PKWARE®

Users Manual

- PKZIP® Server
- SecureZIP® Server
- SecureZIP Partner
- SecureZIP eBusiness Edition

2019-02-06

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 5
Adding a Single File 6
Adding Multiple Files 6
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 9
Extracting New and Newer Files 10

Using Filters When Selecting Files 10
Selecting Files by Date 10
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 14

Using Strong Encryption 15

2 GETTING STARTED 17

Varieties of Server Options 18
SecureZIP Server Standard Edition on UNIX/Linux 18
SecureZIP Server eBusiness Edition 19
PKZIP Server and SecureZIP Server: Enterprise Editions 19

Learning More and Getting Help 19

Using Help 20
Getting Version Information 20
Technical Support 21

Working With Your License 21
Entering License Keys 21
Getting License Information 22
Sharing a License (Windows) 22

Notes for UNIX Users 23
Using Wildcards with PKZIP on UNIX 23
Running the Program as Root 23

Notes for Windows users 24
Setting PKZIP in the Path (Windows) 24

Information for PartnerLink™ Sponsors and Partners 25
If You Are a Sponsor: Sign the Central Directory 25
If You Are a Partner 25
About SecureZIP Partner 25
To Run SecureZIP Partner 26
Designating a Sponsor 26
Listing Available Sponsors 27
Commands and Options Available with SecureZIP Partner 28

3 ADDING FILES TO AN ARCHIVE 29

Default Values for Commands and Options 29

Creating and Updating Archives 29
Adding All Files in a Directory 29
Adding New and Modified Files 30
Adding Only Files That Have Changed 30
Incremental Archiving (Windows) 31

Encrypting Files That You Add to an Archive 31
Encrypting Files with a Passphrase 32
Encrypting Files with a Recipient List 33
Encrypting File Names 36
Encrypting Using Only FIPS-Approved Algorithms 37
Accessing Recipients in an LDAP Directory 39
Contingency Keys 41
Creating OpenPGP Files 42

Attaching Digital Signatures 42
Commands and Options for Signing Archives 43
Setting a Default Certificate 46
Time Stamping Your Signed ZIP Archive 47

Writing an Archive to STDOUT and Special Files 48
Writing an Archive to STDOUT 48
Writing an Archive to a Named Pipe, UNIX Domain Socket, or Device File 49
Setting a Timeout for PKZIP to Wait (UNIX) 50

Adding Data from STDIN or Special Files 50

Adding Streamed Data on UNIX 50
Adding Streamed Data from a Named Pipe on Windows 51
Adding Streamed Data from STDIN 51

Compressing Files in Subdirectories 52

Compressing Open Files 52

Storing Directory Path Information 53
Additional Methods for Storing Directory Path Information 53
Storing and Recreating Directory Path Information 54

Setting the Compression Level 55
Specifying a Compression Level from 0-9 56
Specifying a Compression Level by Name 56

Compressing Files with a List File 57
Getting a List of Files from Standard Input 58

Compressing Files with the Deflate64 Method 58

Compressing Files with the BZIP2 Method 59

Compressing Files with the LZMA Method 59

Compressing Files Compatible with the Data Compression Library 59

Compressing Files with the PPMd Method 60

Compressing Files to a Specified Type of Archive 60

Compressing Files to Removable Media 61
Creating a Spanned Archive (Windows) 61
Creating a Split Archive 61

Preserving International Characters in File Names 62

Creating Multiple, Respective Archives 63

Storing File Information 63
Compressing Files with Specified Attributes (Windows) 63
Compressing Files Based on File Type (UNIX) 64
Following Links (UNIX) 66
Extended Attribute Storage 66

Including Additional Information in a ZIP File 67
Including a Text Comment 67
Including a Header Comment 68
Specifying the Date of a .ZIP File 69

Removing File Attributes (Windows) 69

Removing File Attributes (UNIX) 70

Sorting Files Within a .ZIP File 70

Moving Files to a .ZIP File **72**

Shredding Deleted Files **72**

Working with Self-Extracting (PKSFX) Archives **73**
Setting the PKSFXSDATA Environment Variable (UNIX) 74
Converting a Standard Archive to a Self-Extractor 75
Converting to a Self-Extractor with a Different Name 75
Options for Creating Self-Extractors 75
Run Programs with the Self-Extractor 77
Extraction Options for the Native Self-Extractor 77

4 EXTRACTING FILES **79**

Default Values for Commands and Options **79**

Extracting New and Existing Files **79**
Extracting All Files from an Archive 79
Extracting Newer Versions of Existing Files and New Files 80
Extracting Only Newer Versions of Files 80

Checking for Viruses when Extracting **80**

Extracting from an Archive Embedded in an Archive **81**

Extracting Passphrase-Protected Files **82**

Authenticating Digital Signatures **83**

Extracting Only Trusted Archives **84**
Specifying Trusted Signers 84

Extracting an Archive on STDIN or a Special File **86**
Extracting from an Archive on STDIN 86
Extracting an Archive from a Named Pipe, UNIX Domain Socket, or Device File 87

Extracting Data to STDOUT or Special Files **88**
Extracting to STDOUT 88
Extracting to Special Files 88

Extracting to Dynamically Named Folders **89**

Extracting Files in Lower Case **91**

Changing Ownership When Extracting (UNIX) **92**

Preserving File Times **92**

Retaining Directory Structure while Extracting **92**

Retaining Zone Identifier Information for Downloaded Files (Windows) **93**

Sorting Files in the Extract Directory **93**

Extracting Files Only for Display **94**

Extracting Files with a List File **94**

5 SENDING AN ARCHIVE BY FTP AND EMAIL **95**

Transferring an Archive with FTP **95**

Sending an Archive by Email **96**
Configuring Required Options 97
Specifying a Mail Server 97
Sending to Multiple Recipients 97
Sending to a List of Recipients 98
Sending Encrypted Attachments 98
Specifying Text in a File 98
Sending Copies 99
Sending Split Archives 99
Hiding the TO List 99
Including Instructions on How to Unzip 99
Using a ReplyTo Address 100

6 WORKING WITH DIGITAL SIGNATURES AND OPENPGP KEYS **101**

Public-Key Infrastructure and Digital Certificates **101**
Public-Key Infrastructure (PKI) 102
How the Keys Are Used 102
X.509 102
Digital Certificates 102
Certificate Authority (CA) 103
Private Key 103
Public Key 103
Certificate Authority and Root Certificates 103

Using X.509 Digital Signatures **103**
Attaching a Signature to an Existing Archive 104
Applying Strict Checking to Certificates 104
Checking for Revoked Certificates 106
Using Digital Certificates on Windows 107
Setting Up Stores for Digital Certificates on UNIX/Linux 109
Advanced Encryption Options in Windows 124
Enabling PKCS#11 Support in Windows and UNIX 125

Working with OpenPGP Files **126**
Overview: OpenPGP vs. X.509 126
Setting Up OpenPGP Keyrings 127
Working with Recipient Groups 128
Configuring Other OpenPGP Settings 129

7 MISCELLANEOUS OPERATIONS **130**

Overwriting Files **130**

Viewing the Contents of a ZIP File **131**
Displaying a Brief View of a ZIP File 131

Displaying a Detailed View of the ZIP File 131
Displaying Security Information of the ZIP File 133

Verifying the Encryption Type for an Archive 133

Verifying Recipients Listed for a File 134

Renaming Files 135

Translating End-of-Line Sequence 138

Converting File Names to a Short Format 138

Inserting a Timestamp in the Archive File Name 139

Printing the Contents of a ZIP File (Windows) 140

Testing the Integrity of an Archive 140

Handling Warnings 141
Pausing on Warnings 141
Ignoring Warnings 142

Treating Warnings as Errors 142

Previewing Command and Option Operations 143

Fixing a Corrupt ZIP File 143

Use an Alternate Drive for PKZIP Temporary Files 144

Suppressing Screen Output 145

Setting Internal Attributes 145

Encoding an Archive to another Type 145

Removing an Intermediate Archive 146

Generate a List File 147

Logging Events 147

Sending Information to an SNMP Host 148
Kinds and Contents of SNMP Traps Sent 149
The PKWARE MIB 150

Setting Execution Priority 150

8 CHANGING DEFAULTS FOR COMMANDS AND OPTIONS 152

Viewing Configuration Settings 152

How Default Settings Work 154

Filter Options **155**

Changing a Default Value **155**
Changing Defaults for Filter Options 155
Changing Defaults for Compression Method 156
Using the Options Dialog to Change Defaults 157

Resetting to Original Defaults **157**
Resetting Individual Defaults 157
Resetting All Defaults 158

Using an Alternate Configuration File **158**
Creating an Alternate Configuration File 158
Using an Alternate Configuration File 159

9 COMMAND CHARACTERISTICS **160**

Changing Date and Time Environment Variables **160**

Changing the List Character for List Files **160**

Changing the Command/Option Character **161**

A REFERENCE TO COMMANDS AND OPTIONS **162**

B ERROR AND WARNING MESSAGES **231**

Error Messages **231**

Warning Messages **238**

C FREQUENTLY ASKED QUESTIONS **244**

D HOW PKZIP WORKS **248**

Two Processes **248**

Compression **248**
Information Content 248
Binary Data Representation 249
Speed vs. Size 251

Archiving **251**

How PKZIP builds a .ZIP File **252**
CRC 254

Deleting Files from a .ZIP File **254**

Adding to an Existing .ZIP File **255**

E TIPS FOR SCRIPTING PKZIP ON UNIX 256

F MCAFEE EBUSINESS SERVER COMMAND OPTIONS 257

Using OpenPGP Mode 257

Using Legacy PGP Mode 261

1 The Basics

Welcome to PKZIP/SecureZIP Server. PKZIP Server and SecureZIP Server 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 Server is an enhanced version of PKZIP Server. 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 Server provides additional features—most notably, commands and options for using digital certificates and OpenPGP keys to do strong encryption and attach digital signatures. SecureZIP Server eBusiness Edition adds the ability to create and manage digital certificates and OpenPGP keys.

