p> <p>incremental - Add only files that have the archive attribute on, and then turn off the archive attribute</p> <p>-incremental - Add only files that have the archive attribute on, and do not turn off the archive attribute afterward</p> <p>-----</p> <p>Default = all</p> | <p><i>pkzipc -add save.zip *.doc</i></p> <p><i>pkzipc -add=freshen save.zip *.doc</i></p> <p><i>pkzipc -add=incremental save.zip *.doc</i></p> <p><i>pkzipc -add=-incremental save.zip *.doc</i></p> <p>Outputs the archive to STDOUT instead of to a file:</p> <p><i>pkzipc -add -noarchiveextension -silent=normal - *.txt</i></p> | <p>standalone</p> |
| <p>after</p> <p>Process files that have the specified date or a later one</p> <p>Configurable separately for add and extract operations.</p> | <p>Any date in format specified in Country-Settings or the <i>locale</i> option.</p> <p>For example, the US date format is:</p> <p> mmdyy or mmdyyyy</p> <p>-----</p> <p>No default value.</p> | <p>For compression:</p> <p><i>pkzipc -add -after=09152003 save.zip *.doc</i></p> <p>For extraction:</p> <p><i>pkzipc -ext -after=09152003 save.zip *.doc</i></p> | <p>add, extract, delete, test, view, delete, console</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|--|
| <p>altconfig</p> <p>Creates or updates an alternate configuration file containing alternate, specified defaults when used with the <i>configuration</i> command; loads the specified alternate configuration file when used in a command line with any command other than <i>configuration</i>.</p> | <p>Path and name of alternate configuration file to create, update, or load</p> | <p>Create or update an alternate configuration file <i>secure.xml</i> with specified defaults. File is created if it does not exist already, or updated if it does:</p> <pre>pkzipc -config -altconfig=c:\secure.xml -cryptalgorithm=aes,256 -sign=all -certificate="John Public"</pre> <p>Use the default settings specified in alternate configuration file <i>secure.xml</i> when adding files to archive <i>foo.zip</i>:</p> <pre>pkzipc -add -altconfig=c:\secure.xml -passphrase foo.zip *.doc</pre> | <p>All commands except <i>list-certificates</i>, <i>listcryptalgorithms</i>, <i>listsfxtypes</i>, <i>license</i>, and <i>version</i></p> |
| <p>archivedate</p> <p>Sets the modification date of the archive file.</p> <p>Configurable</p> <p>Note: The <i>archivedate</i> option is the same as the older <i>zipdate</i> option, which is now deprecated.</p> | <p>newest - Sets date of the archive to the date of the newest file in the archive</p> <p>oldest - Sets date of the archive to the date of the oldest file in the archive</p> <p>retain - When updating, keeps the date the archive had before it was updated. When creating a new archive, behaves the same as none</p> <p>none - Sets the date of the archive to the date of its latest modification</p> <p>-----</p> <p>Default = none</p> | <pre>pkzipc -add=update -archivedate=retain save.zip *.txt</pre> | <p>add, delete, fix, header, comment, sfx</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|--------------------------|
| <p>archiveeach</p> <p>Creates a separate archive for each of multiple files specified in a single command line.</p> <p>Can be used with <i>archivetype</i> and <i>encode</i> to create .tar.gz archives.</p> | <p><destination> - Directory in which to create the archives</p> <p>By default, archives are created in the current directory.</p> | <p>Creates a separate ZIP archive for each file in the current directory:</p> <pre>pkzipc -add -archiveeach *.*</pre> <p>Creates the archives in a specified destination:</p> <pre>pkzipc -add -archiveeach=C:\newzips *.*</pre> <p>Creates .tar.gz archives:</p> <pre>pkzipc -add -archiveeach -archivetype=tar -encode=gz C:\data*.*</pre> | add |
| <p>archivetype</p> <p>Explicitly specifies the type of archive to be acted on by the command. PKZIP works with ZIP archives by default or infers the archive type from the archive name. Use the archivetype option if PKZIP would assume or infer the wrong type.</p> <p>Always use the archivetype option to specify OpenPGP files.</p> | <p>bzip2 - Specifies the Bzip2 archive type.*</p> <p>zip - Specifies the .ZIP archive type. (default)</p> <p>gzip - Specifies the GZIP archive type.*</p> <p>pgp - Specifies the OpenPGP archive type.</p> <p>tar - Specifies the TAR archive type.</p> <p>uue - Specifies the UUENCODED archive type.*</p> <p>xxe - Specifies an XXENCODED archive type.*</p> <p>* These archive types can contain only one file. To use with multiple files, create an archive of one of the other archive types and use the <i>encode</i> option to encode this archive as the single-file archive type that you want.</p> | <pre>pkzipc -add -archivetype=tar myfile.foo</pre> <p>Creates a TAR archive named myfile.foo.tar</p> <pre>pkzipc -extract -archivetype=bzip2 -noarchiveextension -silent=input -</pre> <p>Extracts from a BZIP2 archive on STDIN</p> <pre>pkzipc -add -archivetype=pgp -cryptalg=AES,128 -recipient="Test" -cert="Test" myfile.pgp *.txt</pre> <p>Creates a PGP archive called myfile.pgp</p> | add, extract, test, view |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|---------------------|
| <p>ascii</p> <p>Set the internal attribute bit (ASCII/Binary) to ASCII.</p> <p>Configurable</p> | <p>The file(s) or file pattern whose internal attribute bit you wish to set to ASCII; if no files are specified, PKZIP prompts for each file.</p> <p>-----</p> <p>No default value.</p> | <pre>pkzipc -add - ascii="*.txt" save.zip * pkzipc -add -ascii save.zip *</pre> | <p>add</p> |
| <p>attributes</p> <p>Stores files with the specified file attribute information in the archive file.</p> <p>Configurable separately for add and extract operations.</p> | <p>hidden - select hidden files.</p> <p>system - select system files.</p> <p>readonly - select read-only files.</p> <p>archive - select files with the archive bit set.</p> <p>all - select all types of files.</p> <p>none - do not select files that have hidden, system, or read-only attributes; overrides the default attributes setting in configuration file.</p> <p><hex value> -The hex value of an attribute to be selected, or the logical OR of multiple hex values</p> <p>-----</p> <p>Default = readonly, archive</p> | <pre>pkzipc -add - attributes=system,hidden save.zip *</pre> | <p>add, extract</p> |
| <p>avargs</p> <p>Specifies any command line arguments to use when running the anti-virus program given in avscan</p> <p>Configurable</p> | <p><command line></p> <p>- A command line that runs an anti-virus program</p> | <pre>pkzipc -extract -avscan= f-prot.exe -avargs="%e /silent /nomem /noboot" myfiles.zip</pre> | <p>extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|---|
| <p>avscan</p> <p>Turns on virus scanning: runs the specified anti-virus program using the anti-virus command line arguments in avargs</p> <p>Configurable</p> | <p><executable> - The name of the anti-virus program executable—with path, if necessary</p> | <pre>pkzipc -extract -avscan= f-prot.exe -avargs="%e /silent /nomem /nboot" myfiles.zip</pre> | <p>extract</p> |
| <p>before</p> <p>Process files that are older than a specified date.</p> <p>Configurable separately for add and extract operations.</p> | <p>Any date in format specified in Country-Settings or the locale option.</p> <p>For example, the US date format is one of the following:</p> <p>mmddyy mmddyyyy ----- No default value</p> | <p>For compression:</p> <pre>pkzipc -add -before=09152003 save.zip *.doc</pre> <p>For extraction:</p> <pre>pkzipc -extract -bef=09152003 save.zip *.doc</pre> | <p>add, extract, delete, test, view, print, console</p> |
| <p>binary</p> <p>Treats the files to be added as binary files: sets the internal ASCII/Binary attribute bit of the files to binary.</p> <p>Configurable</p> | <p>The file(s) or file pattern whose internal attribute bit you wish to set to binary; if no files are specified, PKZIP will prompt for each file.</p> | <pre>pkzipc -add - binary="*.exe" save.zip *</pre> <pre>pkzipc -add -binary save.zip</pre> | <p>add</p> |
| <p>bzip2</p> <p>Compress files using the BZIP2 method.</p> <p>Note: Files compressed with this method can be extracted with most varieties of PKZIP version 4.6 and later. Other .ZIP programs may not be able to extract files compressed with BZIP2.</p> | <p>No sub-options</p> <p>Default compression level: 5</p> | <p>To compress files using the bzip2 algorithm and level 9 compression:</p> <pre>pkzipc -add -bzip2 - level=9 save.zip doc1.txt</pre> <p>To compress files using the default compression level (level 5):</p> <pre>pkzipc -add -bzip2 save.zip *.doc</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|-------------------------------------|
| <p>cd</p> <p>Encrypt file names and other metadata in a ZIP archive's central directory.</p> <p>Requires that <i>passphrase</i> and or <i>recipient</i> options also be used. Uses strong encryption; does not work with traditional ZIP encryption.</p> <p>Encrypting file names produces an archive that requires PKZIP or SecureZIP version 8.0 or later to open it.</p> <p>Configurable</p> | <p>encrypt - Encrypt file names and the archive's central directory</p> <p>normal - Do not encrypt file names; produces a normal ZIP file. Use to override a configured default setting that would otherwise encrypt file names.</p> <p>-----</p> <p>Default = encrypt</p> | <pre>pkzipc -add - recipient="John Q. Public" -cd test.zip pkzipc -add - recipient="John Q. Public" -cd=normal test.zip pkzipc -add -passphrase=mysecret -cryptalgorithm=aes,256 -cd test.zip</pre> | <p>add</p> |
| <p>certificate</p> <p>Specifies the certificate to use to digitally sign a .ZIP file.</p> <p>Configurable</p> <p>Note: This option is available only with SecureZIP.</p> | <p><Name> - The common name of the subject of the certificate (that is, the <i>cn</i> field in a string representation of a certificate; this is the name as viewed in Outlook, Internet Explorer, or PKZIP for Windows); optionally, precede with:</p> <p>cn=</p> <p>If the certificate name contains a space, enclose the certificate name in quotation marks ("My Name").</p> <p><Email address></p> <p>- The email address of the certificate (that is, the <i>e</i> field in a string representation of a certificate); optionally, precede with:</p> <p>e=</p> <p>The specified certificate must exist in the MY certificate store. If more than one</p> | <pre>pkzipc -add -certificate="John Smith" save.zip *.doc pkzipc -add -certificate=cn="John Smith" save.zip *.doc pkzipc -add -certificate=e= john.public@xyz.com save.zip *.doc pkzipc -add -certificate=#mycert.p12 save.zip *.doc</pre> | <p>add, delete, comment, header</p> |

| <i>Name/Description</i> | <i>Value(s)</i> | <i>Example usage</i> | <i>Used with</i> |
|-------------------------|---|----------------------|------------------|
| | <p>certificate in the MY store has the specified name, the first certificate is used.</p> <p>#<file name> - Specifies a PKCS#12 file that contains the certificate you want to use.</p> <p>If the certificate's private key is not in the PKCS#12 file with the certificate, use the keyfile option to point to the separate file that contains the private key. If necessary, use the keypassphrase option to specify a passphrase to read the private key.</p> <p>The certificate option can be used with the hash and sign options. By default, the .ZIP file is signed using the SHA-1 method, and both the central directory and files are signed.</p> | | |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|---|----------------------------|
| <p>comment</p> <p>Include a text comment for files within an archive file. When you run the command, PKZIP prompts you to enter the comment.</p> <p>Configurable</p> | <p>all - All files already in the archive and all files added to or updated in the archive are commented</p> <p>unchanged - Only files that are not changed in the archive are commented</p> <p>add - Only newly added files and versions of files are commented</p> <p>freshen - Only newly added versions of files already in the archive are commented</p> <p>update - Only newly added files and versions of files already in the archive are commented (the opposite of unchanged)</p> <p>none - No comments added</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = add</p> | <p><i>pkzipc -add -comment=all save.zip *.doc</i></p> | <p>add, standalone</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|---|
| <p>configuration</p> <p>Defines default values for PKZIP commands and options</p> | <p><command or option> - Any configurable command or option</p> <p>GUI - Invokes the configuration dialogs from the graphical PKZIP product. If specified, no other command line arguments are processed for configuration except more and silent, which can be set to govern the screen display of configuration settings.</p> <p>silent - Suppresses list of configured settings that is ordinarily displayed after a command or option is configured.</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -config -extract=freshen</i></p> <p>To see the current configuration values, type:</p> <p><i>pkzipc -config</i></p> <p>To open the Configuration dialogs of the GUI product for use in setting configuration defaults:</p> <p><i>pkzipc -config=gui</i></p> <p>Configures overwrite option and suppresses display of settings afterward:</p> <p><i>pkzipc -config=silent -overwrite=never</i></p> <p>Configures silent option and suppresses display of settings afterward:</p> <p><i>pkzipc -config=silent -silent</i></p> | <p>standalone</p> |
| <p>console</p> <p>Extracts files to the screen (standard output) instead of to disk</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -console save.zip *.txt</i></p> | <p>standalone</p> |
| <p>crl</p> <p>Warns if a certificate to be used for digital signing, encryption, or authentication is listed as revoked in an accessible CRL (certificate revocation list).</p> <p>Use with strict option to prevent the certificate from being used.</p> <p>Configurable</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -add -certificate="John Adams" -crl test.zip</i></p> <p><i>pkzipc -add -recipient="John Q. Public" -crl -strict test.zip *.doc</i></p> <p><i>pkzipc -extract -crl test.zip</i></p> | <p>add, comment, delete, extract, header, listfile, print, test, view</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|------------|
| <p><i>cryptalgorithm</i> Encrypted files using the specified encryption algorithm. Configurable</p> | <p>The encryption algorithm to use. The <i>listcryptalgorithms</i> command lists the strong encryption algorithms available to you. Specify a strong encryption algorithm as it is listed in the output from the <i>listcryptalgorithms</i> command. Default = Traditional PKWARE encryption Value if used on command line without a sub-option = The strongest algorithm available on the system</p> | <p>Encrypt all files added with 128-bit AES using the specified passphrase: <i>pkzipc -add -cryptalgorithm=aes,128 -passphrase save.zip *.doc</i></p> <p>Encrypt all files added with 3DES using the certificate named "My friend": <i>pkzipc -add -cryptalgorithm=3DES,168 -recipient="My friend" save.zip *.doc</i></p> <p>Override a configured strong encryption setting to use traditional encryption just for the current command line: <i>pkzipc -add --cryptalgorithm -passphrase save.zip *.doc</i></p> <p>Create an OpenPGP archive called myfile.pgp using CAST5: <i>pkzipc -add -archivetype=pgp -cryptalg=cast5 -myfile.pgp *.txt</i></p> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p>cryptoptions</p> <p>Controls advanced encryption compatibility options.</p> <p>Makes possible a choice of support for smart cards or for certain other target scenarios when doing certificate-based encryption. Only affects encryption done using the recipient option.</p> <p>Configurable</p> | <p>FastAES – Enables the use of the fastest AES algorithm available, OpenSSL for AES over CryptoAPI. If FIPS 140 mode is enabled, this option is not effective.</p> <p>smartcard - Supports certificate-based encryption for recipients using smart cards, but produces encrypted files that cannot be decrypted by older versions of PKZIP. Turn off to support certificate-based encryption for recipients using versions of PKZIP prior to 6.1 at cost of support for smart cards.</p> <p>win2000 - Supports certificate-based encryption for recipients using smart cards or running on Windows NT or Windows 2000, but uses 3DES encryption to protect access to the key when encrypting with AES. Turn off to use no 3DES when encrypting with AES at cost of support for smart cards and recipients running Windows NT or Windows 2000.</p> <p>-----</p> <p>–smartcard and win2000 sub-options are on by default. FastAES is not.</p> | <p>To configure FastAES on :</p> <pre>pkzipc -config -cryptoptions=FastAES</pre> <p>To turn off smartcard in a command line:</p> <pre>pkzipc -add -cryptoptions=-smartcard -recipient="John Q. Public" test.zip</pre> <p>To configure both sub-options off:</p> <pre>pkzipc -config -cryptoptions=-smartcard,-win2000</pre> <p>To configure both sub-options on:</p> <pre>pkzipc -config -cryptoptions=smartcard,win2000</pre> | <p>Add</p> <p>Windows:</p> <p>Extract, Test</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|------------|
| <p>dclimplode</p> <p>Instructs PKZIP to use the data compression library compression scheme.</p> <p>Configurable</p> | <p>ascii - use with ASCII files.</p> <p>binary - use with BINARY or unknown data files.</p> <p>Specify the size of the dictionary (1024, 2048, or 4096) after the type (ascii or binary). Use a comma to separate type and size. A larger size provides more compression.</p> <p>-----</p> <p>No default value</p> | <p><i>pkzipc -add -dclimplode=ascii,4096 text.zip *.txt</i></p> | add |
| <p>default</p> <p>Reset the original defaults in the configuration file for all commands and options</p> | <p>No sub-options</p> <p>No default value.</p> | <p>To reset all defaults:</p> <p><i>pkzipc -default</i></p> | standalone |
| <p>deflate64</p> <p>Compress files using the Deflate64 method.</p> <p>Configurable</p> <p>Note: Files compressed with this method can be extracted by most versions 2.5x and later of PKZIP, but not all ZIP programs from other vendors can extract such files.</p> | <p>No sub-options.</p> <p>No default value.</p> | <p>To compress files using Deflate64 algorithm and level 9 compression:</p> <p><i>pkzipc -add -deflate64 -level=9 save.zip doc1.txt</i></p> <p>To compress files using the normal, default compression level (level 5):</p> <p><i>pkzipc -add -deflate64 save.zip *.doc</i></p> | add |
| <p>delete</p> <p>Remove (delete) files from an archive</p> | <p><files> -Names or file name pattern of files to delete</p> <p>No default value.</p> | <p>For individual files:</p> <p><i>pkzipc -delete save.zip doc1.txt</i></p> <p>For a specific file pattern:</p> <p><i>pkzipc -delete save.zip *.doc</i></p> | standalone |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|--------------------------------|
| <p>directories</p> <p>When adding, includes matching files in subdirectories and stores directory path names; when extracting, recreates saved directory paths.</p> <p>Configurable</p> <p>Note: Using this command is the same as combining the <i>path</i> and <i>recurse</i> commands.</p> | <p>current - Store the path from the current directory.</p> <p>root or full - Store the entire path beginning at the root of the drive; also referred to as "full" path.</p> <p>specify or relative - Store path information relative to the specified directories, for their subdirectories</p> <p>none - No path information stored</p> <p>-----</p> <p>Default = none when used with add; full when used with extract</p> <p>Value if used on command line without a sub-option = current</p> | <p>Compression example (assumes you are in \wp):</p> <pre>pkzipc -add -directories=root save.zip docs*</pre> <p>The path stored is wp/docs/.</p> <pre>pkzipc -add -directories=current save.zip docs*</pre> <p>The path stored is: "docs\".</p> <p>Extraction:</p> <pre>pkzipc -extract -directories save.zip *</pre> | <p>add, extract</p> |
| <p>embedded</p> <p>Suppresses prompt and, depending on the sub-option, extracts or does not extract the contents of a lone archive file embedded in another archive file of the type specified in the sub-option.</p> <p>Configurable</p> | <p>arj - Extract the contents of lone archives embedded in ARJ archives, without prompting</p> <p>-arj - Do not extract the contents of lone archives embedded in ARJ archives, and do not prompt</p> <p>BinHex - Extract the contents of lone archives embedded in BinHex archives, without prompting</p> <p>-BinHex - Do not extract the contents of lone archives embedded in BinHex archives, and do not prompt</p> <p>bzip2 - Extract the contents of lone archives embedded in</p> | <p>To extract an embedded archive from a ZIP file without prompting:</p> <pre>pkzipc -extract -embedded=zip outerarchive.zip</pre> <p>To suppress the prompt and not extract archives embedded in ZIP files:</p> <pre>pkzipc -extract -embedded=-zip outerarchive.zip</pre> | <p>extract, console, print</p> |

| Name/Description | Value(s) | Example usage | Used with |
|------------------|---|---------------|-----------|
| | <p>BZIP2 archives, without prompting</p> <p>-bzip2 - Do not extract the contents of lone archives embedded in BZIP2 archives, and do not prompt</p> <p>cab - Extract the contents of lone archives embedded in CAB archives, without prompting (Windows only)</p> <p>-cab - Do not extract the contents of lone archives embedded in CAB archives, and do not prompt (Windows only)</p> <p>gzip - Extract the contents of lone archives embedded in GZIP archives, without prompting</p> <p>-gzip - Do not extract the contents of lone archives embedded in GZIP archives, and do not prompt</p> <p>lzh - Extract the contents of lone archives embedded in LZH archives, without prompting</p> <p>-lzh - Do not extract the contents of lone archives embedded in LZH archives, and do not prompt</p> <p>rar - Extract the contents of lone archives embedded in RAR archives, without prompting (Windows only)</p> <p>-rar - Do not extract the contents of lone archives</p> | | |

| Name/Description | Value(s) | Example usage | Used with |
|------------------|---|---------------|-----------|
| | <p>embedded in RAR archives, and do not prompt (Windows only)</p> <p>uue - Extract the contents of lone archives embedded in UUENCODED archives, without prompting</p> <p>-uue - Do not extract the contents of lone archives embedded in UUENCODED archives, and do not prompt</p> <p>xxe - Extract the contents of lone archives embedded in XXENCODED archives, without prompting</p> <p>-xxe - Do not extract the contents of lone archives embedded in XXENCODED archives, and do not prompt</p> <p>zip - Extract the contents of lone archives embedded in ZIP archives, without prompting</p> <p>-zip - Do not extract the contents of lone archives embedded in ZIP archives, and do not prompt</p> <p>-----</p> <p>Disabled by default. When used, a sub-option must be set.</p> | | |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|--|---|
| <p>encode</p> <p>As an option, used with <i>add</i>, creates an archive and converts it to the archive type specified by the sub-option. As a standalone command, converts a specified existing archive.</p> <p>Configurable</p> <p>Note: PKZIP creates two files when the <i>encode</i> option is invoked: an intermediate archive of the type specified for the <i>add</i> command (ZIP, by default), and an archive of the type specified for the <i>encode</i> option.</p> <p>Use the <i>movearchive</i> option with <i>encode</i> to remove (delete) the intermediate archive.</p> | <p>bzip2 - Creates a BZIP2 file</p> <p>gzip - Creates a GZIP file</p> <p>uue - Creates a UUENCODED file</p> <p>xxe - Creates an XXENCODED file</p> <p>-----</p> <p>Default value = uue</p> <p>Value if used on command line without a sub-option = uue</p> | <p>Add files to save.zip and encode to UUE:</p> <pre>pkzipc -add -encode save.zip *</pre> <p>Add files to a TAR archive and encode to a GZIP archive:</p> <pre>pkzipc -add -encode=gz save.tar</pre> <p>Encode the archive as a GZIP archive and delete the intermediate archive created by the <i>add</i> command:</p> <pre>pkzipc -add -encode=gz -movearchive save.tar *</pre> <p>As a command, creates save.tar.gz from existing archive save.tar:</p> <pre>pkzipc -encode=gz save.tar</pre> | <p>add</p> |
| <p>enterlicensekey</p> <p>Prompts for a product license key</p> | <p>None</p> | <pre>pkzipc -enterlicensekey</pre> | <p>standalone</p> |
| <p>error</p> <p>Designates warning conditions, by warning number, to treat as error condition 73 (<i>Warning configured as an error</i>)</p> <p>Configurable</p> | <p><warning number> - One or more warning numbers, separated by commas. To override a warning number configured for the option (and thus <i>not</i> treat that warning as an error), precede the number with a hyphen.</p> | <pre>pkzipc -extract -error=42,43 files.zip</pre> <pre>pkzipc -extract -error=42,-43 files.zip</pre> | <p>add, extract, test, view</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|---|
| <p>exclude</p> <p>Exclude files from being compressed or extracted.</p> <p>Configurable separately for add and extract operations.</p> <p>Note: You must specify a sub-option (for example, <i>file pattern</i> or <i>list file name</i> preceded by an appropriate list character "@" with the <i>exclude</i> option.</p> | <p>The file(s) or file pattern (for example, *.doc) being excluded.</p> <p>No default value.</p> | <p>Compression example:</p> <pre>pkzipc -add -exclude="*.doc" save.zip</pre> <p>Extraction example:</p> <pre>pkzipc -extract -exclude="*.txt" save.zip</pre> <p>Setting exclude default:</p> <pre>pkzipc -config -exclude="*.txt"</pre> <p>Note: When you use the <i>exclude</i> option with the <i>configuration</i> command, PKZIP prompts you to configure the <i>exclude</i> default for <i>add</i> and/or <i>extract</i> operations.</p> | <p>add, extract, delete, test, view, print, console</p> |
| <p>extract</p> <p>Extracts files from an archive file</p> <p>Configurable</p> | <p>all - Extracts all files in an archive file</p> <p>freshen - Extracts only files in the archive that are newer versions of files that already exist in the target directory</p> <p>update - Extracts files in the archive that are newer versions of files that already exist in the target directory or that do not exist in the target directory</p> <p>Default = all</p> | <pre>pkzipc -extract save.zip</pre> <pre>pkzipc -extract=update save.zip</pre> | <p>standalone</p> |
| <p>fast</p> <p>Uses the Deflate algorithm and sets the level of compression to level 2 on a scale of 0 - 9. Files having the following extensions are added uncompressed: bz2, bzip2, cab, gz, gzip, rar, gif, jpeg, jpg, mp3, mpeg, mpg, sxw</p> <p>Configurable</p> | <p>No sub-options.</p> <p>No default value.</p> | <pre>pkzipc -add -fast save.zip *.doc</pre> <pre>pkzipc -config -fast</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|---|
| <p>fipsmode</p> <p>Causes SecureZIP to use only algorithms that comply with the FIPS 140 standard to perform cryptographic operations.</p> <p>Use the commands listcryptalgorithms and listhashalgorithms with the fipsmode option to see lists of algorithms available with fipsmode.</p> <p>Configurable</p> | <p>enabled - Turns the option on</p> <p>disabled - Turns the option off</p> <p>On Windows XP and later, the option is enabled or disabled by default according to the Windows FIPS policy setting "System cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing." Otherwise, disabled by default.</p> | <p>Turns on fipsmode for the current command line:</p> <pre>pkzipc -add -recipient="John Public" -fipsmode save.zip *.doc</pre> <p>Overrides a configured default setting of fipsmode=enabled and turns off fipsmode for the current command line:</p> <pre>pkzipc -extract -fipsmode=disabled wedding_plans.zip *.*</pre> <p>Lists encryption algorithms available with fipsmode:</p> <pre>pkzipc -listcryptalgorithms -fipsmode</pre> | <p>add, extract, test, listCryptAlgorithms, listHashAlgorithms</p> <p>With file name-encrypted (FNE) archives, also applies to: comment, delete, header, view</p> |
| <p>fix</p> <p>Attempts to repair a corrupt ZIP archive file</p> | <p><file name> - The name of the ZIP archive to fix</p> <p>No default value.</p> | <pre>pkzipc -fix save.zip</pre> | <p>standalone</p> |
| <p>hash</p> <p>Sets the hashing algorithm to use when signing an archive. Use listhashalgorithms to list hashing algorithms available.</p> <p>Configurable</p> <p>Note: This option requires SecureZIP.</p> | <p>The hashing algorithm to use, as listed in the output from the listhashalgorithms command.</p> <p>Default = sha1</p> | <pre>pkzipc -add -certificate="John Smith" -hash=sha1 save.zip *.doc</pre> | <p>add, delete, comment, header</p> |
| <p>header</p> <p>Creates a comment for a ZIP archive file in the header area of the file</p> <p>Configurable</p> | <p><file name> - The file that contains the header comment. The file name must be prefixed with the ListChar symbol ("@" by default) to distinguish it from the other sub-option</p> <p><comment> - The literal comment to be used</p> <p>----- -</p> <p>No default value.</p> | <p>To include literal text:</p> <pre>pkzipc -add -header save.zip *.doc</pre> <p>Note: PKZIP prompts for the header text</p> <p>To include an existing file:</p> <pre>pkzipc -add -header=@text.doc save.zip *.doc</pre> | <p>add, standalone</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p>help</p> <p>Displays help screen for PKZIP</p> | <p><command or option> - Any command or option for which help is desired.</p> <p>No default value.</p> | <p><i>pkzipc -help</i></p> <p>Display help for the add command:</p> <p><i>pkzipc -help=add</i></p> | <p>standalone</p> |
| <p>include</p> <p>Include files to compress or extract.</p> <p>Configurable separately for add and extract operations.</p> <p>Note: You must specify a sub-option (for example, file pattern or list file name preceded by an appropriate list character "@") with the include option.</p> | <p>The file(s) or file pattern (for example, *.doc) being included.</p> <p>No default value.</p> | <p>Compress only .doc files:</p> <p><i>pkzipc -add -include="*.doc" save.zip</i></p> <p>Configure default behavior to always include .txt files in folders accessed by the command line even if the command line does not explicitly include them, as long as the command line does not explicitly exclude them:</p> <p><i>pkzipc -config -include="*.txt"</i></p> <p>Note: When you use the include option with the configuration command, PKZIP prompts you to specify whether to configure the option for <i>add</i> and/or <i>extract</i> operations.</p> | <p>add, extract, delete, test, view, print, console</p> |
| <p>jobid</p> <p>Specifies a job ID token used with substitution to construct a destination folder name.</p> | <p><ID> - The job ID to use</p> | <p><i>pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc</i></p> | <p>substitution</p> |
| <p>keyfile</p> <p>Specifies a file containing the private key for the certificate specified by the certificate option. The option is most useful when using SSL server certificates, which often have the private key and certificate in separate files.</p> <p>Configurable</p> | <p><file name> - The name and location of the file</p> | <p><i>pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key save.zip *.doc</i></p> | <p>add, extract, test, view</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|--|
| <p>keypassphrase</p> <p>Specifies the passphrase used to decrypt private key information. This can be the passphrase used for a PKCS#12 file (specified with the certificate option), or a key file specified with the keyfile option.</p> | <p><passphrase> - The passphrase, in quotes</p> | <pre>pkzipc -add -certificate=#mycert.p12 -keypassphrase="my password" save.zip *.doc pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key -keypassphrase="my password" save.zip *.doc</pre> | <p>add, extract, test, view</p> |
| <p>larger</p> <p>Process only those files whose size is greater than (in bytes) or equal to a specified file size.</p> <p>Configurable separately for add and extract operations.</p> | <p>Numerical value (in bytes) that indicates a minimum desired file size.</p> <p>No default value.</p> | <p>Add only files that are at least 5000 bytes in size:</p> <pre>pkzipc -add -larger=5000 save.zip *</pre> | <p>add, extract, test, view, delete, print console</p> |
| <p>level</p> <p>Sets the level of compression.</p> <p>See also the options store, speed, fast, normal, and maximum, which provide non-numeric names for various compression settings with (except store) the Deflate compression method.</p> <p>Configurable</p> | <p>Any digit from 0 through 9, with 0 being no compression at the fastest speed, and 9 being the most compression at the slowest speed.</p> <p>Default = level 5 (normal)</p> | <pre>pkzipc -add -level=9 save.zip *.doc</pre> | <p>add</p> |
| <p>license</p> <p>Displays the product license information for PKZIP</p> | <p>No sub-options.</p> <p>No default value.</p> | <pre>pkzipc -license</pre> | <p>standalone</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|---|
| <p>listcertificates</p> <p>Lists digital certificates in a certificate store.</p> <p>Note: This option is available only with SecureZIP.</p> | <p>my - Lists personal certificates in the MY store</p> <p>addressbook - Lists public certificates in the AddressBook store</p> <p>ca - Lists intermediate, certificate authority certificates in the CA store</p> <p>root - Lists trusted certificates in the Root store</p> <p>Default = my</p> | <p><i>pkzipc -listcertificates</i></p> <p><i>pkzipc -listcertificates=addressbook</i></p> | standalone |
| <p>listchar</p> <p>Set the list character to the specified ASCII character. Prefixing a file name with the list character identifies it as a list file.</p> <p>Configurable</p> | <p>Any character in the printable ASCII range. Must not be the same as OptionChar and must not be "-".</p> <p>default = @</p> | <p><i>pkzipc -config -listchar=+</i></p> | All commands except listcertificates, listcryptalgorithms, listsfxtypes, license, and version |
| <p>listcryptalgorithms</p> <p>Displays a list of the strong encryption algorithms available for use with the cryptalgorithm option. With fipsmode on, it lists only FIPS-validated algorithms.</p> <p>When OpenPGP is enabled through configuration or <code>-archivetype=pgp</code>, CAST,128 will also be listed.</p> <p>Note: This option is only available in versions that have strong encryption.</p> | None | <p><i>pkzipc -listcryptalgorithms</i></p> | standalone |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|---|
| <p>listfile</p> <p>Generates a text file that lists the files to be added to or extracted from an archive. The option causes a list file to be created instead of actually adding or extracting files.</p> | <p>Requires a name for the list file</p> <p>No default value.</p> | <p>Create a list file of files that the command line minus the listfile option would add to myarchive.zip:</p> <pre>pkzipc -add=update -listfile=mylist.txt myarchive.zip *</pre> <p>Generate a list file that lists all files, with any saved path information, that the command line minus the listfile option would extract from the save.zip archive</p> <pre>pkzipc -extract - listfile=list.txt save.zip</pre> | <p>add, extract</p> |
| <p>listhashalgorithms</p> <p>Displays a list of the hash algorithms available to the hash option. With fipsmode on, it lists only FIPS-validated algorithms.</p> | <p>None</p> | <pre>pkzipc - listhashalgorithms</pre> | <p>standalone</p> |
| <p>listsfxtypes</p> <p>Display a list of the types of SFX files that can be created with PKZIP</p> | <p>No sub-options.</p> <p>No default value.</p> | <pre>pkzipc -listsfxtypes</pre> | <p>standalone</p> |
| <p>locale</p> <p>Sets the default PKZIP time and date settings to match your system time and date formats. When disabled, PKZIP uses a 12-hour time format and a date format of MMDDYY.</p> <p>Configurable</p> | <p>enable - Turns the option on</p> <p>disable - Turns the option off</p> <p>Default = enable</p> | <p>Configure the option to be off by default:</p> <pre>pkzipc -config -locale=disable</pre> <p>Turn the option off for the current command line</p> <pre>pkzipc -add - locale=disable test.zip *.doc</pre> | <p>All commands except list-certificates, listcryptalgorithms, listsfxtypes, license, and version</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|-----------|
| <p>lowercase</p> <p>Extracts file name(s) in lower case regardless of how it was originally archived.</p> <p>Configurable</p> | <p>disable – Do not change the case of extracted files.</p> <p>archive - extracts file name(s) such that only path information stored in the archive becomes lower case.</p> <p>full - Extracts file name(s) such that all path information stored in the archive becomes lower case, and any extract path specified on the command line also becomes lower case.</p> <p>-----</p> <p>default = archive.</p> | <p><i>pkzipc -extract -lowercase=full save.zip MixedCaseExtract/</i></p> <p>Extracts all files from save.zip using lower case letters to a directory named mixedcaseextract. That directory's name will have lower case letters.</p> <p><i>pkzipc -extract -lowercase-archive save.zip MixedCaseExtract/</i></p> <p>Extracts all files from save.zip using lower case letters to a directory named MixedCaseExtract. That directory's name will have mixed case letters.</p> | extract |
| <p>lzma</p> <p>Compress files using the LZMA method.</p> <p>Configurable</p> <p>Note: Files compressed with this method can be extracted by PKZIP versions 12.3 and later, but not all ZIP programs from other vendors can extract such files.</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -add -lzma save.zip doc1.txt</i></p> | add |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---------------------|
| <p>mask</p> <p>Strips file attributes that the attribute option would otherwise cause to be stored or set for extracted files</p> <p>Configurable</p> <p>Note: You can only mask attributes that are specified with the attributes option.</p> | <p>hidden - hidden attributes.</p> <p>archive - archive attribute.</p> <p>system - system attributes.</p> <p>readonly - read-only attributes.</p> <p>none - no attributes (turns off attribute mask in the PKZIP Configurations Settings file for this instance only).</p> <p>all - all attributes</p> <p><hex value> -The hex value of an attribute to be masked, or the logical OR of multiple hex values</p> <p>-----</p> <p>Default (add) = none</p> <p>Default (extract) = all</p> <p>Value if used on command line without a sub-option (add and extract) = all</p> | <pre>pkzipc -add - attributes=all -mask=hidden save.zip pkzipc -extract - mask=none save.zip pkzipc -config - mask=hidden</pre> | <p>add, extract</p> |
| <p>maximum</p> <p>Uses the Deflate compression method and sets the level of compression to level 9, the highest level on a 0 - 9 scale, but gives the lowest speed</p> <p>Configurable</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <pre>pkzipc -add -maximum save.zip *.doc pkzipc -config -maximum</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|---------------------|
| <p>messagedigest</p> <p>Display one or more message digests for files inside an archive.</p> | <p>All – Calculates and displays the message digest for all of the algorithms.</p> <p>None – Don't display any message digests or checksums. This is useful for displaying only one.</p> <p>CRC32 – Calculates and displays CRC32 checksum</p> <p>MD5 – Calculates and displays MD5 message digest.</p> <p>SHA1 – Calculates and displays SHA-1 message digest.</p> <p>SHA256 – Calculates and displays SHA-256 message digest.</p> <p>SHA384 – Calculates and displays SHA-384 message digest.</p> <p>SHA512 – Calculates and displays SHA-512 message digest.</p> <p>Prefix an algorithm with - to indicate it should not be used</p> <p>-----</p> <p>Default = all</p> | <p>Shows the message digest using all available hash algorithms for all files inside archive.zip</p> <pre>pkzipc -messagedigest archive.zip</pre> <p>Shows the message digest using all available hash algorithms for only file.doc inside archive.zip</p> <pre>pkzipc -messagedigest archive.zip file.doc</pre> <p>Uses the sha256sum program to verify that file.doc inside archive.zip is the same as file.doc in the current directory.</p> <pre>pkzipc -messagedigest=none,sha256 -silent=banner archive.zip file.doc sha256sum --check</pre> | <p>Standalone</p> |
| <p>more</p> <p>Pauses after one screen of output and prompts to continue.</p> <p>Configurable</p> | <p>The number of rows of information you want to define as a screen</p> <p>-----</p> <p>Default = one screen of information</p> | <pre>pkzipc -view -more=22 save.zip</pre> <pre>pkzipc -config -more</pre> | <p>All commands</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p>move</p> <p>Removes (deletes) files from the source drive after adding them to an archive.</p> | <p>No sub-options. ----- No default value.</p> | <p><code>pkzipc -add -move save.zip *.doc</code></p> | <p>add</p> |
| <p>movearchive</p> <p>Deletes an archive that is created only as an intermediate archive—for example, to be converted by the <i>encode</i> option to an archive of a different type, or to be transferred by FTP.</p> <p>Configurable</p> | <p>No sub-options. ----- No default value.</p> | <p><code>pkzipc -add -encode=gzip -movearchive myfiles.tar</code></p> | <p>add</p> |
| <p>namesfx</p> <p>Specify a file name when converting to a self-extracting file.</p> | <p><file name> - File name for the SFX file ----- No default value.</p> | <p><code>pkzipc -sfx -namesfx=test.exe docs.zip</code></p> | <p>sfx</p> |
| <p>newer</p> <p>Selects files that are no older than a specified interval</p> <p>Configurable separately for add and extract operations</p> <p>Note: With a time unit of days, the interval is measured from the beginning of the current day. With time units of hours, minutes, or seconds, the interval is measured from the current system time.</p> <p>Note: To specify an explicit date, see after..</p> | <p><numeric value> A number of days, hours, minutes, or seconds defining the interval, plus a suffix identifying the kind of units used:</p> <p>Suffixes: d - Days (default) h - Hours m - Minutes s - Seconds ----- No default value.</p> | <p>Add files no older than 24 hours: <code>pkzipc -add -newer=24h save.zip *</code></p> <p>Add files no older than five days: <code>pkzipc -add -newer=5d save.zip *</code></p> <p><code>pkzipc -add -newer=5 save.zip *</code></p> | <p>add, extract, test, view, print, console</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p><i>noarchiveextension</i></p> <p>Suppresses adding a file name extension to the specified archive file name</p> <p>Configurable</p> <p>Note: This option is identical to <i>nozipextension</i>, which is now deprecated.</p> | <p>No sub-options. -----</p> <p>No default value.</p> | <p><i>pkzipc -add -noarchiveextension file.ibm *.doc</i></p> | <p>All commands except list-certificates, listcryptalgorithms, listsfxtypes, license, and version</p> |
| <p><i>noextended</i></p> <p>Suppress the storage of extended attribute information (excluding file permission attributes)</p> <p>Configurable</p> | <p>No sub-options. -----</p> <p>No default value.</p> | <p><i>pkzipc -add -noextended save.zip *</i></p> | <p>add</p> |
| <p><i>nofix</i></p> <p>Suppress the attempt to fix any problems PKZIP encounters in extracting from an archive</p> <p>Configurable</p> | <p>No sub-options. -----</p> <p>No default value.</p> | <p><i>pkzipc -add -nofix save.zip *.doc</i></p> | <p>All commands except list-certificates, listcryptalgorithms, listsfxtypes, license, and version</p> |
| <p><i>normal</i></p> <p>Uses the Deflate algorithm and sets the level of compression to 5 (normal) on a scale of 0 - 9 for a balance of compression and speed. Unlike with the <i>fast</i> option, all files are compressed.</p> <p>Configurable</p> | <p>No sub-options. -----</p> <p>No default value. dod5220 - Overwrites files three times, to the DOD 5220.22-M specification</p> | <p><i>pkzipc -add -normal save.zip</i></p> <p><i>pkzipc -config -normal</i></p> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|--|---------------------|
| <p><i>nosmartcard</i></p> <p>Note: This option is deprecated. Instead of setting nosmartcard, turn off the smartcard sub-option of cryptoptions.</p> <p>Turns off smart card compatibility when set in conjunction with the recipient option.</p> <p>Set this option to enable users of versions of PKZIP prior to 6.1 to decrypt files encrypted using the recipient option.</p> <p>Note: Smart cards cannot decrypt files encrypted using a recipient list if this option is set.</p> <p>configurable</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <pre>pkzipc -add -recipient="Thomas Francis, Jr." nosmartcard save.zip *.doc</pre> | <p>add</p> |
| <p><i>nozipextension</i></p> <p>Note: This option is deprecated. Use the option noarchiveextension instead.</p> <p>Suppress PKZIP's adding of an identifying file extension to an archive file name</p> <p>Configurable</p> | <p>No sub-options.</p> <p>-----</p> <p>No default value.</p> | <pre>pkzipc -add - nozipextension file.ibm *.doc</pre> | <p>All commands</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|--|---|