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 the command-line features of all editions of PKZIP Server and SecureZIP Server, both for Windows and for UNIX/Linux.

In general, references to PKZIP or PKZIP Server in the text apply equally to SecureZIP Server. SecureZIP Server includes all the features of PKZIP Server. If a feature is available only with SecureZIP Server and not with PKZIP Server, is included in the SecureZIP Server eBusiness Edition, or requires the Enterprise edition of one of these programs, this is noted in the text.

From now on, for brevity, the manual will generally refer to PKZIP Server as *PKZIP* and to SecureZIP Server as *SecureZIP*.

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 8 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, 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.

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

Your Work Environment: The Command Line

In PKZIP Server, 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:

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

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*. See a list of archive type extensions in "Supported Archive Types" in this chapter.

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*, 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.

- On Windows, 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.

  On UNIX systems, a trailing dot does not suppress the `.zip` extension. To suppress automatic adding of a file name extension on UNIX systems, use the ***noarchiveextension*** option. This option also works on Windows.

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 Windows 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. On UNIX, use a slash / instead of a backslash \ to indicate a subdirectory or set of files.

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. 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. On UNIX, the *Attr* column is replaced by a *Mode* column with permission numbers for each file.

For more information on the ***view*** command, see "Viewing the Contents of a ZIP File" in Chapter 7.

See Chapter 3 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 slash (`/`) (UNIX) or a backslash (`\`) (Windows):

```
pkzipc -extract data.zip january/
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 (*mmddyyyy*). As a result, only those text files that meet the additional requirement imposed by the ***after*** option are added.

```
pkzipc -add -after=03152006 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 9.

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.

Time unit	*Abbreviation*
Days (default)	d (or nothing)
Hours	h
Minutes	m
Seconds	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
```

You can also use ***k***, ***m***, ***g***, or ***t*** to specify kilobytes (1024 bytes), megabytes (1024 Kb), gigabytes (1024 Mb), and terabytes (1024 Gb), respectively.

The following command line adds files whose size is in the range 500 megabytes to 1 gigabyte, inclusive:

```
pkzipc -add -larger=500m -smaller=1g 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 8.

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 8.

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
```

Note: 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 8.

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 Server. Create OpenPGP-based archives and use its encryption on any file (not just ZIP archives) with SecureZIP Server in all environments, and with PKZIP Server in Windows.

Two methods of authentication exist to ensure that encrypted files are only opened by the right people: Symmetric keys (passphrases) or public/private key pairs (in the form of X.509 digital certificates and OpenPGP keys).

Note: SecureZIP supports both X.509 digital certificates and OpenPGP key pairs. When this document refers to *certificate-based encryption*, or *recipient-based encryption*, you can use either type of key. To learn more about the differences

between digital certificates and OpenPGP keys, see “Public-Key Infrastructure and Digital Certificates” in Chapter 6.

Traditional PKZIP encryption uses only the ***passphrase*** option. Strong encryption can be done with a passphrase, a public/private key pair or both. When you encrypt using a public/private key pair, only the owner of the private key—called a *recipient*—can decrypt. Using both types of authentication widens the number of people who can open the encrypted file, both those on the recipient list with the proper private key and anyone with the passphrase.

Use the ***passphrase*** option to apply either traditional or strong passphrase-based encryption.

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 3, “Extracting Passphrase-Protected Files” in Chapter 4, and “Working with OpenPGP Files” in Chapter 6.

2 Getting Started

Welcome to PKZIP/SecureZIP Server. PKZIP Server and SecureZIP Server 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 Server is an enhanced version of PKZIP Server. 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 Server provides additional features—most notably, commands and options for using digital certificates to do strong encryption and attach digital signatures. SecureZIP Server eBusiness Edition adds the ability to create and manage digital certificates and OpenPGP keys.

PKZIP Server and SecureZIP Server each come in both a Standard edition and an Enterprise edition. This table and the following sections describe the additional features included with SecureZIP Server that are not in PKZIP Server. They also describe the features added by the respective Enterprise editions of PKZIP Server and SecureZIP Server.

Feature	*PKZIP Standard*	*PKZIP Enterprise*	*SecureZIP Standard*	*SecureZIP Enterprise*
Large file size support	X	X	X	X
Very large archive support	X	X	X	X
Self-extracting files for end-users and other platforms		X	X	X
Decryption of PKI public-key encrypted ZIP archives	X	X	X	X
Attaching digital signatures to archives			X	X
Strong passphrase-based AES and 3DES data file protection		Windows only	X	X
Strong encryption using a digital certificate instead of a passphrase			X	X
Strong, certificate-based file name encryption			X	X

Feature	PKZIP Standard	PKZIP Enterprise	SecureZIP Standard	SecureZIP Enterprise
Creating OpenPGP (RFC 4880) encrypted files		Windows only	X	X
Opening OpenPGP files	X	X	X	X
Add digital timestamp from secure Time Stamp Authority			X	X
Error reporting for both attended and unattended operations	X	X	X	X
Email (SMTP) integration		X	X	X
FTP integration		X	X	X
Application Integration	X	X	X	X
Preserving Zone Identifier information added by Internet Explorer		Windows only		Windows only
Contingency Keys				X
LDAP Directory Integration				X

Varieties of Server Options

SecureZIP Server Standard Edition on UNIX/Linux

On UNIX and Linux, SecureZIP Server Standard Edition adds the following features to the set provided by PKZIP Server Standard Edition:

- Email and FTP integration: Options to create and transfer archives by email or FTP directly from the command line. See Chapter 5.
- PKSFX: The ability to create self-extracting ZIP files for use in either the native command line or graphical Windows environment. See “Working with Self-Extracting (PKSFX) Archives” in Chapter 3.
- Strong passphrase-based encryption: Strong encryption—the kind of encryption used by banks and the federal government—is much more secure than the weaker, traditional ZIP encryption provided by PKZIP Server (UNIX). See “Encrypting Files with a Passphrase” in Chapter 3.
- Strong encryption using a digital certificate instead of a passphrase: This kind of encryption is both more convenient and more secure than passphrase-based encryption, and it enables you to encrypt files just for the people you want to see them. See “Encrypting Files with a Recipient List” in Chapter 3.
- Strong file name encryption: With this feature, you can encrypt even the names of files in an archive so that only the intended recipients of the archive can read them. See “Encrypting File Names” in Chapter 3.
- Digital signatures: When you attach a digital signature, recipients of your files can be sure that the files are unchanged and really come from you. See “Attaching Digital Signatures” in Chapter 3.

SecureZIP Server eBusiness Edition

SecureZIP Server eBusiness Edition provides additional functionality regarding OpenPGP keys and X.509 certificates. These include:

- Ability to generate OpenPGP keys
- Convert X.509 certificates to OpenPGP keys
- Convert OpenPGP keys to X.509 certificates
- Signing OpenPGP keys

- If you are transitioning from the McAfee eBusiness Server (EBS), you can use SecureZIP Server eBusiness Edition in OpenPGP Mode to run many of your existing EBS scripts with minimal editing. The commands include ***decrypt***, ***encrypt***, and ***sign***. These commands and options are described in Appendix F, "McAfee eBusiness Server Command Options."

  Find more information about eBusiness Edition features, including installation and a command reference, in the *Getting Started with PKWARE Key Maker* guide.

PKZIP Server and SecureZIP Server: Enterprise Editions

The Enterprise editions of SecureZIP Server and PKZIP Server each add an additional module of functionality to the respective products.

PKZIP Server Enterprise Edition

PKZIP Server Enterprise Edition includes the *Enhanced Data Processing* module. This module adds these features to PKZIP Server (all are included in SecureZIP Server):

- Email and FTP integration: Options to create and transfer archives by email or FTP directly from the command line. See Chapter 5.
- PKSFX: The ability to create self-extracting ZIP files for use in either the native command line or graphical Windows environment. See “Working with Self-Extracting (PKSFX) Archives” in Chapter 3.

SecureZIP Server Enterprise Edition

SecureZIP Server Enterprise Edition includes the *Directory Integration* module. This module enables SecureZIP Server to access digital certificates stored on directory servers anywhere in the enterprise. Being able to access certificates on directory servers makes it much more convenient to do strong certificate-based encryption, as you can encrypt for a set of recipients without needing to have the certificate for each recipient on your own machine. See “Accessing Recipients in an LDAP Directory” in Chapter 3.

SecureZIP Server Enterprise Edition also includes the *Contingency Keys* module. Contingency keys are digital certificate-based keys that an administrator can have automatically included in the recipient list whenever PKZIP does strong encryption. See “Contingency Keys” in Chapter 3 for more information.

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): 14.50.1217
Product Version: 14.50.0009
```

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.