| <p>older</p> <p>Selects files that are older than a specified interval</p> <p>Configurable separately for add and extract operations</p> <p>Note: With a time unit of days, the interval is measured from the beginning of the current day. With time units of hours, minutes, or seconds, the interval is measured from the current system time.</p> <p>Note: To specify an explicit date, see before.</p> | <p><numeric value></p> <p>A number of days, hours, minutes, or seconds defining the interval, plus a suffix identifying the kind of units used:</p> <p>Suffixes:</p> <p>d - Days (default) h - Hours m - Minutes s - Seconds</p> <p>-----</p> <p>No default value.</p> | <p>Adds files older than 24 hours:</p> <pre>pkzipc -add -older=24h save.zip *</pre> <p>Adds files older than five days:</p> <pre>pkzipc -add -older=5d save.zip *</pre> <pre>pkzipc -add -older=5 save.zip *</pre> | <p>add, extract, test, view, print, console</p> |
| <p>OpenFile</p> <p>Determines whether to include files that are open for write access in another application</p> <p>Note: This option is not needed in UNIX and Linux systems, as this is the default behavior.</p> | <p>Never - PKZIP does not include any open files. A warning will appear if a matching file is open</p> <p>All - PKZIP includes all matching open files without prompting first. A message noting each open file is included in the standard output.</p> <p>Prompt - PKZIP notifies you when a matching file is open, and asks whether to add the open file or skip it.</p> <p>-----</p> <p>Default = Never</p> | <pre>pkzipc -add -OpenFile=never test.zip *.bmp</pre> <pre>pkzipc -add -OpenFile test.zip *.bmp</pre> <pre>pkzipc -add -OpenFile=prompt test.zip *.bmp</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---------------------|
| <p>optionchar</p> <p>Specifies the prefix character used to identify a command or option as such on the command line</p> <p>Note: On Windows, the "/" (slash) character can also always be used.</p> <p>Configurable</p> | <p>Any valid single character.</p> <p>-----</p> <p>Default = - (hyphen)</p> | <pre>pkzipc -optionchar=+ +add save.zip *.doc pkzipc +config - optionchar=+</pre> | <p>All commands</p> |
| <p>overwrite</p> <p>Specifies whether to overwrite existing files with files being added or extracted. By default, PKZIP prompts before overwriting when extracting but not when adding.</p> <p>Configurable</p> | <p>prompt - Prompt every file individually on whether to overwrite a file that has the same name as the one being added or extracted</p> <p>all - Overwrite all files that have the same name</p> <p>increment - Increment file name to make it unique.</p> <p>never - Never overwrite a file that already exists in the target directory or archive</p> <p>-----</p> <p>Value if used on command line without a sub-option = all.</p> | <pre>pkzipc -extract - overwrite=all save.zip pkzipc -add -overwrite=prompt save.zip</pre> | <p>add, extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p>passphrase</p> <p>Protects an archive with passphrase-based encryption</p> <p>PKZIP prompts for a passphrase if none is specified with the option.</p> <p>Configurable</p> | <p><passphrase> - The passphrase that must be supplied to extract and decrypt the files</p> <p><file name> - Name of a file that contains the text of the passphrase. The file name must be prefixed with the list character (@ by default) defined with the listchar option.</p> <p>-----</p> <p>No default value.</p> | <p>To include a passphrase in the command:</p> <pre>pkzipc -add -passphrase=beowulf save.zip</pre> <p>To have PKZIP prompt for a passphrase after you type the command:</p> <pre>pkzipc -add -passphrase save.zip</pre> <p>To have PKZIP get the passphrase from a file:</p> <pre>pkzipc -add -passphrase=@secret.txt save.zip</pre> <p>To extract passphrase-protected files from an archive:</p> <pre>pkzipc -extract -passphrase=beowulf9 save.zip</pre> | <p>add, extract, test, print, console</p> |
| <p>path</p> <p>Stores or restores directory path names for files within a .ZIP file</p> <p>By default, PKZIP does not store path information</p> <p>Configurable</p> | <p>current - Store the path from the current directory.</p> <p>root or full - Store the entire path beginning at the root of the drive; also referred to as "full" path.</p> <p>specify or relative - Store path information relative to the specified directories, for their subdirectories</p> <p>none - No path information stored</p> <p>-----</p> <p>Default = none when used with add; full when used with extract</p> <p>Value if used on command line without a sub-option = current</p> | <p>Assuming you are in "/temp":</p> <pre>pkzipc -add -path=root save.zip docs/*</pre> <p>(the complete path is stored including "temp/docs/").</p> <pre>pkzipc -add -path=current save.zip docs/wp/*</pre> <p>(the path stored is "docs/wp").</p> | <p>add, extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|--|
| <p>ppmd</p> <p>Compress files using the PPMd method.</p> <p>Configurable</p> <p>Note: Files compressed with this method can be extracted by PKZIP versions 12.3 and later, but not all ZIP programs from other vendors can extract such files.</p> | <p>No sub-options. ----- No default value.</p> | <p><i>pkzipc -add -ppmd save.zip doc1.txt</i></p> | <p>add</p> |
| <p>preview</p> <p>Prints out messages to preview the results of a set of commands or options without actually performing the tasks</p> | <p>No sub-options. ----- No default value.</p> | <p><i>pkzipc -add -preview save.zip</i></p> | <p>add, delete, header, sfx, comment</p> |
| <p>print</p> <p>Print a file within a .ZIP file. (Windows)</p> | <p><print device> - The print device use, for example, "lpt1". ----- Default = the default printer on your system.</p> | <p><i>pkzipc -print=lpt1 save.zip readme.txt</i></p> <p>Uses default printer if no printer is specified.</p> | <p>standalone</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|------------|
| <p>recipient</p> <p>Specifies one or more recipients for certificate-based encryption. The option can appear more than once on the command line to specify multiple recipients.</p> <p>Configurable</p> <p>Note: Use the recipient option with the nosmartcard option if you want users of versions of PKZIP prior to 6.1 to be able to decrypt your files.</p> <p>Note: This option is available only with SecureZIP.</p> | <p>cn=<Common name> - The Common Name (CN) field of the subject of the certificate. The "cn=" prefix is optional. This sub-option is the default: PKZIP searches the Common Name field if no other field is specified.</p> <p><Friendly name> - The friendly name associated with the certificate. This is often the same as the common name of the subject.</p> <p>e=<email address> - The email address embedded in the subject of a digital certificate. (Note: Not all certificates contain an email address.) The "e=" prefix is optional.</p> <p>@<file name> - Specifies a text file which contains the names of recipients, one on each line.</p> <p>#<file name> - Specifies a PKCS#7 or PKCS#12 file that contains certificates of the recipients you want to list.</p> <p>-----</p> <p>Default = cn=</p> | <pre>pkzipc -add -recipient="Thomas Jones, Jr." save.zip *.doc pkzipc -add -recipient="cn=Thomas Jones, Jr." save.zip *.doc pkzipc -add - recipient=e=john.public@ nowhere.com save.zip *.doc pkzipc -add -recipient=john.public@ nowhere.com save.zip *.doc pkzipc -add -recipient= f>(&(userCertificate=*) (ou=Sales)) save.zip *.doc pkzipc -add -recipient= "f>(&(userCertificate=*) (ou=Sales With A Space))" save.zip *.doc pkzipc -add - recipient=@recipients.txt save.zip *.doc pkzipc -add - recipient=#recipients.p7 b save.zip *.doc pkzipc -add - recipient=#recipients.p1 2 save.zip *.doc</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|-------------------------------|
| <p>recurse</p> <p>Search subdirectories for files to compress</p> <p>Use with path to store path information for files in subdirectories. Tip: You can use directories to combine the functionality of recurse and path.</p> <p>Configurable</p> | <p>No sub-options. ----- No default value.</p> | <pre>pkzipc -add -recurse save.zip *</pre> | <p>add</p> |
| <p>rename</p> <p>Uses regular expressions to rename files as they are added or extracted.</p> | <p>@<list file> - A list file specifying replacement expressions, one on each line</p> <p><Replacement expression> - A separator character followed by a regular expression followed by another separator character followed by a replacement string followed by a final separator character optionally followed by "i" to ignore case</p> | <pre>pkzipc -add -rename=/blue/green/ mydata.zip *.txt pkzipc -add -rename=/blue/green/i mydata.zip *.txt pkzipc -extract -rename=-/output.txt/ data.zip output.txt</pre> | <p>add, extract, test</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|-----------------------|
| <p>runafter</p> <p>Run or open a specified file after extraction by a self-extractor</p> <p>Configurable</p> | <p><file name> - The file to run or open</p> <p>-----</p> <p>No default value.</p> | <p>Launch the file (for example, readme.txt) via the specified applications (for example, notepad.exe):</p> <pre>pkzipc -add -sfx -runafter="notepad.exe readme.txt" test.exe *</pre> <p>Launch the file (for example, readme.txt) via the associated application (Windows only):</p> <pre>pkzipc -add -sfx -runafter="{ } readme.txt" test.exe *</pre> <p>Run the install script (for example, install.inf) (Windows only):</p> <pre>pkzipc -add -sfx - runafter="{install.inf} " test.exe *</pre> <p>Run the install script (for example, install.inf) with the full short path pre-appended (for example, c:\program~1\temp) (Windows only):</p> <pre>pkzipc -add -sfx -runafter= "{install}%0install.inf " test.exe</pre> | <p>(add) sfx</p> |
| <p>runContext</p> <p>Specifies that program is running in the context of the Current User or Local Machine</p> <p>This option processed earlier than any other option, even earlier than the - OptionChar, as a result it is not configurable and it only works with '-' or '/' as option indicators.</p> | <p>user - program is running in the context of the Current User</p> <p>machine - program is running in the context of the Local Machine</p> <p>-----</p> <p>Default = user</p> | <pre>pkzipc -extract archive.zip</pre> <p>Extracts archive in the context of the Current User</p> <pre>pkzipc -extract - RunContext archive.zip</pre> <p>Extracts archive in the context of the Current User</p> <pre>pkzipc -extract - RunContext=user archive.zip</pre> <p>Extracts archive in the context of the Current User</p> <pre>pkzipc -extract - RunContext=machine archive.zip</pre> <p>Extracts archive in the context of the Local Machine</p> | <p>extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|------------------------|
| <p>sfx</p> <p>With the add command, creates a self-extracting ZIP file with a .exe file name extension. As a standalone command, converts an existing ZIP file to a self-extracting archive.</p> <p>Configurable</p> <p>Note: For a listing of available self-extractors, use the listsfxtypes command.</p> | <p><no sub-option></p> <p>- Create a native command line self-extractor</p> <p>win32_x86_g610</p> <p>- Create a graphical Windows self-extractor that, when run, opens a dialog to let the user select a target extract folder</p> <p>-----</p> <p>Default = Create a native command line self-extractor for use in the command line environment of the operating system in which it was created</p> | <p>To create myfiles.exe:</p> <pre>pkzipc -add -sfx myfiles *.doc</pre> <p>To convert existing ZIP file myfiles.zip to self-extracting graphical Windows archive myfiles.exe:</p> <pre>pkzipc -sfx=win32_x86_g myfiles.zip</pre> <p>To convert existing ZIP file myfiles.zip to a self-extractor and specify a name for the self-extractor:</p> <pre>pkzipc -sfx -namesfx=newname myfiles.zip</pre> <p>(Converts myfiles.zip to newname.exe.)</p> | <p>add, standalone</p> |
| <p>sfxdestination</p> <p>Specifies a default target folder for files extracted from a self-extractor</p> <p>Configurable</p> | <p><path> - Path to target folder</p> <p>-----</p> <p>No default value</p> | <pre>pkzipc -add -sfx -sfxdestination="My Documents\newstuff" mysfx *.doc</pre> | <p>add, sfx</p> |
| <p>sfxdirectories</p> <p>Causes a self-extractor to restore a saved path structure on extraction. To recurse subdirectories and store path information when adding files to the archive, use with the directories option.</p> <p>Configurable</p> | <p>No sub-options</p> <p>-----</p> <p>No default value</p> | <pre>pkzipc -add -sfx -sfxdirectories -directories mysfx "docs*.*"</pre> | <p>add, sfx</p> |
| <p>sfxlogfile</p> <p>Creates an ASCII text error log (named pkerrlog.txt) in the destination directory on extraction</p> <p>Configurable</p> | <p>No sub-options</p> <p>-----</p> <p>No default value</p> | <pre>pkzipc -add -sfx -sfxlogfile test.exe *</pre> | <p>(add) sfx</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|--|-----------------|
| <p>sfxoverwrite</p> <p>Specifies when a self-extractor overwrites files that have the same name as a file being extracted</p> <p>Configurable</p> | <p>prompt - (Default) The user is asked whether to overwrite files</p> <p>always - Files that have the same name in the destination folders are overwritten without prompting</p> <p>update - Only files that do not already exist or are newer than same-named files</p> <p>freshen - Only newer versions of files that already exist in the destination folders are extracted; the older files are overwritten without prompting</p> <p>never - Files are never overwritten</p> <p>-----</p> <p>Default = prompt</p> | <pre>pkzipc -add -sfx -sfxoverwrite=freshen mysfx *.doc</pre> | <p>add, sfx</p> |
| <p>sfxuietype</p> <p>Specifies the type of graphical interface (GUI) that a self-extractor presents to the user.</p> <p>This option only affects GUI self-extractors. (Command line self-extractors do not present a GUI.)</p> <p>Configurable</p> | <p>autosfx - Presents a dialog that displays a bar to show progress extracting, and a Cancel button</p> <p>easysfx - (Default) Presents a dialog that enables the user to select a destination folder and to turn off any runafter option set</p> <p>regularsfx - Presents a dialog that enables the user to change the destination folder and other options before the archive is extracted</p> <p>-----</p> <p>Default = easysfx</p> | <pre>pkzipc -add -sfx -sfxuietype=regularsfx mysfx *.doc</pre> | <p>add, sfx</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|------------|
| <p>shortname</p> <p>Convert long file names of files added to an archive to WIN32-equivalent "short" file names</p> <p>Configurable</p> | <p>dos - Convert long file names to DOS-equivalent short file names (8+3)</p> <p>none - Do not convert file names</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = dos.</p> | <p><i>pkzipc -add -short=dos save.zip</i></p> | <p>add</p> |
| <p>shred</p> <p>Overwrites PKZIP temporary files and files deleted by PKZIP to prevent recovery of their data</p> <p>Configurable</p> <p>Note: The <i>shred</i> option is the same as the older <i>wipe</i> option, which is now deprecated.</p> | <p>none - turns off shredding: files are not overwritten</p> <p>random - Overwrites files once with random data</p> <p>dod5220 - Overwrites files three times, to the DOD 5220.22-M specification</p> <p>nsa - Overwrites files seven times, to the NSA standard</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = random</p> | <p><i>pkzipc -add -move -shred=nsa myfiles.zip *</i></p> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|--|------------|
| <p>sign</p> <p>Indicates whether the central directory or only files should be signed when using digital signatures. Use the certificate option (which can be configured) to specify the certificate to use.</p> <p>For maximum security, sign both the central directory and local files.</p> <p>Configurable</p> <p>Note: This option requires SecureZIP.</p> | <p>cd - sign central directory.</p> <p>files - sign files.</p> <p>all - sign both the central directory and files.</p> <p>timestamp – Sign files and apply a digital timestamp to the central directory.</p> <p>none - do not sign files (Used for turning signing off if it has been configured)</p> <p>-----</p> <p>Value if used on command line without a sub-option = all.</p> | <pre>pkzipc -add -certificate="John Smith" -sign=cd save.zip *.doc</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|---|---|
| <p>silent</p> <p>Suppresses the display of some or all of PKZIP's messages to the user, including warnings and errors. It can also suppress prompts for inputs.</p> <p>Configurable</p> | <p>none - Turns off the silent option; displays all messages</p> <p>banner - Suppresses printing the banner</p> <p>copy - Suppresses "Copy file" messages when updating archives</p> <p>error - Suppresses all error and warning outputs</p> <p>fileheader - Suppresses file headers when using the console command</p> <p>input - Suppresses all requests for input. If any operation requests input, an error is given</p> <p>normal - Suppresses all message outputs except warnings, errors, and prompts for input</p> <p>output - Suppresses all normal, error, and warning outputs</p> <p>progress - Suppresses "percent complete" messages</p> <p>all - Same as specifying both Input and Output. (Default if option is specified without a sub-option)</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -add -silent save.zip *.doc</i></p> <p><i>pkzipc -config -silent</i></p> | <p>All commands except list-certificates, listcryptalgorithms, listsfxtypes, license, and version</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|---|---|
| <p>smaller</p> <p>Process only files that are smaller than or equal to a given file size, specified in bytes</p> <p>Configurable separately for add and extract operations.</p> | <p>Numerical value that indicates a maximum desired file size (in bytes)</p> <p>-----</p> <p>No default value.</p> | <p><i>pkzipc -add -smaller=5000 save.zip *</i></p> <p>In this example, PKZIP adds only files no larger than 5000 bytes in size.</p> | <p>add, extract, test, view, delete, print, console</p> |
| <p>sort</p> <p>Sort files in an archive based on specific criteria (for example, by file size). Files are then viewed, added, and extracted in the order sorted.</p> <p>Configurable</p> <p>Note: The crc and ratio sub-options do not work with the add command and sort option.</p> | <p>crc - sort by CRC value</p> <p>date - sort by file date of file</p> <p>extension - sort by file extension</p> <p>name - alphabetically sort files and folders together in one series by path name</p> <p>natural - sort in the order files occur in the archive</p> <p>ratio - sort by compression ratio</p> <p>size - sort by the original, uncompressed size of the file ("length" in display)</p> <p>comment - sort by file comment</p> <p>none - first alphabetically sort path names that contain folders and then separately sort file names that lack folder information. (The default.)</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = name</p> | <p><i>pkzipc -add -sort=date save.zip *.doc</i></p> <p><i>pkzipc -config -sort=date</i></p> | <p>add, extract, test, view, delete, print, console</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|------------|
| <p>span</p> <p>Forces PKZIP to create a split archive, even when creating the archive on non-removable media.</p> <p>Also formats or wipes removable media prior to writing an archive.</p> <p>On Windows, the option causes PKZIP to write an archive in segments if necessary to span multiple removable media.</p> <p>This option is available only for ZIP archives.</p> <p>Note: On Windows, spanning should take place automatically when writing to removable media, so the span option does not normally need to be included on the command line.</p> <p>Configurable</p> | <p>Force - Fully format media without checking for existing files</p> <p>Format - Fully format media before attempting to write to it</p> <p>Quick - Quick-format media before attempting to write to it</p> <p>Wipe - Delete contents of media before attempting to write to it</p> <p>None - Do not format or erase media before attempting to write to it</p> <p><segment size> - Split archive into segments of predefined size (see choices below) or a specified size (in bytes) greater than 65535.</p> <p>Predefined sizes:</p> <p>360 = 360KB floppy</p> <p>720 = 720KB floppy</p> <p>1.2 = 1.2MB floppy</p> <p>1.44 = 1.44MB floppy</p> <p>2.88 = 2.88MB floppy.</p> <p>95.7 = 100MB ZIP disk</p> <p>650 = 650MB CD-ROM</p> <p>700 = 700MB CD-ROM</p> <p>-----</p> <p>Default = none</p> | <pre>pkzipc -add -span a:\save.zip *.doc pkzipc -add -span=format a:\save.zip *.doc pkzipc -add -span=1.44 c:\save.zip *.doc pkzipc -add - span=1457664 c:\save.zip *.doc</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|---|-------------------------------------|
| <p>speed</p> <p>Uses the Deflate algorithm and sets the level of compression to 1 on a scale of 0 - 9. Some files are stored (level 0) uncompressed.</p> <p>Provides the fastest performance but the least compression. Files having the following extensions are stored uncompressed: bz2, bzip2, cab, gz, gzip, rar, gif, jpeg, jpg, mp3, mpeg, mpg, sxw</p> <p>Configurable</p> | <p>No sub-options. ----- No default value.</p> | <pre>pkzipc -add -speed save.zip *.doc pkzipc -config -speed</pre> | <p>add</p> |
| <p>store</p> <p>Sets the level of compression to 0 (no compression) on a scale of 0 - 9; stores the files in the archive without compressing them</p> <p>Configurable</p> | <p>No sub-options. ----- No default value.</p> | <pre>pkzipc -add -store save.zip *.doc pkzipc -config -store</pre> | <p>add</p> |
| <p>strict</p> <p>Applies strict checking to allow X.509 certificates to be used only if they are valid and are designated (on the certificate) for use for the intended type of operation (signing or encryption).</p> <p>Configurable</p> | <p>KeyUsage - Controls key usage checks</p> <p>TimeNesting - Controls time nesting checks</p> <p>TimeValid - Controls time validity checks</p> <p>----- No default value</p> | <pre>pkzipc -add -cryptalg -recipient="John Q. Public" -strict test.zip *.doc pkzipc -add - recipient="John Q. Public" -crl -strict test.zip *.doc</pre> | <p>add, delete, comment, header</p> |
| <p>substitution</p> <p>Used with add, inserts a timestamp constructed from specified tokens in the name of a new or updated archive or, when used with the archiveeach option, in the name of the specified destination directory. Used with extract,</p> | <p>Available tokens. These are replaced by associated values on execution:</p> <p>{archivename} - (For use only with extract) Base name of archive, without the extension</p> <p>{archiveext} - (For use only with</p> | <p>This command line using tokens:</p> <pre>pkzipc -add - substitution "Design Spec {yyyy}-{mm}-{dd}- {h}-{MM}-{SS}{ampm}.zip" plan.doc</pre> <p>produces a ZIP file with a name like:</p> <p>Design Spec 2006-08-09-12-06-29am.zip</p> <p>This command line uses the</p> | <p>add, extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|--|-----------|
| <p>dynamically constructs the name of the destination folder from embedded tokens. A single command line can extract multiple archives each to a custom-named folder.</p> <p>Configurable</p> | <p>extract) The file name extension of the archive (without a leading dot)</p> <p>{archivepath} - (For use only with extract) The path of the archive, without the file name, preceded by a leading slash or backslash and excluding the drive letter or share path if the name is a UNC name</p> <p>{id} - A job ID specified separately with the jobid option</p> <p>{mm} - Month, 2-digit</p> <p>{m} - Month, 1-digit (if possible)</p> <p>{dd} - Day, 2-digit</p> <p>{d} - Day, 1-digit (if possible)</p> <p>{yyyy} - Year, 4-digit</p> <p>{yy} - Year, 2-digit</p> <p>{HH} - Hour, 2-digit, 24-hour format</p> <p>{H} - Hour, 1-digit (if possible), 24-hour format</p> <p>{hh} - Hour, 2-digit, 12-hour format</p> <p>{h} - Hour, 1-digit (if possible), 12-hour format</p> <p>{MM} - Minute, 2-digit</p> <p>{M} - Minute, 1-digit (if possible)</p> <p>{SS} - Second, 2-digit</p> <p>{S} - Second, 1-digit (if possible)</p> <p>{ampm} - a.m. or p.m. indicator to identify current 12-hour segment</p> | <p><i>jobid</i> option to set a value for {id):</p> <pre>pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc</pre> <p>and results in a ZIP file with a name like:</p> <p>myJob2006.zip</p> <p>Extracts all ZIP files in the current directory, each to a subdirectory named after the ZIP archive extracted there</p> <pre>pkzipc -extract - substitution *.zip {archivename}\</pre> | |