Sub-Option	PKZIP Returns	For example
major	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
minor	The minor number of the release. For example, if the version number is 12.10.1054, the value returned is 10.	pkzipc -version=minor
step	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
product	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:

https://support.pkware.com/

Working With Your License

Entering License Keys

Note: To use SecureZIP Partner, as a participant in PKWARE PartnerLink, *you do not need to enter a license key*. You can ignore this section and related sections on getting license information and sharing a license, later in this chapter.

On UNIX/Linux, you must enter your license keys to activate the product and for any add-on modules after you complete the installation. On Windows, you can activate and enter license keys during the installation.

To enter a (single) license key after installing PKZIP, use the ***enterlicensekey*** command.

On UNIX, you must run PKZIP as root to use the ***enterlicensekey*** command. If you try to run the command as an ordinary user instead of as the super user, you get an error.

On UNIX, running the ***enterlicensekey*** command creates a file `license.ini` (if it does not exist already) in the PKZIP installation directory where the `pkzipc` executable is located. The license file must be in this directory for PKZIP to find it. The default location of this directory is:

- `/opt/pkware/pkzip/bin/` on Solaris and HP-UX
- `/usr/pkware/pkzip/bin/` on AIX and Linux.

Make the directory and its files readable for all users and writable for none.

You can use the ***enterlicensekey*** command to enter license keys on Windows as well. You may want to do this if you need to enter the license key for an add-on module that you purchase sometime after you purchased the base product.

To enter a license key:

1. (UNIX) Become the super user, to run the program as root.
2. At the command prompt, type the following and press ENTER:

   pkzipc -enterlicensekey

   PKZIP prompts you for a product license key.
3. Enter a product license key and press ENTER.

Repeat these steps for each license key you have. For example, if you have a license key for an add-on module, repeat the steps above to enter the license key for that module after you enter the license key for the base product.

Getting License Information

To display the PKZIP license information on your screen, do the following:

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

pkzipc -license

Sharing a License (Windows)

To enable multiple users on different Windows machines to share a site license for PKZIP or an add-on module, supply the license key on each machine. To do this, you can install PKZIP from a batch file and pass the license key as a property to the installer.

The installation command line looks like this:

<name of PKZIP installation file> /S /v"<properties>"

where:

- `/S` is a switch that tells InstallShield® to run silently and not to display various initial screens (that say, for example, *Preparing to install…*)
- `/v` is a switch that must be used to pass any specified PKZIP properties to the Windows installer.
- `<properties>` is a list of property settings

You can also optionally pass in a switch to specify either the *Basic UI*, that displays a dialog containing only a *Cancel* button to allow canceling of the installation; or *No UI*, that displays no dialog. Both *Basic UI* and *No UI* can run unattended. The default is the full, graphical UI, which is interactive and so cannot run unattended.

Switch	*Specifies*
`/qb`	Basic UI
`/qn`	No UI

Any quotes (`"`) in the parameters must be escaped with a backslash (`\`).

Examples:

<name of PKZIP installation file> /S /v/qb

<name of PKZIP installation file> /S /v"/qb LICENSE_KEY=<Your license key>"

If you want PKZIP to install somewhere other than the system's `Program Files` directory, use the `INSTALLDIR` property and set it to the new location. For example:

```
<name of PKZIP installation file> /S /v"INSTALLDIR=\"\My
Programs\PKWARE\""
```

PKZIP checks the PKWARE license key each time the program runs. Use the `LICENSE_KEY` property to set the license key on users' systems. For example, the following command line specifies both a custom installation directory and a license key:

```
<name of PKZIP installation file> /S /v"INSTALLDIR=\"\My
Programs\PKWARE\" LICENSE_KEY=<Your license key>"
```

Notes for UNIX Users

Using Wildcards with PKZIP on UNIX

If your UNIX shell is set up to automatically expand wildcards, you should put file specifications that use wildcards—for example, `*.htm`— in quotation marks—like this: `"*.htm"`—on the command line to prevent the shell from expanding them.

Allowing the shell to expand wildcard file specifications into an explicit list of files can cause the PKZIP ***recurse*** and ***directories*** options not to work properly. Placing a wildcard pattern in quotes instructs the shell to pass the pattern as an argument to PKZIP, which then expands it.

PKZIP can interpret and expand the following wildcard patterns:

Pattern	*Example*
*****	`*`
<pattern>	`*.txt, *f.txt`
<pattern>*	`h*, file.f*`
<pattern>*<pattern>	`a*.txt`
<pattern>*	`*.*, *ab*`

Running the Program as Root

Setting the `set-uid` bit on the `pkzipc` binary causes PKZIP to run as `root`. It also causes PKZIP to run any program that it may launch—such as the ftp client (***ftp*** option) or a virus scanner (***avscan*** option)—as root.

Use considerable caution in setting the `set-uid` bit to run PKZIP as root. It is very easy for a program running as root to overwrite system files, and setting the `set-uid` bit on any program raises security concerns. Configure PKZIP to run this way only in keeping with organizational security policies and on the instructions of a system administrator.

Notes for Windows users

Setting PKZIP in the Path (Windows)

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. In Windows 10, open Settings.
3. In the Control Panel, double click the System icon (or click Advanced System Settings). The System (Properties) dialog appears. In Windows 10, search for Environment Variables to display the System Properties dialog.
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. In Windows 10, click Edit. 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 Server is installed. In Windows 10, click New, then enter the path (quotes are not necessary).

   For example, assuming that PKZIP Server (`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 Server from any directory without specifying a path. This change will take effect the next time you open a Command Prompt window to run PKZIP Server.

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

Information for PartnerLink™ Sponsors and Partners

PKWARE PartnerLink enables a *sponsor* organization that has SecureZIP Server to distribute to *partner* organizations the SecureZIP Partner application. That is, an organization that licenses SecureZIP Server can exchange strongly encrypted archives with selected Partner organizations, offering both sponsor and partner organizations secure business-to-business communication. SecureZIP Partner is a special version of SecureZIP Server. It provides most of the commands and options of SecureZIP Server but works only with archives created by (or for) a sponsor. Archives created using SecureZIP Partner are automatically strongly encrypted for sponsor recipients.

This section applies only to participants in the PKWARE PartnerLink program, including users of SecureZIP Partner. Other readers may skip this section.

Note: SecureZIP Partner was called *SecureZIP Reader/SecureLink* prior to release 8.5 of SecureZIP Server.

To use SecureZIP Partner, *you do not need to enter a license key*. Use of the software is controlled by the Sponsor Distribution Packages you install. Users of SecureZIP Partner can ignore the section “Entering License Keys” and related sections on getting license information and sharing a license.

If You Are a Sponsor: Sign the Central Directory

A sponsor organization uses SecureZIP Server as usual to work with archives for, or from, a partner. There is just one special requirement when creating an archive for a partner: you must sign the central directory of the archive using a certificate included in the Sponsor Distribution Package (SDP). Otherwise a partner cannot extract the archive.

To sign an archive, use the ***certificate*** option. (See “Attaching Digital Signatures” in Chapter 3.) You may optionally sign files in addition to signing the archive itself. Use the ***sign*** option to specify what to sign: the central directory, the archive’s files, or both.

For example, the following command line adds files to archive `test.zip`. The command line signs using the *John Q. Public* certificate and attaches the signature to the central directory only, not to the archive’s files.

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

Contact PKWARE for information about participating in the PartnerLink program or assembling a Sponsor Distribution Package for partners.

If You Are a Partner

A PartnerLink partner uses the SecureZIP Partner application to work with archives. The SecureZIP Server users manual you are now reading also serves as a user guide for SecureZIP Partner.

See the *PartnerLink Partner Setup Guide: Windows/UNIX/Linux* for information on installing SecureZIP Partner and on setting up as a partner to work with sponsor archives.

About SecureZIP Partner

SecureZIP Partner does basically two kinds of operations:

- **Extracts files from sponsor archives:** SecureZIP Partner uses SecureZIP Server commands and options to extract files from a ZIP archive received from a sponsor. These commands and options include those to decrypt and decompress files and to authenticate digital signatures. SecureZIP Partner can only extract archives digitally signed by a PartnerLink sponsor.
- **Creates archives for sponsors:** SecureZIP Partner uses SecureZIP commands and options to add files to a ZIP archive, including commands and options to compress, encrypt, and digitally sign files.

  SecureZIP Partner can create and update archives only for a designated sponsor. Archives are automatically encrypted for all sponsor recipients whose certificates are included in the sponsor's SDP. Only those sponsor recipients can decrypt and read the files in an archive created by SecureZIP Partner. SecureZIP Partner does not use passphrase-based encryption.

Note: Because SecureZIP Partner automatically encrypts for sponsor recipients—and only for sponsor recipients—when adding files to an archive, partners cannot decrypt archives that they use SecureZIP Partner to create. So partners must be careful not to delete files they want to keep after placing them in an archive. A copy of a file in an archive will be inaccessible to the creator of the archive.

To Run SecureZIP Partner

The command to run SecureZIP Partner is ***pkzipr***; the command to run SecureZIP Server is ***pkzipc***. So, for example, where the manual says to use a command like the following to extract all files from archive `myfiles.zip`:

```
pkzipc -extract myfiles.zip
```

you would instead use a command line like one of those below to do the same thing with SecureZIP Partner:

```
pkzipr -extract myfiles.zip
pkzipr -extract -sponsor="Example Corp" myfiles.zip
```

Designating a Sponsor

SecureZIP Partner only operates on archives from, or for, a sponsor. A special ***sponsor*** option is provided just for SecureZIP Partner to designate a sponsor.

sponsor

The ***sponsor*** option is only required when creating or updating an archive with the ***add*** command. The option can be explicitly included on the command line, or you can configure SecureZIP to designate a sponsor by default (see Chapter 8).

You do not need the ***sponsor*** option when extracting an archive with the ***extract*** command. If the option is not used when extracting, the signature on the archive is checked against all sponsors defined on the system.

Use the ***sponsor*** option when extracting if you want to ensure that only an archive from the specified sponsor is extracted. For example, you may have a script to process archives from a particular sponsor. Use the ***sponsor*** option

with command lines in the script to ensure that the script does not inadvertently process an archive from some other sponsor.

You can use the ***sponsor*** option multiple times on the same command line when extracting but only once per command line when adding files to an archive.

The ***sponsor*** option accepts either a sponsor's common name or sponsor ID to identify a sponsor. To find out this information about a sponsor, use the PKSponsor ***list*** command or the SecureZIP Partner ***listSponsors*** command, to list sponsors. (PKSponsor is a tool included with SecureZIP Partner for setting up as a partner. See the *PartnerLink Partner Setup Guide: Windows/UNIX/Linux*.)

For example, the following command line adds files to a ZIP archive for sponsor Example Corp. It references Example Corp by common name:

```
pkzipr -add myfiles.zip -sponsor="Example Corp" *.doc
```

The similar example below uses the sponsor ID to reference a sponsor:

```
pkzipr -add myfiles.zip -sponsor=15 *.doc
```

If you have received an archive called `myfiles.zip`, and you are uncertain which of your sponsors it came from, use the ***sponsor*** option twice to extract the archive's files from either sponsor:

```
pkzipr -extract -sponsor="Example Corp" -sponsor=20
myfiles.zip
```

Listing Available Sponsors

SecureZIP Partner provides a ***listSponsors*** command to list sponsors, like the PKSponsor ***list*** command.

listSponsors

The following command line returns a list of sponsors on the system:

```
pkzipr -listsponsors
```

Output from ***listSponsors*** looks like this:

```
----- Sponsor #1 -----
        Sponsor: PKWARE, Inc.
        Sponsor ID: 0
        Type: Read/Write
        Description: <Sponsor 1's comment, if any>
----------------------
----- Sponsor #2 -----
        Sponsor: ABC Corp
        Sponsor ID: 1
        Type: Read/Write
        Description:
----------------------
2 sponsor(s) installed
```

The table below explains the fields.

Field	Description
Sponsor	Common name of a sponsor
Sponsor ID	ID of a sponsor
Type	Functionality profile. *Read/Write* indicates that functionality is supported both for extracting sponsor archives and for creating archives for sponsors.
Description	Optional comment of sponsor's

Commands and Options Available with SecureZIP Partner

SecureZIP Partner enables you to use nearly all SecureZIP Server commands and options. Only a few cannot be used, generally because they cannot be constrained to work only with archives created by or for a sponsor.

The SecureZIP Server commands and options that you cannot use are listed in the following table.

Commands and options *not* available in SecureZIP Partner

ArchiveType	MailTo*	SfxDirectories
Encode*	NameSfx	SfxLogfile
EnterLicenseKey	NoFix	SfxOverwrite
Fix	Recipient	SfxUIType
FTP*	RunAfter	VerifySigner
LDAP	Sfx	
ListSfxTypes	SfxDestination	

Notes:

- Items flagged with an asterisk (*) in the table above have both a command form and an option form. The command form is not available in SecureZIP Partner. See "Understanding Commands and Options" in Chapter 1 for more information.
- The ***view*** command does not work on archives that you create for a sponsor using encrypted file names (see the ***cd*** option).

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 ***all***, which causes the command to add all files. See Chapter 8 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 ***all*** 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 8.

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 (Windows)

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 your platform and 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 6.

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, particularly on UNIX, 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.

Specify an Encryption Method

listcryptalgorithms, cryptalgorithm

When you use strong encryption (available on UNIX only with SecureZIP), you have 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, SecureZIP applies traditional PKWARE encryption.

Encrypting Files with a Recipient List

recipient

Note: The ***recipient*** option is available only with SecureZIP. You will learn more about digital certificates and recipient lists in Chapter 6.

Use the ***recipient*** option with the ***add*** command to strongly encrypt files and specify one or more digital certificates representing the people whom you want to allow to decrypt, also known as a recipient list

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 the ***ldap*** option (see page 39) to cause SecureZIP to search a specified LDAP directory.

Use any of the following criteria to specify recipients:

Criterion	*To use*	*For example*
Common name	Specify, in quotes, the common name of the subject of the certificate (that is, the *cn* 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"
Email address	Specify the email address of the certificate (that is, the *e* field in a string representation of a certificate); optionally, precede with: e=	-recipient=e=john.public@xyz.com -recipient=john.public@xyz.com

Criterion	To use	For example
LDAP filter	Specify the LDAP filter that you want to use to filter a search for certificates on an LDAP server that you are accessing with the ***ldap*** option; precede with: f= Use quotes if the filter string contains a space. Place the quotes around the entire filter string, including "f=". Include the following LDAP presence filter, as shown in the examples at right, to limit the search to LDAP entries that are certificates: (&(userCertificate=*)(…)) Use standard LDAP filter syntax after the "f=" prefix. This sub-option is for use only when the ***ldap*** option is used.	-recipient=f=(&(userCertificate=*)(ou=Sales)) -recipient="f=(&(userCertificate=*)(ou=Regional Sales))"

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 ***listcryptalgorithms*** command.

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

Encrypting with OpenPGP

You can also specify OpenPGP keys to define a recipient list. You must first configure SecureZIP to enable OpenPGP on your system. See “Setting Up OpenPGP Keyrings ” in Chapter 6.

When defining a recipient list, you can search on the name and email address as with X.509 certificates. You can also search on the OpenPGP KeyID (short or long) with this command:

```
pkzipc -add -recipient=kid=<KeyID_characters> test.zip
```

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:

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

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.

FIPS-validated encryption algorithms
3DES-168, AES-128, AES-192, AES-256
FIPS-validated hashing algorithms
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 UNIX, the option is disabled by 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 ***-cryptoptions*** command. See "Advanced Encryption Options in Windows" in Chapter 6.

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
```

Accessing Recipients in an LDAP Directory

ldap

The ***ldap*** option enables you to access X.509 digital certificates in a Lightweight Directory Access Protocol (LDAP) directory.

> To access certificate stores in directories requires SecureZIP Enterprise. SecureZIP accesses certificates in directory stores by making LDAP-based queries to the target directories.

Ordinarily, when you use the ***recipient*** option to do certificate-based encryption, SecureZIP looks for certificates only in your system's local certificate stores. The ***ldap*** option enables you to point SecureZIP to an LDAP directory instead. With the ***ldap*** option, SecureZIP searches the specified LDAP directory first and only looks in local stores if it does not find the certificate it is seeking on an LDAP server.

You cannot use the ***ldap*** option to locate OpenPGP keys.

You can use the ***ldap*** option multiple times to specify multiple LDAP directories to search. Directories are searched in the order listed. If SecureZIP is unable to connect to a directory, SecureZIP issues a warning and tries the next directory.

Here is what SecureZIP does if multiple certificates are found that match a recipient:

- If multiple matching certificates are found in the same LDAP entry, SecureZIP picks the (valid) certificate whose expiration date is farthest in the future. No warning is generated.
- If multiple LDAP entries are found, each containing a matching certificate, SecureZIP uses a certificate from each entry to encrypt the archive and issues warning 59 (*Multiple certificates found*). The certificates may belong to different people, in which case the owner of any of them can decrypt.

The ***ldap*** option has several components, or fields. Only the last one, ***ldap_base***, is always required. The other fields are required only if needed to access a particular LDAP server.

The ***ldap*** option has the following syntax (optional fields are bracketed):

`-ldap=[[`*userid*`:`*passphrase*`@]`*server*`[:`*port*`][,ssl]/]`*ldap_base*

where:

- `userid` (optional) is the user account with which to log in if the LDAP server requires a login.
- `passphrase` (optional) is the passphrase associated with the user account.
- `server` (optional) is the LDAP server name or TCP/IP address.
- `port` (optional) is the TCP/IP port to use. The default is 389 if no port is specified.
- `ssl` (optional) tells underlying library to use encrypted communication to LDAP server, the default port will be 636 if no port is specified. (Windows Only)
- `ldap_base` (required) is the name of the entry that SecureZIP should use as the base or root of the LDAP search for certificates, analogous to a root folder or directory in a file system.

  The query string format for ldap_base can vary between LDAP implementations. For example, a server may expect query strings in the Internet domain-style format used by default by Microsoft Active Directory (for example, `cn=users,dc=xyz,dc=com`), or it may expect them in X.500 naming format (for example, `o=xyz,c=US`). Check with your LDAP or network administrator for the format to use.

Examples:

```
pkzipc -add -
ldap=john_p:mysecret@192.172.0.1:389/cn=users,dc=xyz,dc=com
-recipient="Mary Samplename" save.zip *.doc

pkzipc -add -
ldap=jon_p:mysecret@192.172.0.1/cn=users,dc=xyz,dc=com
-recipient="Mary Samplename" save.zip *.doc

pkzipc -add -ldap=192.172.0.1/cn=users,dc=xyz,dc=com
-recipient=e=mary.samplename@xyz.com save.zip *.doc

pkzipc -add -ldap=/cn=users,dc=xyz,dc=com
-recipient=e=mary.samplename@xyz.com save.zip *.doc
```

The ***ldap*** option must appear *before* the ***recipient*** option, as shown in the examples above, when the two options are used together in a command line.

To avoid having to type a frequently used ***ldap*** option setting, use the ***configure*** command to enable the option setting by default. For example:

```
pkzipc -config -ldap=192.172.0.1/cn=users,dc=xyz,dc=com
```

SecureZIP tests an LDAP connection immediately when you configure it. If the connection is bad, SecureZIP returns a warning to inform you of the problem before you try to use the connection to do encryption.

If you configure a default ***ldap*** option setting, it is applied implicitly whenever you use the ***recipient*** option to encrypt.

To remove configured settings for LDAP servers, use the ***--ldap*** option (two hyphens):

- Use the ***--ldap*** option with the ***add*** command (and the ***recipient*** option) to ignore configured ***ldap*** settings just in the current command.
- Use the ***--ldap*** option with the ***configuration*** command to remove any configured default ***ldap*** settings.

  The ***default*** command, which globally restores initial defaults, also removes configured ***ldap*** settings.

Note: The ***ldap*** option can only be used to point SecureZIP to an LDAP server to search for X.509 certificates to use for encryption, not for digitally signing files. Certificate-based encryption uses *public keys*; attaching a digital signature requires access to a *private key*. SecureZIP can only access public keys in certificates in an LDAP directory.

Contingency Keys

Note: To configure contingency keys, you must have a license for SecureZIP Server Enterprise Edition. A separate Policy Manager tool, which runs on Windows, is used to configure contingency keys. The Policy Manager is not provided for PKZIP Server.

Contingency keys, if configured, are used whenever SecureZIP users encrypt.

Contingency keys are recipient keys that an administrator can have automatically included in the recipient list whenever SecureZIP does strong encryption.

Contingency keys enable an organization to decrypt files encrypted by anyone in the organization, whether the files are passphrase encrypted or encrypted for specific recipients. Contingency keys are a safeguard to be sure that important information belonging to the organization does not become inaccessible because no one in the organization can decrypt it.

A contingency key is an ordinary cryptographic key from a digital certificate. The special thing about it is that, once the key is designated as a contingency key, it is automatically included as a recipient whenever SecureZIP encrypts files. This enables the owner of the key to decrypt the files.

If defined, contingency keys are used whenever SecureZIP encrypts. They are used even when the user chooses (strong) passphrase-based encryption and does not pick any recipients.

When defining a contingency key, the administrator can set these options:

- Show Contingency Key Settings allows the administrator to display a list of contingency keys in use.
- Show Contingency Keys During Encryption causes SecureZIP to display a line that states the number of contingency keys in use when encrypting. For example:

```
Using 2 contingency keys
```

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 6 for more information.

Before creating any OpenPGP files with SecureZIP, be sure to set up at least one keyring. See "Setting Up OpenPGP Keyrings" in Chapter 6 for details.

When you create an OpenPGP file containing more than one source file using ***-archivetype=pgp***, as in the following example, SecureZIP creates a GNU TAR archive, copies the selected files 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.

If there is only one source file to be encrypted with OpenPGP, SecureZIP skips creating the GNU TAR archive and simply creates the *.pgp file. You can also use the ***-archiveeach*** command to create separate OpenPGP files for each matching file. Be sure to add the command before specifying the file(s) you want to compress.

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

Note: When adding text files to an OpenPGP archive, be aware that the format automatically converts line endings in the text file to the Windows CR/LF format. By default, when you extract the file on a Macintosh or UNIX system using PKZIP or SecureZIP, line endings will convert to the native format.

To ensure proper handling of text files across platforms, add the ***-binary*** option to your command.

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, 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 6.

Commands and Options for Signing Archives

certificate

Use the ***certificate*** option to specify a certificate or OpenPGP key to use to sign files. To specify a certificate or key, 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 *cn* field in a string representation of a certificate); optionally, precede with: cn= SecureZIP searches for certificates by common name by default.	***-certificate=cn="John Public"*** ***-certificate="John Public"***
<Email address>	Specify the email address of the certificate (that is, the *e* field in a string representation of a certificate); optionally, precede with: e=	***-certificate=e=john.public@xyz.com*** ***-certificate=john.public@xyz.com***
#<file name>	Specify the name and location of a file containing the certificate to use. 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.	***pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key save.zip *.doc*** ***pkzipc -add -certificate=#mycert.p12 -keypassphrase="my passphrase" save.zip *.doc***

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 ***keypassphrase*** 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), an OpenPGP private key, 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*
sha1	Uses the SHA-1 hashing algorithm (default) (not FIPS-compatible; cannot be used with the ***fipsmode*** option)
sha256	Uses the SHA-256 hashing algorithm (***fipsmode*** default)
sha384	Uses the SHA-384 hashing algorithm
sha512	Uses the SHA-512 hashing algorithm

md5	Uses the MD5 hashing algorithm (not FIPS-compatible; cannot be used with the ***fipsmode*** 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*
cd	Sign only the central directory of the archive, not the files in the archive	-sign=cd
files	Sign only the files in the archive, not the archive itself	-sign=files
all (Default)	Sign both the archived files and the archive itself	-sign=all
none	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 "Using X.509 Digital Signatures" in Chapter 6 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 keys are enabled and available on the system, these will be displayed, including the Key ID value.

Note: If an OpenPGP key contains more than one userid, multiple certificates will display.

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.

Sub-option	*Description*	*Example*
my	Lists certificates in the MY store. This store contains your personal certificates with private keys. If OpenPGP keys are enabled and available on the system, these will be displayed, including the Key ID value.	pkzipc -listcertificates or pkzipc -listcert=my
addressbook	Lists certificates in the AddressBook store. This store contains public certificates and public keys belonging to other people. If OpenPGP keys are enabled and available on the system, these will be displayed, including the Key ID value.	pkzipc -listcert=addressbook
ca	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
root	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 8 for more information on setting and changing defaults in PKZIP Server.

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 time stamping 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 time stamps 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 PKZIP Server's support for digital time-stamping, you can add a timestamp to any signed archive.

Note: PKZIP Server only supports digital time-stamping 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 *.doc
```

where `<TSA_URL>` is the location of your Time Stamp Authority's service.

PKZIP 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. PKZIP 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. PKZIP Server 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 time-stamped 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
```

In this section, you will see how to create an archive as a data stream to send to 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, UNIX Domain Socket, or Device File

An archive can be written to a named pipe (Windows or UNIX) or UNIX socket or a device file 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 or socket in the command line in place of the name of an archive file.

```
pkzipc -add -noarchiveextension <name of pipe or socket> <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 or socket.

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 or
socket> <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.

Setting a Timeout for PKZIP to Wait (UNIX)

timeout

The ***timeout*** option sets a specified number of seconds for PKZIP to wait for another process to send or be ready to receive (more) data on a socket or device file. The timeout gives the other process time to handle data from PKZIP or to produce data for PKZIP to act on. By default, PKZIP waits 30 seconds. If the timeout period elapses without a response from the other process, PKZIP returns an error and halts processing.

For example, the following command line extracts files from an archive on a socket using a timeout of 60 seconds:

```
pkzipc -extract -noarchiveextension -timeout=60 mysocket
```

The timeout option has a default value of 30 seconds if no value is specified. The option is configurable.

Adding Data from STDIN or Special Files

Besides regular files, you can add to an archive data streamed from STDIN, a named pipe, a UNIX domain socket, or a device file. On Windows, adding streamed data uses the same command line syntax used to add regular files. On UNIX, you must specify the ***stream*** option.

Adding Streamed Data on UNIX

stream

On UNIX, PKZIP ordinarily ignores pipe and socket files. The ***filetype*** option can be used to cause PKZIP to include definitions (name, permissions, times, and so on) to recreate pipes, sockets, or devices but does not capture their data. The ***stream*** option, used with ***filetype***, gets the data from these sorts of files.

Sub-options of the ***filetype*** option enable you to specify types of files to include or exclude. The command line below uses ***filetype*** with the ***pipe*** sub-option to include pipe files. (There are also ***block***, ***char***, and ***socket*** sub-options). The command line uses the ***stream*** option to get the pipe data. Data is saved and referenced in the archive by the name of the pipe—`mystream` in the example.

```
pkzipc -add -filetype=pipe -stream data.zip mystream
```

You can use the ***rename*** option to change the name by which the data is stored in the archive. For example, the following command line renames it `stream_data.txt`:

```
pkzipc -add -filetype=pipe -stream -
rename=/mystream/stream_data.txt/ data.zip mystream
```

The ***stream*** option is also used when extracting streamed data, to write the data to a pipe of the same name as the file to be extracted:

```
pkzipc -extract -stream data.zip mystream
```

Without the ***stream*** option, the command line writes the data to an ordinary file named `mystream`. If a pipe of that name exists, the pipe is overwritten (See "Extracting Data to STDOUT or Special Files" in Chapter 4).