| Name/Description | Value(s) | Example usage | Used with |
|--|---|--|-----------------------------------|
| | of the day No sub-options ----- No default value | | |
| temp Specifies the directory to use for temporary files created by PKZIP Configurable | The drive and/or path. For example: C: or /root/temp ----- No default value. | Update the .ZIP file test.zip and uses the z:\public directory location for temporary files: <pre>pkzipc -add -temp=z:\public test.zip *.txt</pre> Updates the .ZIP file test.zip and uses the /temp directory location for temporary files: <pre>pkzipc -add -temp=/temp test.zip *.txt</pre> | add, delete, sfx, header, comment |
| test Tests the integrity of files in a ZIP file to ensure that they can be extracted. Also authenticates signatures. Configurable | all - all files in the archive file are tested freshen - tests only those files in the archive that are newer versions of files that already exist in the extract directory update - tests files in the archive that are newer versions of files that already exist in the extract directory or that do not already exist there ----- Default = all | <pre>pkzipc -test save.zip</pre> | standalone |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|----------------|
| <p><i>times</i></p> <p>Specifies that PKZIP should restore the extended time fields, and/or other dates stored in the archive.</p> <p>Configurable</p> | <p>access - restores the time of last access to file(s) on extraction.</p> <p>modify - restores the time of last modification to files on extraction.</p> <p>create - restores the time of creation to files on extraction (Windows).</p> <p>all - all file times are restored.</p> <p>none - file times are not restored.</p> <p>-----</p> <p>Default = all</p> | <p><i>pkzipc -extract -times=access save.zip</i></p> | <p>extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|---|
| <p>translate</p> <p>Translates EOL ("end of line") characters when adding or extracting files. For .ZIP archives, the translation occurs only for files which are marked as ASCII. For other archive types, the translation may occur on all files, including binary files.</p> <p>The ebcdic sub-options work only with data compressed using SecureZIP for z/OS or SecureZIP for i5/OS with the Zip Descriptor Word (ZDW) option to preserve variable length records.</p> <p>Configurable</p> | <p>none - no translation is performed.</p> <p>dos - translates text files so that lines end with a return/newline pair (Windows default)</p> <p>mac - translates text files so lines end with a single carriage return</p> <p>unix - translates text files so lines end with a single newline</p> <p>ebcdic,nl - With ZDW files, substitute EBCDIC newline (0x15)</p> <p>ebcdic,lf - With ZDW files, substitute EBCDIC linefeed (0x25)</p> <p>ebcdic,crlf - With ZDW files, substitute EBCDIC carriage return/linefeed (0x0D25)</p> <p>ebcdic,lfcrlf - With ZDW files, substitute EBCDIC linefeed/carriage return (0x250D)</p> <p>ebcdic,crnl - With ZDW files, substitute EBCDIC carriage return/newline (0x0D15)</p> <p>remove - Remove end of line marks</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = native operating system compatibility translation.</p> | <pre>pkzipc -extract -<i>translate=unix</i> save.zip pkzipc -add -<i>translate=unix</i> scripts.zip *.pl</pre> | <p>add, extract, console, print</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|---------------------------------------|
| <p>ts</p> <p>Contact a Time Stamp Authority (TSA) with the supplied URL to apply a digital timestamp to the archive.</p> <p>Configurable</p> | <p>Syntax (optional fields in brackets):</p> <p>ts= [username[:pwd @]server [:port/]page</p> <p>where:</p> <p>username (optional) is the user account with which to log in if the FTP server requires a login</p> <p>pwd (optional) is the passphrase associated with the user account. Colons are not allowed in the passphrase.</p> <p>server is the TSA server name</p> <p>port is the TCP/IP port to use.</p> <p>No sub-options</p> | <p>Add files and digitally sign and timestamp the archive test.zip using the "My Name" certificate:</p> <pre>pkzipc -add - sign=timestamp - certificate="My Cert" - ts=http://tsa.example.com/tsa test.zip *.txt</pre> <p>Digitally sign and timestamp archive the archive test.zip using the "My Name" certificate</p> <pre>pkzipc -sign=timestamp - ts=http://tsa.example.com/tsa -certificate="My Name" test.zip</pre> | <p>sign</p> |
| <p>utf8</p> <p>Enables UTF-8 characters in file names and file comments to be correctly displayed when an archive's contents are viewed or extracted in compatible non-UTF-8 locales</p> <p>Configurable</p> | <p>No sub-options. No default value.</p> | <pre>pkzipc -add -utf8 test.zip *.*</pre> | <p>add, comment</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|---|---|----------------|
| <p>verifysigner</p> <p>Specifies one or more certificates and constrains PKZIP to extract only archives whose central directories are signed using one of these certificates. PKZIP must also find the specified certificates locally.</p> <p>The option can appear more than once in the same command line, to specify multiple certificates.</p> <p>Configurable</p> | <p>cn=<Common name> - The Common Name (CN) field of the subject of the certificate. The "cn=" prefix is optional.</p> <p><Friendly name> - The friendly name associated with the certificate. This is often the same as the common name of the subject.</p> <p>e=<email address> - The email address embedded in the subject of a digital certificate. (Note: Not all certificates contain an email address.) The "e=" prefix is optional.</p> <p>@<file name> - Specifies a text file which contains a list of certificates, one on each line.</p> <p>#<file name> - Specifies a PKCS#7 or PKCS#12 file that contains the certificates themselves.</p> <p>----- No default value.</p> | <pre>pkzipc -extract -verifysigner="Thomas Jones, Jr." save.zip *.doc pkzipc -extract -verifysigner="cn=Thomas Jones, Jr." save.zip *.doc pkzipc -extract - verifysigner=e=john.publ ic@ nowhere.com save.zip *.doc pkzipc -extract - verifysigner=john.public @ nowhere.com save.zip *.doc pkzipc -extract - verifysigner= f>(&(userCertificate=*) (ou=Sales)) save.zip *.doc pkzipc -extract - verifysigner= "f>(&(userCertificate=*) (ou=Sales With A Space))" save.zip *.doc pkzipc -extract - verifysigner=@recipients .txt save.zip *.doc pkzipc -extract - verifysigner=#recipients .p7b save.zip *.doc pkzipc -extract - verifysigner=#recipients .p12 save.zip *.doc</pre> | <p>extract</p> |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|--|------------|
| <p>version</p> <p>Gives information about the version of the release. Displays complete version information; also returns to the shell particular version numbers specified by sub-options.</p> | <p>major - Returns the major release number. For example, if the version number is 12.10.1054, the value returned is 12.</p> <p>minor - Returns the minor number of the release. For example, if the version number is 12.10.1054, the value returned is 10.</p> <p><i>step</i> - Returns the step, or patch value (minus 1000 if ≥ 1000). For example, if the program version is PKZIPC 12.10.1054, the value returned is 54.</p> <p>product - Returns the build number of the product. For example, if the product version is SecureZIP 12.10.0003, the value returned is 3.</p> <p>-----</p> <p>Default = major</p> | <p>The command line:</p> <p>pkzipc -version</p> <p>outputs two lines like the following after the usual header information:</p> <p>Program File Version (pkzipc): 12.30.1062</p> <p>Product Version: 12.30.0004</p> <p>The minor sub-option outputs just the minor version number, for example, 10:</p> <p>pkzipc -version=minor</p> | standalone |
| <p>view</p> <p>Displays information about the files in an archive—for example, the compressed size of a file</p> <p>Configurable</p> | <p>brief - present information in the most compact manner.</p> <p>detail - present information in the most detailed manner</p> <p>normal - present information in the normal manner.</p> <p>-----</p> <p>Default = normal</p> | <p>pkzipc -view save.zip</p> | standalone |

| Name/Description | Value(s) | Example usage | Used with |
|--|--|---|---|
| <p>warning</p> <p>Pauses after every specified warning and prompts whether to continue. If no warning is specified, pauses after every warning.</p> <p>Configurable</p> | <p><warning number> - One or more warning numbers, separated by commas. To override a warning number configured for the option (and thus <i>not</i> pause and prompt on that warning), precede the number with a hyphen</p> <p>-----</p> <p>No default value.</p> | <pre>pkzipc -extract -warning=43 save.zip * pkzipc -extract -warning save.zip * pkzipc -extract -warning=-43 save.zip *</pre> | <p>add, extract, test, view</p> |
| <p>wipe</p> <p>Overwrites PKZIP temporary files and files deleted by PKZIP to prevent recovery of their data</p> <p>Configurable</p> <p>Note: This option is deprecated. Use the functionally identical shred option instead.</p> | <p>none - turns off shredding: files are not overwritten</p> <p>random - Overwrites files once with random data</p> <p>dod5220 - Overwrites files three times, to the DOD 5220.22-M specification</p> <p>nsa - Overwrites files seven times, to the NSA standard</p> <p>-----</p> <p>Default = none</p> <p>Value if used on command line without a sub-option = random</p> | <pre>pkzipc -add -move -wipe=nsa myfiles.zip *</pre> | <p>add</p> |

| Name/Description | Value(s) | Example usage | Used with |
|---|--|---|---|
| <p>zipdate</p> <p>Note: This option is deprecated. Use the functionally identical option archivedate instead.</p> <p>Set the file modification date of the archive file.</p> <p>Configurable</p> | <p>newest - set to the date of the newest file within the archive file.</p> <p>oldest - set to the date of the oldest file in the archive file.</p> <p>retain - retain the original date of the archive file (the date when the file was created).</p> <p>none - disable the file date in the configuration file and set the archive date as the last modification date.</p> <p>-----</p> <p>Default = none</p> | <pre>pkzipc -add=update -zipdate=retain save.zip *.txt</pre> | <p>add, delete, fix, header, comment, sfx</p> |
| <p>zoneidentifier</p> <p>Specifies that any Zone Identifier information for the archive should be copied to all files extracted from the archive.</p> <p>Configurable (Windows)</p> | <p>enable – Indicates the Zone Identifier should be copied from the archive.</p> <p>disable – Indicates the Zone Identifier should not be copied from the archive.</p> <p>-----</p> <p>Default = enable</p> | <pre>pkzipc -extract - zoneidentifier archiveFromInternet.zip</pre> | <p>Extract</p> |

B

Error and Warning Messages

This appendix contains reference information on all error and warning messages that can occur in PKZIP. An error usually causes the canceling of the task you are performing such as compressing a file. A warning usually indicates that something is wrong, but it is not severe enough to cancel an entire task. It might also be a reminder or query prompt. PKZIP will also return any error codes to the shell. If there were no warnings or errors, 0 is returned.

Error Messages

When an error occurs, PKZIP displays an error message. The following is a description of each error message.

| <i>Error</i> | <i>Potential Cause(s)</i> |
|--|---|
| (E2) Ambiguous option or command specified - XXX. | If you abbreviate an option on your command line, make sure that you are supplying enough characters in the option to delineate it from similarly spelled options. If, for example, you only specify -pr on your command line, PKZIP will generate the (E2) error because it cannot determine whether you are specifying the print or preview option. |
| (E3) Ambiguous sub-option specified - XXX. | If you abbreviate a sub-option on your command line, make sure that you are supplying enough characters in the sub-option to delineate it from similarly spelled sub-options. If, for example, you only specify -sort=na on your command line, PKZIP will generate the (E3) error because it cannot determine whether you are specifying the name or natural sub-option. |
| (E4) Unknown or illegal option - XXX. | The option you specified on the command line is invalid. It does not match any known options. Verify that you typed the option correctly. Check the spelling. |
| (E5) Unknown or illegal sub-option - XXX. | The sub-option you specified on the command line is invalid. It does not match any known sub-options. Verify that you typed the sub-option on your command line correctly. Verify that you are not using an illegal sub-option (-add -sort=crc). Check the spelling. |
| (E6) No .ZIP file specified. | There was not a .ZIP file specified on the command line. PKZIP does not accept wildcards for .ZIP file name when adding files to a .ZIP archive. |
| (E7) Can't create: XXX. | PKZIP could not create a .ZIP file when fixing. PKZIP could not create a volume label on a spanned archive. PKZIP could not create a temporary file for a spanned archive. Verify that you have write access to the drive or diskette on which you are creating these files. |

| Error | Potential Cause(s) |
|---|--|
| (E8) Nothing to do! | You did not do something that is required for a particular task. For example, PKZIP could not find the file you are trying to open or access. You might have specified to update a pattern such as *.txt and PKZIP did not find any files that matched or that needed updating. |
| (E9) No file(s) were processed | PKZIP cannot find the file you are trying to access. For example, you might be trying to extract files from a .ZIP archive that do not exist in that archive. Verify that the file(s) you specify on the command line exactly match the file(s) in the .ZIP file. If, for example, the file in the archive is stored with path information, and you attempt to extract it but specify only the file name, you will get the (E9) error. |
| (E10) No files specified for deletion. | There are no files or file patterns specified for deletion on the command line. In lieu of a specified file or file pattern, PKZIP will not assume that the user wishes to specify all (*) files. |
| (E11) Disk full, file: XXX. | The hard disk or floppy disk you are writing to is full. This error occurs when PKZIP attempts to write a .ZIP file, or extract a file contained in a .ZIP file to a hard or floppy disk that is full. Free up sufficient disk space and try again. |
| (E12) Can't find file: XXX. | PKZIP cannot find the .ZIP file you specified. This error will only occur when you use commands/options/sub-options that work with existing .ZIP files. Verify that the file is specified correctly. If you are adding files to an archive, verify that you place the .ZIP file name before specifying files to be added on the command line. If the .ZIP file is not in the same directory where you typed the command, make sure to include path information. <i>(e.g., pkzipc -add=freshen /temp/test.zip *.txt)</i> |
| (E13) Can't open .ZIP file: XXX. | The named .ZIP file is read-only or locked by another application and cannot be modified. This may also occur on a Network drive if you do not have sufficient access rights to the file to allow you to modify it. |
| (E14) Can't create archive: XXX. | PKZIP is not able to create the archive file. Verify that the destination directory is not full, and the archive file does not already exist. If you are creating the file on a network drive, confirm that you have the appropriate rights to the network file system. |
| (E15) Renaming temporary .ZIP file, saved as: XXX. | PKZIP could not rename the temporary file to the specified .ZIP file name. Verify that the destination drive is not full. If you are updating a non-spanned .ZIP file on removable media (floppy diskette) and the updated archive exceeds the size available on the removable media, you will receive the (E15) error. You will need to recreate the archive for spanning. Keep in mind that you cannot update a spanned archive. If you are creating the file on a network drive, confirm that you have the appropriate rights to the network file system. |

| <i>Error</i> | <i>Potential Cause(s)</i> |
|---|---|
| (E16) Can't open for write access, file: XXX. | PKZIP is unable to write to the specified file or device. Verify that you have write access to the file or that your printer is configured correctly. Additionally if you are using the PKSFXS.DAT file, verify that you have PKSFXSDATA environment variable configured correctly. |
| (E17) Error encrypting file data. | PKZIP encountered a problem with the compressed data that it was trying to encrypt. For example, the disk on which the compressed data was located was bad or corrupt. |
| (E18) Can't open list file: XXX. | The named list file could not be found. It does not exist, was spelled incorrectly, is not located in the specified directory, or cannot be accessed because the user does not have the appropriate rights to the file. |
| (E19) Aborted file extract. | Extraction process was terminated by the user while changing disks during a disk spanning operation. The error also occurs on attempting to extract a bad TAR archive. |
| (E20) Aborted file compression. | Compression process was terminated by the user while changing disks during a disk spanning operation. |
| (E21) Can't modify a spanned or split .ZIP file | Spanned or split .ZIP files cannot be modified. The archive will need to be recreated. |
| (E22) Cannot format removable media. | The media cannot be formatted. The media may be write-protected. |
| (E23) Suboption is too long | The option is too long; that is, longer than 270 characters. See if you can abbreviate the name of the option or its sub-option to make it shorter. |
| (E24) Insufficient disk space for ZIP comment. | There is not enough space on the system or media to write the ZIP comment. |
| (E25) Insufficient disk space for updated file. | Insufficient disk space for the new archive. If you are adding files to an archive on a removable media, the media may not be large enough to write the modified file (too large). |
| (E26) Device not ready: XXX. | The removable media device is not ready. The disk may not be in the drive properly. |
| (E27) 2.04g compatibility cannot be used with the option - XXX | Option 204 , which creates an archive compatible with PKZIP for DOS v. 2.04g, was used with another option that is not supported for that version of PKZIP |
| (E28) Share violation, file is in use by another process: XXX | The archive XXX is not in a format which PKZIP can understand or contains errors. The errors could be caused by many things but usually mean the archive is corrupted. |
| (E29) Missing sub-option -XXX | Many options require a sub-option to work. In this case, a required sub-option is missing. Add the appropriate sub-option to your command. |
| (E34) Invalid archive format: <archive name> | The file is not in a format currently supported by PKZIP, or you attempted to use SecureZIP Partner to extract a non-ZIP archive. |