Adding Streamed Data from a Named Pipe on Windows

On Windows, named pipes are all located in the same `pipe` folder on each system: `\\<machine_name>\pipe\`. The command line below archives data from pipe `mystream` on the local system:

```
pkzipc -add data.zip \\.\pipe\mystream
```

The dot (.) indicates the local system. To reference a pipe on another system, specify the machine name:

```
pkzipc -add data.zip \\boulder\pipe\mystream
```

Data is stored in the archive under the name `mystream`.

You can use the ***rename*** option to store the data under a different name in the archive. The following command line renames the archived copy of the data `output.txt`:

```
pkzipc -add -rename=/mystream/output.txt/ data.zip
\\boulder\pipe\mystream
```

Adding Streamed Data from STDIN

To specify standard input (STDIN) as a data source, use a hyphen (-).

In the following command line, a program pipes data to PKZIP, which PKZIP reads from STDIN and archives. The data is stored and referenced in the archive as - (hyphen).

```
<command> | pkzipc -add data.zip -
```

Use the ***rename*** option to give the archived copy of the data a friendlier name. For example:

```
pkzipc -add -rename=/-/output.txt/ data.zip -
```

The command line below combines reading streamed data and writing it to a streamed archive. From STDIN, PKZIP reads the data output by <command>, archives the data, and writes the anonymous archive to STDOUT. (See "Writing an Archive to STDOUT and Special Files."):

```
<command> | pkzipc -add -noArchiveExtension -silent=all -stream --
```

Like STDIN, STDOUT is indicated by a hyphen. At the end of the command line, the first hyphen identifies where to write the archive (STDOUT), and the second hyphen specifies the data source (STDIN).

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.

Note: UNIX users should use the ***include*** option or place quotation marks around wildcard designations to avoid automatic wildcard expansion by the shell, which may interfere with your pattern search. See “Using Wildcards with PKZIP on UNIX” in Chapter 1.

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.

Note: PKZIP on UNIX and Linux includes open files automatically in your archive. You do not need to use this option when running these operating systems.

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.

Sub-Option	*Description*	*For example*
never	PKZIP does not include any open files. A warning will appear if a matching file is open.	***pkzipc -add -OpenFile=never test.zip *.bmp***
all	PKZIP includes all matching open files without prompting first. A message noting each open file is included in the standard output.	***pkzipc -add -OpenFile test.zip *.bmp***

Sub-Option	Description	For example
prompt	PKZIP notifies you when a matching file is open, and asks whether to add the open file or skip it.	***pkzipc -add -OpenFile=prompt test.zip *.bmp***

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.

Sub-option	To	For example
current	Store the directory path relative to the current location.	***pkzipc -add -path=current docs.zip docs/**** In this example, only directory information under the docs directory will be stored. Parent directory information will not be stored.
root, full	Store the full path, starting from the root directory down.	***pkzipc -add -path=root docs.zip docs/**** In this example, the entire directory path, starting from "root" directory will be stored.
specify	Stores path information for subdirectories under the specified directories	***pkzipc -add -path=specify docs.zip temp/docs/**** Stores path information for subdirectories under temp\docs.

Sub-option	To	For example
relative	Store the directory path relative to the current working directory of the drive specified. **(Windows)**	***pkzipc -add -directories=relative docs.zip c:*.doc z:*.doc*** 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.
none	Turn off the path option. (Used to override configuration file).	***pkzipc -add -path=none docs.zip /temp/docs/**** In this example, only the file names are stored.

Storing and Recreating Directory Path Information

directories

The ***directories*** option works with both ***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.

Note: UNIX users should use the ***include*** option or place quotation marks around wildcard designations to avoid automatic wildcard expansion by the shell, which may interfere with your pattern search. See "Using Wildcards with PKZIP on UNIX" in Chapter 1.

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:

Sub-option	*To*	*For example*
current	Store the directory path relative to the current location.	***pkzipc -add -directories=current docs.zip docs/**** In this example, only directory information under the docs directory will be stored. Parent directory information will not be stored.
root or full	Store the full path, starting from the root directory down.	***pkzipc -add -directories=root docs.zip docs/**** In this example, the entire directory path, starting from "root" directory will be stored.
specify	Store path information for subdirectories under the specified directories	***pkzipc -add -directories=specify docs.zip temp/docs/**** Stores path information for subdirectories under temp\docs.
relative	Store the directory path relative to the current working directory of the drive specified. (Windows)	***pkzipc -add -directories=relative docs.zip c:*.doc z:*.doc*** 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.
none	Turn off the path option. (Used to override configuration file).	***pkzipc -add -directories=none docs.zip /temp/docs/**** 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 8.

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:

Option	Description	Example
speed	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.	***pkzipc -add -speed test.zip *.doc*** ***pkzipc -add -bzip2 -speed test.zip *.doc***
fast	Provides the second fastest compression: some files are compressed with the Deflate method, using level 2 compression; others* are stored (level 0) uncompressed	***pkzipc -add -fast test.zip *.doc***
maximum	Provides the highest level of compression (level 9)	***pkzipc -add -max test.zip *.doc***
store	Provides zero compression: just stores files inside the archive (level 0)	***pkzipc -add -store test.zip *.doc***
normal (Default)	Provides a middle balance of compression and speed (level 5)	***pkzipc -add -norm test.zip *.doc*** You would only need to use this option if you changed the default compression level. See Chapter 8 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
```

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

```
/usr/local/pkware/pkzipc/*.doc
/usr/local/pkware/pkzipc/pkzip.html
/usr/local/pkware/pkzipc/?????.exe
/*
```

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 *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 9.

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 only 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 Removable Media

span

With PKZIP, you can save your .ZIP or self-extracting archive to removable media 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 (Windows)

On Windows, you can save a ZIP file to multiple removable media if it is too large to fit on a single one. This is called disk *spanning*. PKZIP prompts you to insert removable media as they are needed.

To create a spanned archive:

1. Insert a removable disk (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:

Predefined size	*Comment*
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
```

You can also use ***k***, ***m***, ***g***, or ***t*** to specify split sizes in kilobytes (1024 bytes), megabytes (1024 Kb), gigabytes (1024 Mb), and terabytes (1024 Gb), respectively.

The following command line creates a split archive of 75 Mb:

```
pkzipc -add -span=75m c:\test.zip *.doc
```

To have PKZIP format or wipe removable media before writing to it, use the ***span*** command with ***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.)

Only recent versions of PKZIP/SecureZIP (Server or Windows desktop) will 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`. Use ***archiveeach*** with ***overwrite*** to specify different behavior, such as

```
pkzipc –add –archiveeach -overwrite=never *.xls
```

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 7.

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 (Windows)

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:

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

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
```

Compressing Files Based on File Type (UNIX)

filetype

The ***filetype*** option enables you to include or exclude files by type when adding or extracting files. Specify the type of file in the sub-option. Precede the sub-option with a hyphen to exclude files of that type, or use the sub-option without a hyphen to include such files. For example,

```
… -filetype=-hidden …
```

on the command line excludes hidden files regardless of the default configuration setting. To specify multiple sub-options, separate them with commas.

The following table lists sub-options for file types you can specify with ***filetype***.

Sub-Option	*To*	*For example*
block	Include/exclude block special files. These are files with a mode that begins with a "b" (`brw-------`).	***pkzipc -add -filetype=block test.zip /dev/fd****
char	Include/exclude character special files. These are files with a mode that begins with a "c" (`crw-------`).	***pkzipc -add -filetype=char test.zip /dev/tty****
directory	Include/exclude directory information.	***pkzipc -add -filetype=dir test.zip***
hidden	Include/exclude hidden files. These are files that have a dot (.) in the first position of the file name (`.profile`).	***pkzipc -add -filetype=hid test.zip***
hlink	Include/exclude hard linked files. Hard linked files have a link count greater than one.	***pkzipc -add -filetype=hlink test.zip***
pipe	Include/exclude pipe files. These are files with a mode that begins with a "p" (`prwxr-xr-x`). Adds the pipe specification or definition (name, permissions, times, and so on), not pipe data.	***pkzipc -add -filetype=pipe test.zip***
regular	Include/exclude regular files. These are included by default if no file type is specified.	***pkzipc -add -filetype=regular test.zip***
slink	Include/exclude symbolically linked files. These are files with a mode that begins with a "l" (`lrwxr-xr-x`)	***pkzipc -add -filetype=slink test.zip***
socket	Include/exclude sockets. These are the items that the command `ls -l` lists with an "s" at the beginning of the permissions.	***pkzipc -add -filetype=socket test.zip***
none	Exclude all file types except for those specified on the command line. List file types to include after the *none* sub-option. For example, to include only pipe files: *-filetype=none,pipe* **Note**: If you use the *none* sub-option without listing a filetype to include, all files will be included.	***pkzipc -config -filetype=none,slink***
all	Include/exclude all file types.	***pkzipc -add -filetype=all test.zip***

Following Links (UNIX)

links

PKZIP allows you to follow the UNIX links of a file when compressing files by using the *links* option.

Note: When following links using the link option, the resulting .ZIP archive will be larger since two copies of the file data are compressed as though each link is a separate file. You must also use the filetype option with the links command.

Use the ***links*** option with the following sub-options to process specific file types:

Sub-Option	*To*	*For example*
slink	Symbolic links will be stored (followed) rather than preserved.	***pkzipc -add -links=slink save.zip***
hlink	Hard links will be stored (followed) rather than preserved.	***pkzipc -add -links=hlink save.zip***
none	Symbolic and hard links will be preserved (rather than stored).	***pkzipc -add -filetype=hlink -links=none save.zip***
all	Symbolic and hard links will be stored (followed).	***pkzipc -add -links=all save.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 and `userid`, `groupid`, and UNIX times on UNIX. 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 on DOS systems, or storage of filetype attributes on UNIX systems.

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 running on a UNIX system stores different extended attributes than PKZIP running on a Win32 system. The following table lists the extended attributes that PKZIP stores relative to the UNIX and Win32 operating systems:

UNIX	*Win32*
user ID	create time

UNIX	Win32
group ID	last modification time.
last modification time	last access time.
last access time	.
link information	

Whether, PKZIP overwrites existing files, directories and extended attributes with those stored in the archive when extracting 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 a "comment," containing additional identifying information in your .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:

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

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 *
```