| Error | Potential Cause(s) |
|--|---|
| (E58) Invalid archive - method not supported. | The archive uses a compression method that currently is not supported. |
| (E65) Could not encode archive file: XXX. | The file could not be encoded. |
| (E71) Can't open PKCS#7 file: XXX. | PKZIP cannot open the PKCS#7 because the file does not exist, user cannot read the file, or file is not a valid PKCS#7. |
| (E72) PKZIP wanted user input, but silent=input or silent=all was specified | If PKZIP needs user input—for example, to say whether files should be overwritten—but -silent=input or -silent=all is specified on the command line to hide PKZIP messages and prompts, PKZIP halts processing and issues this error. |
| (E73) Warning configured as an error | The warning immediately preceding this error message has been specified (with the error option) to be treated as a fatal error. |
| (E75) Incorrect passphrase or certificate not found, unable to open archive: <archive name> | The archive contains encrypted file names that PKZIP cannot decrypt. If the archive is passphrase-protected, you must include the passphrase option with the <i>extract</i> command in the command line. |
| (E76) Cannot open alternate config file: <file name> | The <i>altconfig</i> option was used, but the specified file could not be opened. |
| (E77) Archive can only support one file inside! | You tried to add more than one file to an archive of a type that cannot contain multiple files. For example, a GZIP archive can contain no more than one file. If you try to create a GZIP archive to contain three files, PKZIP displays this error and does not create the archive. |
| (E78) Unable to FTP archive file: <file name> | PKZIP could not transfer the specified file. |
| (E79) Unable to E-mail archive file: XXX | A problem, perhaps with the network or the mail server, prevented PKZIP from emailing the specified file. |
| (E80) Unable to run anti-virus | PKZIP was unable to run the anti-virus scanning program. The anti-virus program did not respond to the command line used to launch the program. |
| (E81) Possible virus detected | The anti-virus program returned a non-zero value after doing a scan. Most anti-virus programs use this return to indicate the possible presence of a virus. |
| (E82) Too many recipients, recipient count limited to 3275 certificates | You specified too many recipients for encryption. The ZIP file format limits the number of recipients to 3275. |
| (E83) Specified SFX cannot extract archive created with the option - XXXX | You tried to create an SFX that is not able to handle a feature turned on by the option XXXX. For example, you tried to create a strongly encrypted DOS SFX, or an SFX that uses FNE. |
| (E84) Fatal policy error - <i>nnnnn</i> , contact your system administrator | <p>A critical problem has occurred with a policy file or policy certificate. The number is a policy error code to help your administrator resolve the problem.</p> <p>On this error, PKZIP goes into read-only mode. In read-only mode, PKZIP will still extract files from archives but will not add files to a new or existing archive and disables the related controls.</p> |

| <i>Error</i> | <i>Potential Cause(s)</i> |
|--|--|
| (E85) Unable to encrypt, no certificates passed -strict check | The strict option was used, and no recipient certificates passed strict checking, so no certificate was available to use to encrypt |
| (E86) Archive is not signed by a specified verification certificate | The verifysigner option was used to specify one or more certificates, but the archive to be extracted was not signed using any of these certificates |
| (E87) Certificate not found: XXX | The verifysigner option was used to specify one or more certificates, but not all of the certificates could be found, either locally or in a specified LDAP directory |
| (E88) Multiple certificates found: XXX | The verifysigner option was used to specify one or more certificates, and multiple certificates were found—probably in an LDAP directory—that matched the criterion XXX |
| (E89) Policy requires the ZIP archive to be encrypted | A policy requires encryption but no passphrase or recipient was specified |
| (E90) Policy requires the ZIP archive and/or files to be signed | A policy requires the archive to be signed but no signing certificate was specified or none is available |
| (E91) Policy prohibits creation of non-ZIP archives | Only ZIP archives can be created when a policy requires encryption or signing |
| (E92) Timeout error on file: XXX | The timeout period elapsed while PKZIP waited for a response from another process before reading or writing more of the specified archive file on a socket or block device |
| (E94) Can't modify a timestamped .ZIP: <name> | You cannot change an archive (or files in an archive) that has been timestamped. |
| (E100) Insufficient memory | Insufficient memory is available to process the archive. Try making more memory available to PKZIP. If this does not rectify the problem, then the archive may be corrupted. The -fix command may correct the problem. If you receive this message when you try to create a new archive, possibly you are attempting to compress too many files. Reduce the number of files and try again. If you are using a LIST file in your PKZIP command, the LIST file may be too large. See "Compressing Files with a List File" in Chapter 0 |
| (E150) Error reading .ZIP file. | PKZIP cannot read the .ZIP file or is unable to read the central directory record. The file might be located on a corrupt disk or part of a disk. This includes floppy disks. |
| (E155) Too many files in XXX. | PKZIP cannot add or extract files in excess of the limit of 16,383 with the 204 option enabled. Reduce the number of files you are trying to process. |
| (E156) File is now too big for valid zip data. | The .ZIP archive is too large and PKZIP is unable to locate the central end record in the .ZIP file. The file is not a valid .ZIP archive or has been corrupted. The fix command may repair the .ZIP file. |

| <i>Error</i> | <i>Potential Cause(s)</i> |
|---|--|
| (E157) This archive requires a product compliant with ZIP APPNOTE version XX.X | The archive requires a more recent version of PKZIP, or other archiving program, that supports the version of the ZIP file format described in the specified APPNOTE ("application note"). The APPNOTE is a document that is available on the PKWARE Web site. |
| (E158) Errors encountered reading archive | PKZIP was unable to read the archive. |
| (E200) FIPS 140 mode is enabled, but archive is not encrypted with a FIPS-approved algorithm | With the fipsmode option, SecureZIP cannot work with (test, extract, add to, update, view, or open) an archive with encrypted file names that is encrypted using a non-FIPS-approved algorithm. |
| (E201) FIPS 140 mode is enabled, but encryption requested is not a FIPS-approved algorithm | With the fipsmode option, SecureZIP cannot use the specified algorithm to encrypt. Use listcryptalgorithms with the fipsmode option on to see FIPS-approved algorithms available. |
| (E202) FIPS 140 mode is enabled, but signature hash requested is not a FIPS-approved algorithm | With the fipsmode option, SecureZIP cannot use the specified hashing algorithm. Use listhashalgorithms with the fipsmode option on to see FIPS-approved algorithms available. |
| (E203) FIPS 140 mode failed to initialize | (UNIX only) FIPS 140 mode could not be initialized for the fipsmode option. The pkzipc binary may be corrupted. |
| (E253) This program is not licensed for use on Windows Server platforms. Please contact PKWARE to obtain an appropriate server product for this machine. | <i>PKZIP for Windows Command Line</i> is intended for single-user desktop use and cannot be run on server platforms such as Windows 2003 Server. |
| (E254) Your evaluation period for PKZIP has expired. Please register to continue using this product. | This copy of PKZIP is an evaluation version. If you have purchased PKZIP and have the serial number, enter it when prompted. |
| (E255) User pressed ctrl-c or control-break. | This error occurs when you press CTRL+BREAK or CTRL+C in the middle of a PKZIP operation. |

Warning Messages

Sometimes a condition occurs that might cause a task to pause temporarily. This could be something that prevents part of a task from happening, or simply a message or reminder. For several of these conditions, PKZIP displays a warning message. When a warning occurs, PKZIP returns a value of 1 to the shell.

The following is a description of each warning message:

| <i>PKZIP Warning</i> | <i>Potential Cause(s)</i> |
|---|--|
| (W1) Can't create: XXX. | PKZIP could not create volume label, file, or directory. Verify that you have appropriate access rights to the file or directory. |
| (W2) Illegal path or drive specified: XXX. | The file being extracted has an invalid name or path. Verify that you have entered the correct path in your command line and that the file does not contain any inappropriate characters such as a colon or leading slash. |

| PKZIP Warning | Potential Cause(s) |
|--|--|
| (W3) Warning! This file requires a product compliant with ZIP APPNOTE version XX.X | The file requires a more recent version of PKZIP, or other archiving program, that supports the version of the ZIP file format described in the specified APPNOTE ("application note"). The APPNOTE is a document that is available on the PKWARE Web site. |
| (W4) File fails CRC check. | It is likely that the file PKZIP is trying to extract is corrupt, and was not extracted correctly. For more information, see the CRC section in Appendix D. |
| (W7) file: XXX already exists. Overwrite (<Y>es/<N>o/<A>ll/ne<V>er/<R>ename/<Esc>)? | The file(s) you are trying to extract already exists in the location to which you are extracting. By default, PKZIP prompts you before overwriting a file. |
| (W8) Could not open file: XXX. | You may not have the proper permissions to access the file or the file may have been locked by another program while PKZIP was trying to access it. If the file is located on a network file system, consult your System Administrator to verify your access rights. |
| (W9) Could not delete file: XXX. | You do not have the proper permissions to access and delete the file, or another application has the file open. This warning only occurs when the move option is used on the command line. |
| (W12) Unexpected end of compressed data. | Corrupt data caused PKZIP to abort the extraction before it could finish. |
| (W13) Skipping encrypted file: XXX. | PKZIP encountered a file that has been passphrase protected. You need the passphrase to access this file. |
| (W18) Unknown compression method for file: XXX. | An unfamiliar compression method has been used with the current .ZIP file. |
| (W19) Could not clear archive attribute on file: XXX. | PKZIP could not clear the archive attribute on a file. The file will be compressed but the archive bit cannot be cleared. This warning usually occurs when the add=incremental option is used on the command line. |
| (W20) Incorrect passphrase for file: XXX. | Verify that you entered the correct passphrase for the file. When a file is passphrase protected, you can only access its contents with the correct passphrase. Note: Passphrases are case sensitive. |
| (W21) Invalid temporary file directory: <dir> | PKZIP creates a temporary file for the file(s) being compressed when updating a .ZIP file. PKZIP was unable to create the temporary .ZIP file in the specified location and so used the default temp directory for your system. |
| (W22) Authenticity Verification Failed! | The Authenticity Verification (AV) information contained in the .ZIP file is corrupt. Failure of AV indicates a file that has been tampered with or damaged. If the file has failed the AV check, the contents are suspect. |

| PKZIP Warning | Potential Cause(s) |
|---|--|
| (W23) Authenticity Verification Failed! | The stored Authenticity Verification (AV) checksum value did not match the calculated checksum value. The .ZIP file has been tampered with or is perhaps corrupt. |
| (W26) directory: XXX already exists. Overwrite (<Y>es/<N>o/<A>ll/ne<V>er/<Esc>)? | Assuming the overwrite option is set to prompt, this warning appears when PKZIP attempts to extract a directory over an existing directory with the same name. Answering Y at this prompt will update any extended attributes (EAs) stored in the .ZIP file. |
| (W29) Can't rename temporary file. Saved as XXX. | PKZIP cannot rename the temporary archive created when updating an archive. The archive was saved under the specified name. |
| (W36) Empty passphrase, files will not be passphrase protected. | When trying to passphrase protect your file, you entered a passphrase containing no letters or numbers. |
| (W37) Can't sign file. | This warning appears when PKZIP fails to sign a file using the specified digital certificate. Common reasons are incorrect passphrase for the certificate (not all certificates have passphrases), no private key (certificate needs to have a private key). |
| (W38) Can't sign central directory. | PKZIP failed to sign the central directory. Common reasons are that an incorrect passphrase was supplied to access the certificate (not all certificates have passphrases) or the certificate lacks a private key (needed to apply a digital signature). |
| (W39) Signature is invalid. | Someone or something has changed the archive since it was digitally signed. For example, the archive may be corrupt. |
| (W40) Certificate not trusted. | The certificate is currently not to be trusted. |
| (W41) Certificate expired. | The certificate has expired. This does not necessarily mean that the certificate or signatures applied with it are not to be trusted. They may simply be old. |
| (W42) Certificate was revoked. | The issuer has revoked the certificate. |
| (W43) Certificate not found: XXX. | PKZIP was unable to find a certificate of that name on the system. |
| (W45) Bad data in compressed stream. | Something was wrong in the stream of compressed data. The ZIP file is corrupt. |
| (W46) Encryption algorithm is not available. Using: XXX. | PKZIP cannot use the specified algorithm on this system. Use the <i>ListCryptAlgorithms</i> command to view a list of the encryption algorithms that PKZIP can use |

| PKZIP Warning | Potential Cause(s) |
|--|---|
| (W47) No recipients specified. Recipients will not be used. | You specified the <i>recipient</i> option, but did not include any actual recipients (or specified bogus recipients). When this occurs, SecureZIP will not strongly encrypt files for recipients. If you did not tell it to use passphrases; that is, you did not use the -passphrase option, it will not encrypt files at all. In addition, if you specify <i>passphrase</i> and did not also specify <i>cryptalgorithm</i> , you will not get strong encryption. You will, however, get traditional encryption. |
| (W48) Invalid item name | The name of an item (file) in the archive is invalid. Possible reasons are: The file has the same name as another file in the same folder; the path name of the archive item contains a file or folder name that exceeds the maximum number of characters allowed (254 for Windows, 255 for UNIX); the name contains characters that may not be used in file names on your operating system (the characters : * ? \ " < > " may not be used in file names on Windows). |
| (W52) Certificate verification failed! | Something is wrong with the certificate. |
| (W53) Unknown exception caught: Exception code: XXX | An internal error occurred. Please contact PKWARE Technical Support with the exact command you used and the error code. |
| (W54) Option 'XXX' is not licensed for use in your copy of PKZIP | Your license key does not allow you to use that option. You must purchase an appropriate license key from PKWARE to use it. |
| (W55) Command 'XXX' is not licensed for use in your copy of PKZIP | Your license key does not allow you to use that command. You must purchase an appropriate license key from PKWARE to use it. |
| (W56) Recipient not found for file: XXX | The file was encrypted only for recipients, and PKZIP was unable to find a certificate for any of them. Verify that you have access to the private key for one of the recipients. |
| (W57) Incorrect passphrase or recipient not found for file: XXX | Verify that you entered the correct passphrase for the file. When an archived file is passphrase protected, you can only access the file if you have the correct passphrase. Passphrases are case sensitive. If the file is encrypted with a certificate, verify that you have access to the private key for one of the recipients. |
| (W58) Problem reading .ZIP file: <zipfile name> | The .ZIP archive is corrupted. PKZIP can read it, but probably other zipping programs cannot. Use the -fix command to fix the archive so that other programs can read it. |
| (W59) Multiple certificates found | Multiple digital certificates were found that match the same recipient. These certificates may belong to different people. The archive is encrypted using each of the certificates; the owner of any of them can decrypt. |

| PKZIP Warning | Potential Cause(s) |
|---|--|
| (W60) Unable to connect to LDAP server: <server name/address> | PKZIP was unable to access certificates on an LDAP server specified using the ldap option: the server address was bad. |
| (W61) Unable to login to LDAP server: <server name/address> | PKZIP was unable to access certificates on an LDAP server specified using the ldap option: the LDAP login failed. |
| (W62) Central Directory can only be encrypted with strong encryption. Central Directory will not be encrypted. | The <i>cd</i> option was used, which requires strong encryption, but one or both of the following were neither explicitly specified nor configured for use by default: encryption method (<i>passphrase</i> , <i>recipient</i> options), encryption algorithm (<i>cryptalgorithm</i> option). |
| (W63) You must specify <i>-passphrase</i> or <i>-recipient</i> to encrypt files! | You specified <i>-cryptalgorithm</i> or <i>-cd=encrypt</i> but did not specify either the <i>recipient</i> or <i>passphrase</i> option. Files are not encrypted unless one of these options is used. |
| (W68) Must specify <i>MailTo</i> , <i>MailFrom</i> and <i>MailServer</i> to email the archive. | You tried to email an archive without specifying all three options <i>MailTo</i> , <i>MailFrom</i> and <i>MailServer</i> . Values for all three must be specified on the command line or configured for use by default. |
| (W69) Skipping FTP file transfer because of encryption warning XXX. | PKZIP encountered a problem encrypting an archive that you directed to send by FTP, so PKZIP did not send the archive. This warning occurs if, for example, PKZIP can encrypt for only some but not all recipients, or if no passphrase is supplied to use for passphrase encryption. |
| (W70) Skipping mail file transfer because of encryption warning XXX. | PKZIP encountered a problem encrypting an archive that you directed to send by email, so PKZIP did not send the archive. This warning occurs if, for example, PKZIP can encrypt for only some but not all recipients, or if no passphrase is supplied to use for passphrase encryption. |
| (W71) Could not attach unzip instructions to the email message | PKZIP failed to attach instructions on how to unzip, as specified by the MailOptions option |
| (W72) Could not find the unzip instructions | PKZIP could not find the instructions on how to unzip |
| (W73) Some of the encryption recipients do not have email addresses | PKZIP was told to encrypt for recipients but could not find email addresses for some of the recipients |
| (W74) PKZIP is unable to access the default user's private key. | PKZIP is unable to access the private key of the default user. The logon passphrase needed to access the certificate that contains the key may have been reset or changed by an administrator. To fix this warning, the user must change his passphrase from his own computer, rather than let an administrator change it from another system. |
| (W75) Unable to resolve link: XXXX | While updating an archive, PKZIP could not find the original file or the new file |

| PKZIP Warning | Potential Cause(s) |
|--|--|
| (W76) <certificate> does not pass the strict certificate checks, and will not be used. | The certificate did not pass the strict checking applied by the strict option, used in a command line that updates an archive. The certificate will not be used for the intended signing or encryption. |
| (W78) Policy error - <i>nnnnn</i> , contact your system administrator | A noncritical problem has occurred with a policy file or policy certificate. Encryption may be disabled. The number is a policy error code to help your administrator resolve the problem. |
| (W79) Certificate chain is not time nested | A certificate lists a start date or end date that falls outside the period during which an issuing certificate in its trust chain is nominally valid. This may not be cause for concern, but it might indicate a problem. |
| (W80) Passphrase encryption not available in SecureZIP Partner | You tried to encrypt using a passphrase. SecureZIP Partner automatically applies certificate-based encryption to every new or updated archive for sponsor recipients but does not do any other encryption. |
| (W85) Warning! Error shredding file: XXX | You used the <i>shred</i> option, but PKZIP was unable to overwrite the file. For example, PKZIP can delete a file on a network drive but cannot overwrite its data on disk. |
| (W87) Skipping file that is not encrypted with a FIPS-approved algorithm | The same algorithm that was used to encrypt must be used to decrypt. With the <i>fipsmode</i> option, SecureZIP uses only algorithms that are FIPS-validated for your operating system and skips—does not decrypt—any file that was encrypted using some other algorithm. On Windows 2000, for example, files encrypted with AES algorithms are skipped. |
| (W88) Warning! Signature cannot be verified, because it does not use a FIPS-approved algorithm | The file is signed, but the <i>fipsmode</i> option was used, and the signature hash algorithm is not FIPS-approved. The SHA-1 algorithm is not approved after 2010. On some versions of Windows, the algorithms SHA-256, SHA-384, and SHA-512 are not approved. |
| (W92) Warning! File: <file name> is in use by another program. You might have problems opening the archived copy if it is currently saving changes. | You are using the OpenFile option with the prompt sub-option. Choose from the options to Add the open file to your archive. |
| (W93) Unable to obtain timestamp: <url> | The URL you specified for your Time Stamping Authority (TSA) is missing, incorrect, unavailable, or you are not connected to the Internet. |
| (W94) Evidence record verification failed | Timestamp hash does not match. |
| (W95) Evidence record has expired | Last timestamp certificate is no longer valid. |
| (W96) Evidence record missing file signature(s) | All timestamped files must be signed with the same hash algorithm as the central directory. |

C

Frequently Asked Questions

This section lists some commonly asked questions about PKZIP and related subjects. We hope you will find this information helpful.

Why do I get the message "SYS1041: The name specified is not recognized as an internal or external command, operable program or batch file." or "Bad command or file name" or "XXXX: not found"?

These messages tell you that your operating system cannot find the program to which you are referring. This occurs because you are either not spelling the name of the program correctly, or you did not put a space between the program name and its options, or the program has not been properly installed. If you are trying to run PKZIP and you get this error, it may be because pkzipc.exe is not in your search path.

Why didn't the files I zipped get any smaller?

On occasion, you may find that the files you add to a .ZIP file do not compress. These files are "stored". This occurs when a file is either already compressed or encrypted. You will often find that files distributed with commercial applications are already compressed.

I zipped up a bunch of files but now I have LESS disk space?

When PKZIP compresses files, it makes a copy of the original file. The original file(s) still exist. If you wish to recover space that was taken up by the original file(s), you must either delete them yourself, or instruct PKZIP to delete the file(s) with the *move* option.

What is the difference between add=freshen and add=update?

The *freshen* and *update* sub-options are very similar. This may be confusing at first, but the difference between them is easy to understand.

Freshen tells PKZIP to archive any files which match those already in the .ZIP file. These files are re-compressed only if they are newer than the files already in the .ZIP file. Each file is evaluated individually.

Update archives all files, with one distinction. If the *update* option is not used, all files specified are compressed and added to the .ZIP file, even if they already exist in the .ZIP file. By using the *update* sub-option, you instruct PKZIP to compare what is already in the .ZIP file against what it was asked to compress. If a file is already present in the .ZIP file as well as the source directory, PKZIP compresses a file only if it is newer than the copy of the file within the .ZIP file. If a file in the source directory is not already present in the target .ZIP file, PKZIP adds it to the .ZIP file.

Is PKZIP compression "lossy" or "lossless"?

PKZIP uses a "lossless" compression scheme. This means that 100% of the original data is preserved and re-created. There is no difference between the data that you put in and the data that you get back out.

There are other compression methods that are known as "lossy." The idea behind these compression methods is that if you throw away some of the data, it becomes

less complex and therefore can be compressed more. This type of compression is only useful for data that need not be precise. This applies to some applications that use pictures and sound.

How do I include subdirectory information in my .ZIP file?

In order to include subdirectory information in your .ZIP file, you must recurse the subdirectories and preserve path names. This is done with the *directories* option. For example:

```
pkzipc -add -directories test.zip *
```

In this example, the current directory as well as all subdirectories and files contained therein are archived in a file called test.zip.

When a .ZIP file is created with paths stored, these paths are visible in a view of the file (*view*).

To re-create these subdirectories, or to place files into their original subdirectories, the *directories* option must be used with the *extract* command.

I zipped up some subdirectories, but I cannot get them to come back.

Did you remember to use the *directories* option when you originally created the .ZIP file? Did you use the *directories* option when you extracted the contents of your .ZIP file? To verify that there are paths in the .ZIP file, do a view of the file:

```
pkzipc -view test.zip
```

If you do not see paths as part of the file names within the .ZIP file, then paths are not stored and therefore cannot be recovered. If you do see paths make sure that you are using the *directories* option when you extract the files. For example:

```
pkzipc -extract -directories test.zip
```

How do I unzip a single file that is in a subdirectory in the .ZIP file?

Type *pkzipc -extract* with the name of the .ZIP file and the name of the particular file you want. With a .ZIP file that contains paths, the procedure is the same.

Assume you are working with a file called test.zip that contains the following files:

```
file1.txt  
temp/file2.txt  
temp/tut/file3.txt
```

To extract only "file3.txt" from this .ZIP file, you must specify the complete name and path.

```
pkzipc -extract test.zip temp/tut/file3.txt
```

If you wanted to extract it with its subdirectory, simply include the *directories* option on the command line.

How do I unzip a directory without also extracting its subdirectories?

Using the test.zip file we discussed in the previous question, we could extract the entire contents of the temp subdirectory easily:

```
pkzipc -extract -directories test.zip "temp/*"
```

If we did it as shown above we would not only extract all the files in the "temp" subdirectory, but also the "tut" subdirectory below it and any files it contains.

To extract only the "temp" subdirectory but not its subdirectories, we must exclude the subdirectories we do not wish to extract:

```
pkzipc -extract -directories test.zip "temp/*" -  
exclude="temp/tut/*"
```

If the "temp" subdirectory had multiple subdirectories nested in it, you would need to exclude each one individually on the command line.

I forgot my passphrase; what do I do?

- Try to remember the passphrase.
- Try passphrases that are "close" to what you think it was.
- Try mixed upper and lower case versions of your passphrase.

Do not forget or lose your passphrases! PKWARE has no special means for "getting around" the encryption and may not be able to assist in the recovery of an encrypted file. To help avoid the loss of data, you may wish to keep a written copy of your passphrase(s) in a secure place.

What does "Unknown Compression Method" mean?

There are many different methods of compression. In the history of PKZIP alone, there have been seven different methods to date. The .ZIP file format was designed so that additional methods of compression can be added as they are developed. Therefore, the .ZIP file format will never need to be abandoned. This means that the .ZIP file in question was created or updated by a newer version of PKZIP than is being used to extract the data. You must use a newer version of PKZIP to extract these files.

How can I make PKZIP run faster?

PKZIP defaults to a compression method that is average in both compression amount and speed. If you want to get the most speed out of PKZIP, try the following:

- Specify a faster compression method with a level sub-option (for example, `-level=0`). See "Setting the Compression Level" in Chapter 0.
- Compression speeds are highly dependent on the location of files being added, as well as the temporary file PKZIP creates when performing certain compression operations. If these files are located on a network drive, you may want to move them to a local drive before running PKZIP. Be aware of the effects file location can have on PKZIP's speed.

How many files can be in a .ZIP file?

There is no limit to the number of files you can add to a .ZIP file. However, if you use the `204` option for PKZIP 204g compatibility, your .ZIP file may contain no more than 16,383 file entries.

Can I send a .ZIP file to a different type of computer?

As of the publication of this manual, PKWARE supports PKZIP on MS-DOS, Windows (98, NT, Me, 2000, XP, Vista, 7), OpenVMS, HP-UX, IBM AIX, Linux, Sun Solaris, MVS/ESA, OS/390, z/OS, VSE, and OS/400 platforms. PKWARE intends to support additional platforms and will announce this support as it becomes available.

D

How PKZIP Works

This Appendix provides a description of how PKZIP actually does its job. It is not necessary for you to know or understand the information presented here, any more than you need to know how your carburetor works to drive a car. It is presented to help you feel more knowledgeable about the software.

Two Processes

PKZIP performs two functions: compression and archiving. Although the two ideas may seem related, they are actually completely separate.

- **Compression** is the process of representing a given piece of information with a smaller piece of information.
- **Archiving** is the process of combining many files into a single unit, usually along with vital information about each file.

Compression

The actual process used by PKZIP for its compression is too complex to explain in detail. Instead, some of the general principles behind information theory and compression are explained.

To understand data compression, you need to understand two ideas: Information Content and Binary Coding.

Information Content

Everything in your computer, everything you ever read, is "information". The more complex a message is, the higher the information content. The less complex, the less "random" a message is, the lower the information content.

If a message contains a low amount of information, it should be possible to represent it in a smaller amount of space. Look at this page, for example. How much of the page is white space with no letters (information) on it? If you took away all of the white space this page would be significantly smaller. How many times are the words "the", "information" and "compression" on this page? If you could replace each of these words with something smaller, you would save a significant amount of space.

The more frequently the same group of symbols (in this case, letters) appear, the lower the information content of the message.

The "Field of Information Theory" uses the term *entropy* to describe the "true" information content of a message. Formulas can be used to determine the *entropy* of a message. The idea behind data compression is to derive a new smaller message from a larger original message, while maintaining the *entropy* of the original message.

As a simple example, consider this sentence:

she sells sea shells by the sea shore

This sentence is 37 characters long, including spaces. The spaces cannot be simply thrown away as the meaning of the original message would be lost.

There are obvious patterns to the sentence. The combination 'se' appears three times, 'sh' three times, and 'lls' twice. In fact, the 'se' pairs all have a space in front of them, so these can be ' se'.

she sells sea shells by the sea shore

We can replace each of these patterns with a single character:

#=" se"

\$="sh"

%="lls"

Note that the first replacement string includes a space at the beginning. If we reproduce the sentence with these symbols, it now looks like:

\$e#%#a \$e% by the#a \$ore

The new representation is 24 characters long; this is a saving of 13 characters, or 36%.

Binary Data Representation

All information used, stored, and processed by computers is represented by two values, *zero* and *one*. Everything that you see on your screen, everything stored on disk, is represented by combinations of zero and one.

You can think of it as a sort of Morse Code. In Morse Code there are also only two values, dot and dash. When a computer stores a character, it uses a combination of eight zeros and ones.

Having eight positions in which to store a zero or one gives the computer 256 different possible combinations. You arrive at this number of combinations in this way:

If you have one coin, it can be in either of two positions: Heads(0) or Tails(1)

0 or 1

If you have two coins, there are four possible combinations:

00, 11, 10, 01

If you have three coins, there are eight possible combinations:

000, 001, 010, 011, 100, 101, 110, 111

As you can see, each time you add another coin (binary digit), the number of possible combinations doubles: 2, 4, 8, 16, 32, 64, 128, 256.

The computer uses eight binary digits to get 256 possible values. These values are mapped onto a table called ASCII (American Standard Code for Information Interchange). Each different combination has a particular character that is mapped to it, such as a letter, number or symbol. Each of these positions of 0 or 1 is called a **bit**.

she sells sea shells by the sea shore

The sample message above would be represented by 296 bits (37x8 bits).

If we follow standard ASCII, we have 256 different symbols being represented for our use. The sample sentence we are using only contains alphabetical characters, and

only 11 of them at that. If we only need 11 different values, we could use a lower number of bits per character.

The closest value to 11 using binary combinations is 16 combinations, using 4 bits per character. If we wrote a new table of our own using four bits per character, and used it to represent the message, we would use only 98 bits. This would be half as many bits, a considerable savings.

We can do better!

It is possible to have binary codes of varying length. To do this we must use codes with unique values that are not repeated as the beginning of another code. In this way, we can find the codes in a long stream of zeros and ones.

If the codes were not constructed to have unique beginnings, it would not be possible to find each individual code within a long stream of zeroes and ones.

There are many types of coding techniques that produce codes of varying length, based upon symbol frequency. Some well-known coding schemes are Huffman and Shannon-Fanno. PKZIP uses Huffman encoding. The scheme is too complex to document here fully, however, we will discuss some rudiments of encoding. It is necessary for you to understand the principles described here.

A table of variable length codes for 11 symbols would look like this:

| | |
|------|-------|
| 11 | 1101 |
| 110 | 0100 |
| 101 | 1000 |
| 001 | 01010 |
| 1011 | 00000 |
| 0010 | |

As you can see, the codes are getting longer and longer. Because of this, we will get the best results if we map the shortest code to the most common symbol in the message. If you know Morse code, or have occasion to look at it, you will notice that frequent characters, such as 'e', 't', 's' and so on have shorter codes assigned to them. Morse code tends to be about 25% more efficient because of this than it would have been had the codes been assigned at random.

A useful idea here is to allow a symbol to be not only a character, but also a group of characters.

Using the common patterns found in the first analysis of the message, we can map the following table:

| Occurrences | Symbol | New Code | Bits in Message |
|-------------|---------|----------|-----------------|
| 4 | e | 11 | 8 |
| 4 | (space) | 110 | 12 |
| 3 | 'se' | 001 | 9 |
| 3 | sh | 101 | 9 |
| 2 | lls | 1011 | 8 |
| 2 | a | 0010 | 8 |
| 1 | b | 1101 | 4 |
| 1 | y | 0100 | 4 |
| 1 | t | 1000 | 4 |
| 1 | o | 01010 | 5 |
| 1 | r | 00000 | 5 |

Our new coding scheme can represent the message with only 74 bits. This is a savings of 222 bits from the 296 bits used in the "natural" encoding. This is one quarter of the original message size.

One important factor that would affect a real situation is the table we are using. In order for the data to be re-created from the "compressed" representation, we must include a copy of the table used to encode the data.

This can be a seriously limiting factor. If the data is too complex, or the encoding scheme too inefficient, the table used can be as big as the space saved by the encoding. In the worst cases, an attempt to re-encode the message using a table results in the encoded message plus the table being larger than the original message.

This is why data which uses a low number of symbols and frequently repeated combinations of symbols, such as a text file, compresses well. Complex, highly random data, such as the information representing a program on disk is difficult to encode efficiently, and therefore compresses less.

Speed vs. Size

Searching for these patterns, and determining an efficient way to encode the data, takes a lot of computer power and time. The more time taken to analyze the data the better the compression will be. To get more speed, you must sacrifice some level of compression.

There are other steps and methods used in powerful compression schemes such as those used by PKWARE products. Hopefully this explanation gives you a better understanding of what happens when PKZIP compresses data.

Archiving

Programs usually rely heavily on associated data files, or may actually consist of several related programs. Some programs may require dozens or even hundreds of files.

In the "dawn" of the PC age, people wanted a way to keep all of these associated files in one location. "Library" programs were created to take a number of files and group them together into a single file. This made them easier to find, easier to store, and much easier to send to someone by modem. It makes much more sense to be able to send someone a single "package" instead of many files. If you forget a file, all sorts of problems arise.

These programs were the birth of Archiving. In order for a single file to hold many files, information about each file also had to be stored in the archive. This information could then be used by the archival software to locate a file and pull it out, or to list information about the files contained within an archive.

Compression was first available as a utility that would take a single file and produce its compressed equivalent. People began to group files together with a Library program and then compress the archive file.

The next and obvious step in this process was to combine the two ideas. Compress the files and archive them. This made storage very simple; the compression was no longer a separate step and could be taken for granted as part of the archiving process.