Note: You can also use ***header*** to add a comment to OpenPGP files wrapped in ASCII Armor (see "Encoding an Archive to another Type" in Chapter 7). The comment is displayed when viewing, testing or extracting the archive.

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:

Sub-Option	*To use*	*For example*
retain	The date on which the archive file was created.	***pkzipc -add=update -archivedate=retain test.zip ****
none (Default)	The current date.	***pkzipc -add -archivedate=none test.zip ****
oldest	The date of the oldest file within the archive file.	***pkzipc -add -archivedate=oldest test.zip ****
newest	The date of the newest file within the archive file.	***pkzipc -add -archivedate=newest test.zip ****

Removing File Attributes (Windows)

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:

Sub-Option	*To*	*For example*
hidden	Remove the hidden file attribute from files.	***pkzipc -add -mask=hidden test.zip ****
system	Remove the system file attribute from files.	***pkzipc -add -mask=system test.zip ****

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

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
```

Removing File Attributes (UNIX)

mask

The ***mask*** option specifies a permissions mask for files to be added or extracted. The mask specifies permissions which should *not* be archived or restored on extraction.

On extraction, the ***mask*** option can be used with the ***permission*** option (configured or given on the command line) to explicitly strip permissions specified by that option. (The *setuid*, *setgid*, and *sticky* bits are set on extracted files only if the ***permission*** option is used.)

Use an octal value to specify a permissions mask for the ***mask*** option. For example, the following command line masks write permission for group:

```
pkzipc -add -mask=20 myfiles.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.

Sub-Option	To sort by	For example
date	File date.	***pkzipc -add -sort=date temp.zip***

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

Sub-Option	*Description*
None	Turns shredding off if it is configured on
Random	Overwrites files once with random data (the default)
Dod5220	Overwrites files three times, to the DOD 5220.22-M specification
NSA	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 on a Windows system:

```
The SFX sub-option choices are:

    AIX5X_PPC_C1230 - V12.30 Command Line SFX for AIX on PPC
    DOSJR_X86_C250 - 2.04g compatible SFX Junior for DOS
    DOS_X86_C250   - 2.04g compatible SFX for DOS
    HPUX_ITA_C1230 - V12.30 Command Line SFX for HP-UX on Itanium
    LNX2X_X86_C1230 - V12.30 Command Line SFX for Linux on X86
    SOL2X_SPC_C1230 - V12.30 Command Line SFX for Solaris on SPARC
    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.

Setting the PKSFXSDATA Environment Variable (UNIX)

PKZIP requires the file `pksfxs.dat` to create PKSFX files. Ordinarily, PKZIP searches for this file where it is installed by default, namely, in the directory with the PKZIPC executable.

If you want to keep `pksfxs.dat` in a different location, you can set the environment variable `PKSFXSDATA` to tell PKZIP where to find the file. PKZIP searches for the file first on the path set in the environment variable, second on the current path, and last on a path specified on the command line.

To set the `PKSFXSDATA` environment variable, do the following:

1. Using a text editor such as vi, Pico, Emacs, open your start-up file.
2. What you do next depends on the shell you are using:

- If you are using the Korn Shell (***ksh***) or the Bourne Shell (***sh***), add the following lines to your `.profile` file:

```
PKSFXSDATA=<path to the pksfxs.dat file>
export PKSFXSDATA
```

- If you are using the C Shell (***csh***), add the following line to your `.login` file:

```
setenv PKSFXSDATA <path to the pksfxs.dat file>
```

3. Save and exit the file.
4. To reset your current environment settings, log off your account. The `PKSFXSDATA` variable will be set the next time you log on to your account.

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 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.

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

Note: Some interfaces do not support the full range of self-extracting options. You may want to experiment with sub-options before distributing your self-extracting archive.

Sub-option	*Description*
AutoSFX	Presents a dialog that displays a bar to show progress extracting, and a Cancel button. Supports adding a title and creating subfolders.
EasySFX	(Default) Presents a dialog that enables the user to select a destination folder and to turn off any **runafter** option set. (See "Run Programs with the Self-Extractor," below.) Supports adding a title, creating subfolders, and setting overwrite options.
RegularSFX	Presents a dialog that enables the user to change the destination folder and other options before the archive is extracted. Supports all PKZFX options.

For example:

```
pkzipc -add -sfx=win32_x86_g -sfxuitype=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	keypassphrase (UNIX only)	print (Windows only)
before	larger	silent
console	license	smaller
directories	lowercase	sort
exclude	mask	test
extract	more	times
filetype (UNIX only)	newer	translate
fipsmode	older	version
help	overwrite	warning
id (UNIX only)	passphrase	
include	permission (UNIX only)	

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 8 for information on configuring default sub-option values for commands and options.

Extracting New and Existing Files

You don't have to extract all the files included in a .ZIP archive. You can select files to extract and exclude files you do not need now. 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.

- 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.

Note: Passphrases are case sensitive.

- If you do not specify the ***passphrase*** option when extracting an archive that contains passphrase-protected files, PKZIP prompts you as if you had included the ***passphrase*** command..
- 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.

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 E.

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

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 ***crl*** 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 6.)

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.

Message	*Explanation*	*What to do?*
Signature is invalid	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.
Certificate is not trusted	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.
Certificate is expired	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.
Certificate is revoked	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.

Message	Explanation	What to do?
Certificate not found: XXX	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 or OpenPGP keys. 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 (X.509 and OpenPGP) or a specified LDAP directory (X.509 only)

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. Use the ***ldap*** option (see page 39) to cause PKZIP to search a specified LDAP directory in addition.

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 *cn* 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.	***-verifysigner=cn="John Public"*** ***-verifysigner="John Public"***
Keyid (OpenPGP)	Specify the KeyID of an OpenPGP key; optionally precede the KeyID with 0x	***-archiveType=pgp -verifysigner=kid="XXXXXXXXXXXXXXXX"*** ***-verifysigner=0xXXXXXXXXXXXXXXXX***
Email address	Specify the email address of the certificate (that is, the *e* field in a string representation of a certificate); optionally, precede with: e= 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.	***-verifysigner=e=john.public@xyz.com*** ***-verifysigner=john.public@xyz.com***
LDAP filter	Specify the LDAP filter that you want to use to filter a search for certificates on an LDAP server that you are accessing with the ***ldap*** option; precede with: f= Use quotes if the filter string contains a space. Place the quotes around the entire filter string, including "f=". Include the following LDAP presence filter, as shown in the examples at right, to limit the search to LDAP entries that are certificates: (&(userCertificate=*)(…)) Use standard LDAP filter syntax after the "f=" prefix. This sub-option is for use only when the ***ldap*** option is used.	***-verifysigner=f=(&(userCertificate=*)(ou=Sales))*** ***-verifysigner="f=(&(userCertificate=*)(ou=Regional Sales))"***

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 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.

  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.

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 -

or (UNIX):

cat file.zip | pkzipc -view -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 -
```

or (UNIX):

```
cat file.bz2 | pkzipc -view -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 3 for a way to create an archive that is presented through STDIN.

Extracting an Archive from a Named Pipe, UNIX Domain Socket, or Device File

You can specify a named pipe, UNIX socket, or device file instead of a physical file as the location of an archive from which to extract files. The pipe, socket, or device must first be created, perhaps by another program, and an archive must be written to it. To extract, use the name of the pipe, socket, or device in the command line in place of the name of an archive file. For example:

```
pkzipc -extract -noarchiveextension <name of pipe or socket>
```

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`.

On UNIX, you can use the ***timeout*** option to have PKZIP wait a specified number of seconds before checking a socket or device for more data.

Extracting Data to STDOUT or Special Files

"Adding Data from STDIN or Special Files" in Chapter 3 describes how to capture data from STDIN or such special files as named pipes or UNIX sockets and add it to an archive. Files can also be extracted from an archive to STDOUT or to special files.

To extract to STDOUT, use the ***console*** command. To extract to special files, you can use either ***console*** or ***extract***.

Extracting to STDOUT

To extract to STDOUT, use the ***console*** command instead of ***extract***.