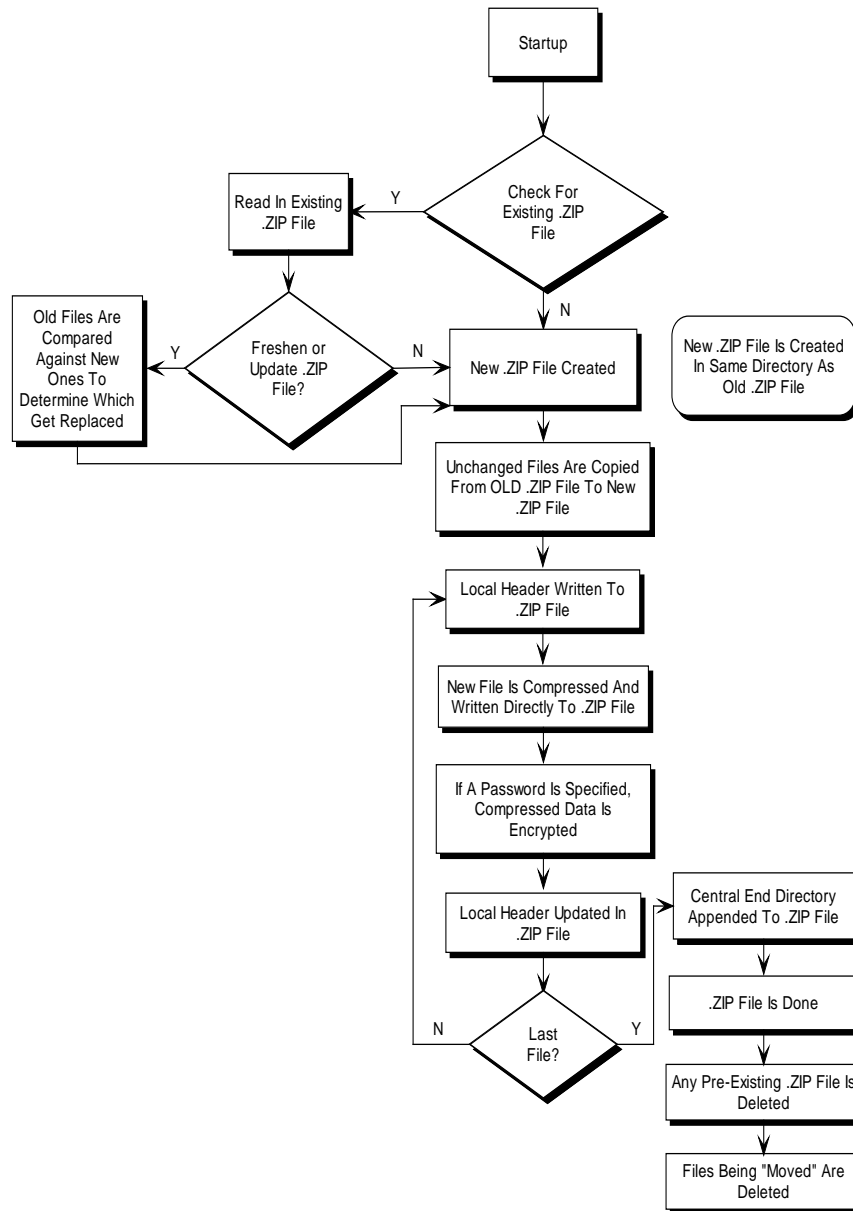
PKZIP is the second generation of these programs. PKZIP can not only compress and archive files, but also stores a great deal of vital information about the files. PKZIP even stores directory structures.

How PKZIP builds a .ZIP File

When you specify a PKZIP command line, PKZIP goes through several steps:

1. Parsing the command line.
2. Reserves the memory it will need to perform the compression, archiving and buffering.
3. Next, PKZIP looks for a .ZIP file with the same name as the one you specified on the command line. If it finds one, PKZIP reads the information on the files that it contains.
4. PKZIP then performs the requested action; it builds a new .ZIP file if none was found.
5. PKZIP reads the information from the command line specifying what files it is supposed to take, what files it should not take, and if there is an ***exclude*** command.

How PKZIP Builds A .ZIP File



- If a @list file is used, PKZIP reads it, then checks for which files exist. If a pattern is specified in the @list file, PKZIP generates a list of the files which match this pattern.
- If directory recursion has been specified with the *recurse* option, PKZIP next looks for any subdirectories. If it locates subdirectories it goes into them and looks for any files matching the files specified on the command line or in the @list file. If PKZIP finds subdirectories in the subdirectories, it repeats the process. It will continue this process until it finds no additional subdirectories.

Now PKZIP has a list in memory of all the files it should take. The files specified for exclusion are now compared against this list, and any that match are removed. If after this step is complete the list in memory is empty, PKZIP finishes with a "Nothing to do!" message.

Now PKZIP reads-in each file, one at a time, and compresses it. When it is finished compressing a file, it adds it to the .ZIP file being created.

6. As PKZIP reads each file, it computes a CRC value for it. This CRC value is stored as part of the information concerning the file.

CRC

This is an acronym for Cyclic Redundancy Check. When a CRC is performed, the data making up a file is passed through an algorithm. The algorithm computes a value based upon the contents resulting in an eight digit hexadecimal number representing the value of the file.

If even a single bit of a file is altered, and the CRC is performed again, the resulting CRC value will be different. By using a CRC value, it can be determined that there is an exact match for a particular file.

PKZIP calculates a CRC value for the original file before it is compressed. This value is then stored with a file in the .ZIP file. When a file is extracted it calculates a CRC value for the extracted data and compares it against the original CRC value. If the data has been damaged or altered, PKZIP can recognize and report this.

1. When PKZIP adds the compressed file to the .ZIP file, it first writes out a "Local Header" about the file. This Header contains useful information about the file, including:
 - The minimum version of PKZIP needed to extract this file.
 - The compression method used on this file.
 - File time.
 - File date.
 - The CRC value.
 - The size of the compressed data.
 - The uncompressed size of the file.
 - The file name.
2. After PKZIP has written all of the files to disk, it appends the "Central Directory" to the end of the .ZIP file. This Directory contains the same information as the Local Header for every file, as well as additional information. Some of this additional information includes:
 - The version of PKZIP that created the file.
 - A comment about each file (if any).
 - File attributes (Hidden, Read Only, System).
 - Extended Attributes (If Specified).

Deleting Files from a .ZIP File

PKZIP deletes files from a .ZIP file in the following manner:

1. PKZIP reads in the names of all the files contained in the .ZIP file.
2. PKZIP compares this list against the files you wish to delete.
3. Whatever files remain are moved into a new .ZIP file.
4. The original .ZIP file is superseded by a newer version of the .ZIP file.

This means that in order to delete files from a .ZIP file, you must have enough disk space to hold both the original .ZIP file and the new .ZIP file that lacks the deleted files.

Adding to an Existing .ZIP File

Adding files to a .ZIP file is the same as creating a .ZIP file, but with one difference. Before PKZIP begins to add files, it first reads in the files that were in the existing .ZIP file. These old files and the new files are then both written out to a new .ZIP file, the old files being superseded by the new .ZIP files. This means that there must be enough free space for the old .ZIP file as well as the new .ZIP file to co-exist.

Index

7

7Zip, 3, See also: archives

A

About This Manual, 1
 anti-virus, 64
 archive attribute, 22
 archiveeach, 49
 archives
 adding to existing, 183
 comments, 52
 convert to self-extracting, 58
 dates, 53
 deleting files, 182
 digital time stamping, 35
 extracting, 9
 fixing corrupt, 95
 freshen, 22, 64
 including open files, 38
 intermediate, 97
 moving files, 8, 56
 naming conventions, 6
 printing, 92
 self-extracting, 57
 sorting files, 55
 span and split, 47
 supported types, 2
 testing, 92
 update, 21
 verify signature, 74
 view contents, 8, 88
 writing to STDOUT, 36
 archiving, 176, 179
 ARJ, 3, See also: archives
 ASCII, 177
 attributes, 50
 extended, 51, 52
 setting internal, 96
 authentication, 72, 77
 avargs, 64
 avscan, 64

B

backup, 22
 incremental, 22
 binary, 176
 binary data representation, 177
 BinHex, 3, See also: archives
 bzip2, 3, 45, See also: archives

C

CAB, 3, See also: archives
 CDR, 3, See also: archives
 Certificate Authority (CA), 78
 certificate stores, 34
 certificates, 77, See digital certificates
 changing defaults, 99
 command line, 3
 changing command/option character, 108
 options, 14
 syntax, 4
 commands, 13, 109
 abbreviating, 14
 changing character, 108
 default values, 21, 100
 difference from options, 13
 values, 15
 Commands/Options
 204, 52, 109
 add, 21, 110
 after, 11, 110, 113
 altconfig, 105, 111
 archivedate, 53, 111, 161
 archiveeach, 49, 112
 archivetype, 46, 112
 ascii, 96
 attributes, 50, 113
 avargs, 64, 113
 avscan, 64, 114
 before, 11, 114
 binary, 96, 114
 bzip2, 45, 114
 cd, 27, 115
 certificate, 31, 115
 comment, 52, 117
 configuration, 99, 118
 console, 72, 118
 crl, 80, 118
 cryptalgorithm, 25, 119
 cryptoptions, 83, 120
 dclimplode, 121
 default, 121
 deflate64, 121
 delete, 121
 directories, 40, 70, 122
 embedded, 65, 122
 encode, 97, 125
 enterlicensekey, 125
 error, 93, 125
 exclude, 13, 126
 extract, 63, 126

fast, 43, 126
fipsmode, 28
fipsmode, 127
fix, 95, 127
freshen, 22, 64
generate list file, 98
hash, 33, 127
header, 53, 127
help, 18, 128
include, 12, 128
jobid, 128
keyfile, 32, 128
keyphrase, 32, 129
larger, 12, 129
level, 42, 129
license, 129
listcertificates, 34, 130
listchar, 107, 130
listcryptalgorithms, 25, 130
listfile, 131
listhashalgorithms, 33, 131
listsfxtypes, 131
locale, 107, 131
lowercase, 70, 132
lzma, 45, 132
mask, 54, 133
maximum, 43, 133
more, 134
move, 8, 56, 135
movearchive, 97, 135
namesfx, 59, 135
newer, 11
noextended, 51, 136
nofix, 136
normal, 43, 136
nosmartcard, 137
nozipextension, 136, 137
older, 11, 138
OpenFile, 38
optionchar, 108, 139
overwrite, 87, 139
passphrase, 23, 24, 140
path, 39, 140
ppmd, 46, 141
preview, 94, 141
print, 92, 141
recipient, 142
recurse, 38, 143
rename, 143
runafter, 61, 144
sfx, 57, 145
sfxdestination, 59, 145
sfxdirectories, 60, 145
sfxlogfile, 60, 145
sfxoverwrite, 60, 146
sfxuitype, 60, 146
shortname, 90, 147
shred, 56, 147
sign, 33
silent, 96, 149
smaller, 12, 150
sort, 55, 71, 150
span, 47, 151
speed, 43, 152
store, 43, 152
strict, 79, 152
substitution, 68, 90, 152
temp, 95, 154
test, 92, 154
times, 70, 155
translate, 89, 156
ts, 35
update, 21, 64
utf8, 157
utf8, 48
verifysigner, 74, 158
version, 18, 159
view, 8, 88, 159
warning, 93, 160
wipe, 160
zipdate, 53, 111, 161
zoneidentifier, 71
comments, 52
compress. See also: archives
compressing, 20, 176
 all files in a directory, 21
 ASCII/BINARY internal attribute, 96
 compression level, 42
 current directory, 6, 7
 digital certificate, 31, 33
 directories, 40
 diskette, 47
 encode, 97
 file attribute information, 50
 files in subdirectories, 38
 filters, 10
 freshen, 22
 generate list file, 98
 hash, 33
 incremental archiving, 22
 list files, 44
 listcertificates, 34
 methods, 45, 46, 102
 only changed files, 22
 only new files, 21
 open files, 38
 overview, 5
 path information, 39
 removing file attributes, 54
 selected files, 7
 signature, 31, 33
 single file, 6

- update, 21
 - with attributes, 50
- configuration file, 99
 - alternate, 105
- CRC, 182

D

- dates, 53
 - environment variable, 107
- dclimplode, 46
- defaults
 - changing, 102
 - changing with Options dialog, 103
 - resetting, 104
- deflate64, 45
- deleting original files, 8, 56
 - shred, 56
- digital certificates, 31, 76, 77
 - hash, 33
 - listcertificates, 34
 - revocation lists, 80
 - root, 78
 - setting a default, 35
 - sign, 33
 - strict checking, 79
 - Windows, 81
- digital signatures, 72, 76
 - attaching, 31, 79
 - authenticating, 72, 92
 - time stamping, 35
 - verifying, 74
- DOS file names, 90

E

- encode, 97
- encrypting files, 23
 - file names, 27
 - passphrase, 24
 - recipient list, 25
 - strong encryption, 16, 25
 - traditional ZIP encryption, 25
- end-of-line characters, 89
- entropy, 176
- environment variables
 - date, 107
 - locale, 107
 - path, 20
 - time, 107
- error messages, 162
 - treat warnings as, 93
- extended attribute storage, 51
- extracting
 - all files from an archive, 63
 - directory structure, 70
 - embedded files, 65

- files only for display, 72
- filters, 10
- freshen, 64
- from named pipe, 67
- list files, 72
- lowercase, 70
- new and existing files, 63
- newer versions and new files, 64
- only newer versions of files, 64
- overriding default settings, 64
- overview, 9
- sorting, 71
- times, 70
- to STDIN, 37
- translate, 89
- update, 10, 64

F

- FastAES, 30, 83
- Field Of Information Theory, 176
- file name encryption (FNE), 27
- filters, 10
- FIPS mode, 28
- format or wipe removable media, 48
- freshen, 22, 64

G

- GnuPG. *See* OpenPGP
- GZIP, 3, *See also*: archives

H

- hash algorithms, 33
- header comments, 53
- help system, 18

I

- IMG, 3, *See also*: archives
- information content, 176
- integrity test, 92
- international characters, 48
- Internet Explorer, 71
- ISO, 3, *See also*: archives

J

- JAR, 3, *See also*: archives

K

- keyfile, 32
- keypassphrase, 32
- keys, 77

L

licenses
 entering keys, 19
 list files, 44, 72, 98
 changing the list character, 107
 LZH, 3, See also: archives
 lzma, 45

M

md5, 33
 moving files, 8, 56

N

namesfx, 59
 naming conventions, 6
 noextended, 51

O

OpenPGP, 3, 16, 30, 84, See also: archives
 compared to X.509, 84
 keyrings, 85
 settings, 86
 options, 13, 109
 abbreviating, 14
 changing character, 108
 combining, 14
 command line, 14
 default values, 21, 100
 difference from commands, 13
 values, 15
 overwriting existing files, 87

P

passphrase, 23, 175
 password. *See* passphrase
 PKI, 77
 PKSFX, 57
 PKWARE, 19
 PKZIP, 176
 configuring, 99
 help, 18
 license keys, 19
 previewing command and option
 operations, 94
 support, 19
 v2.04 compatibility, 29, 52
 version information, 18
 ppmd, 46
 Pretty Good Privacy (PGP). *See* OpenPGP
 printing, 92
 private key, 77, 78
 backup, 81
 keyfile, 32

public key, 77, 78
 exporting, 81
 Public-Key Infrastructure (PKI), 77

R

RAR, 3, See also: archives
 recipient list, 25

S

selecting files, 10
 self-extracting archives, 57
 command-line options, 61
 converting from ZIP, 58
 converting to, 59
 graphical interface, 60
 logging messages, 60
 options, 59
 overwrite rules, 60
 run program, 61
 sfxdestination, 59
 sfxdirectories, 60
 sfxuitype, 60
 sha256, 33
 shred, 56
 signing, 77
 smart cards, 83
 sorting files, 55, 71
 spanning/splitting, 47
 split sizes, 47
 STDIN, 37
 STDOUT, 36
 strict checking, 79
 strong encryption, 16
 subdirectories, 174, 175, 181
 sub-options, 15
 substitution, 68, 90
 suppressing screen output, 96
 syntax, 4
 options, 14

T

TAR, 3, See also: archives
 Technical Support, 19
 test, 92
 text comments, 52
 time
 environment variable, 107
 time stamping, 35

U

update, 21, 64
 extract, 10
 utf8, 48
 UUEncode, 3, 97, See also: archives

V

verifysigner, 74
version command, 18
viewing archive contents, 8, 88
virus scanning, 64

W

warning messages, 162, 167
 pause, 93
 treat as error, 93
wildcards, 7
Windows
 file attributes, 22

 including open files in archives, 38
 path, 20
Windows 2000, 83

X

X.509, 77
 compared to OpenPGP, 84
XXEncode, 3, 97, *See also*: archives

Z

ZIP archives. *See* archives
zone identifier, 71