By default, the ***console*** command writes a PKZIP banner of text at the beginning of the output; it also inserts a file header containing the name of the file before each file's data, like this:

```
==============================
output.txt
==============================
```

To suppress the banner, use the ***silent*** option with the ***banner*** sub-option. To suppress the file header as well, append the ***fileheader*** sub-option. The command line below writes only the data from `output.txt`.

```
pkzipc -console -silent=banner,fileheader data.zip output.txt
```

Extracting to Special Files

By redirecting its output, you can also use the ***console*** command to extract data to a named pipe or UNIX socket. The following command line extracts all files in an archive to a UNIX pipe or socket file named `mystream`:

```
pkzipc -console -silent=banner,fileheader data.zip > mystream
```

On Windows, named pipes are all located in the same `pipe` folder on each system: `\\<machine_name>\pipe\`. The following command line uses the ***console*** command to extract file `mystream` and redirects the output from STDOUT to the pipe named `mystream` on the local system:

```
pkzipc -console -silent=banner,fileheader data.zip mystream >
\\.\pipe\mystream
```

The dot (.) indicates the local system. To reference a pipe on another system, specify the machine name:

```
pkzipc -console -silent=banner,fileheader data.zip mystream >
\\boulder\pipe\mystream
```

The ***console*** command has the limitation that all output goes to one place: multiple files cannot be extracted to multiple destinations—different pipes, for example—using a single command line. To extract to one or more special files, use ***extract*** with the ***stream*** (UNIX) or ***rename*** option (Windows) instead of using ***console***.

Extracting to Special Files on UNIX

To extract archived data to a special file on UNIX, use ***extract*** with the ***stream*** option to extract to a pipe or socket file.

The following command line extracts `mystream` either to an existing pipe named `mystream` or, if no such pipe exists, to an ordinary file named `mystream`.

```
pkzipc -extract -stream data.zip mystream
```

If the ***stream*** option is omitted, the file is extracted as an ordinary file, and any existing `mystream` pipe is removed.

The following command line extracts multiple files to multiple pipes:

```
pkzipc -extract -stream data.zip mystream otherstream
```

You can use the ***rename*** option to rename the extracted files to match the names of their intended targets. For example, if `data.zip` contains the files `myinfo` and `otherinfo`, the following command line renames the extracted copies to `mystream` and `otherstream` to enable them to extract to pipes having those names:

```
pkzipc -extract -stream -rename=/info/stream/ data.zip mystream
otherstream
```

The ***rename*** option can be used multiple times in a command line. See “Renaming Files” in Chapter 7.

Extracting to Named Pipes on Windows

To extract to a named pipe on Windows, you must use the ***rename*** option with ***extract***. The ***stream*** option is not used.

On Windows, named pipes are all located in the folder `\\<machine_name>\pipe\`. This path cannot be saved in the archive, but you can specify it on extraction using ***rename***. Extracting a file to an identically named pipe places the data in the pipe and preserves the pipe. A file extracted to a folder that does not contain an identically named pipe file extracts as an ordinary file.

The following example renames a file `myinfo` to match the name of a `\\.\pipe\mystream` pipe on the local system. The ***rename*** option uses the backslash as an escape character even in the replacement expression, so each backslash must itself be preceded by a backslash for PKZIP to see it as a literal character:

```
pkzipc -extract -rename=/myinfo/\\\\.\\pipe\\mystream/ data.zip
mystream
```

See “Renaming Files” in Chapter 7.

Extracting to Dynamically Named Folders

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

Token	Replaced by
{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 ***jobid*** option. For example, if run in 2006: pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc produces a ZIP file named: myJob2006.zip
{mm}	Month, 2-digit
{m}	Month, 1-digit (if possible); 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}\
```

Most UNIX shells treat { and } 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 on UNIX.

Formatted for UNIX, the preceding example looks like this:

```
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 7.

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
```

Changing Ownership When Extracting (UNIX)

id

By default, SecureZIP stores the UID and/or GID of the user who adds a file. When extracting, SecureZIP gives files the UID and GID of the user performing the extraction. The ***id*** option restores files' original ownership when extracting.

```
pkzipc -extract -id=jon test.zip
```

To use the ***id*** option, you must be either the super user or the user listed in the archive as the owner of the file.

The ***id*** option is configurable.

owner

The ***owner*** option can be used when adding or extracting files. The option changes files' associated UID and/or GID to a specified UID or GID. The following example specifies both. It sets the owner to `jon` and the group to `eng`.

```
pkzipc -extract -owner=jon:eng test.zip
```

When adding files, you can use the option to mask off your ownership information. For example, the following command line marks files as owned by the root account.

```
pkzipc -add -owner=0:0 test.zip
```

To use the ***owner*** option when extracting, you must be either the super user or the user listed in the archive as the owner of the file.

When extracting files, you can use the option to mark the files as owned by someone else.

The ***owner*** option is configurable.

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
```

Note: On UNIX systems, no creation time is preserved, as most UNIX file systems do not track when a file was created.

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 (Windows)

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 UNIX and 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.

5 Sending an Archive by FTP and Email

This chapter describes the command line options to transfer a new or existing archive to other people by FTP or email. This functionality requires PKZIP Enterprise or SecureZIP.

Transferring an Archive with FTP

ftp

If your machine has a standard FTP (File Transfer Protocol) program to transfer files over the Internet, you can include an instruction to PKZIP to use the program to send an archive after creating it. For example, the following command lines each create an archive `mydocs.zip` and transfer it to the address specified in the ***ftp*** sub-option. The second example explicitly specifies an FTP user name, passphrase, and account:

```
pkzipc -add -ftp=wash/home/thomas mydocs.zip *.doc
pkzipc -add -ftp=jefferson:monticello:vip@wash/home/thomas
mydocs.zip *.doc
```

The ***ftp*** command/option can be used with the ***add*** command, as in the command lines above, or by itself. When used as a command by itself, ***ftp*** simply transfers the specified file. For example, the following command line transfers existing file `mydocs.zip`:

```
pkzipc -ftp=jefferson:monticello@wash/home/jefferson mydocs.zip
```

Ftp can also be used with the ***delete*** command to transfer an archive after deleting some files in it:

```
pkzipc -delete -ftp=wash/home/jefferson mydocs.zip *.txt
```

You can configure ***ftp*** to use a default address, but you must still include the option on the command line to actually perform an FTP transfer.

```
pkzipc -add -ftp mydocs.zip mydocs.zip *.doc
```

The ***ftp*** address sub-option has the following syntax (optional fields are bracketed).

- To specify a full path on the server:

  -ftp=[*username*[:*passphrase*[:*account*]]@]*server*//*fullpath*

- To specify a relative path on the server, that is, a path relative to the directory that the server chooses for your login:

-ftp=[*username*[:*passphrase*[:*account*]]@]*server*/*relpath*

where:

- *username* (optional) is the user account with which to log in if the FTP server requires a login. If a username is not supplied, PKZIP tries to log in as the user `ftp`.
- *passphrase* (optional) is the passphrase associated with the user account. If no passphrase is given, PKZIP tries an empty passphrase. A colon (:) is not allowed in the passphrase as this character is used to separate username, passphrase, and account values.
- *account* (optional) is for use only with FTP servers that require additional authentication. Do not specify the account for servers that do not require it.
- *server* is the FTP server name
- *path* (relative path or full path; optional) is the path to the destination of the transferred file on the server. If you omit a path, PKZIP transfers the archive to the default folder on the FTP server.

  You can include the ***movearchive*** option to delete from your hard disk an archive that you no longer want after transferring it:

  ```
  pkzipc -add -movearchive -ftp=wash/home/jefferson mydocs.zip *.doc
  ```

  If for some reason an archive is not transferred to the FTP server, ***movearchive*** does not delete it.

Note: The ***ftp*** option can only send ZIP archives that are physical files. It cannot send ZIP archives from STDIN, STDOUT, or special files (named pipes, sockets).

sftp

UNIX users may also use the SSH File Transfer Protocol (***sftp***) to send ZIP archives through a secure channel.

It works exactly as ftp does, as shown in this example:

```
pkzipc -add -sftp=jefferson:monticello:vip@wash/home/thomas
mydocs.zip *.doc
```

Sending an Archive by Email

mailTo, mailFrom, mailServer, mailBCC, mailBody, mailCC, mailOptions, mailReplyTo, mailSubject

You can send a new or existing archive as an email attachment directly from the PKZIP command line. To do so, use the ***mailTo*** option to specify recipients of the message, ***mailFrom*** to give your own address, and ***mailServer*** to list the SMTP server to use to send the message. Other options are available for such other common email-related fields as CC (for recipients to be sent a copy) and BCC (for recipients to be sent a blind copy).

For example, the following command line adds files to archive `data.zip` and emails the archive to John Public as an attachment:

```
pkzipc -add -mailTo=john.public@abc.com -mailFrom=me@myplace.com
-mailServer=smtp.myplace.net -mailSubject="Latest sales" data.zip
*.doc
```

In the following example, ***mailTo*** is used as a standalone command, without ***add***, to send an existing archive:

```
pkzipc -mailTo=john.public@abc.com -mailFrom=me@myplace.com
-mailServer=smtp.myplace.net -mailSubject="Latest sales" data.zip
```

You can include the ***movearchive*** option to delete from your hard disk an archive that you no longer want after emailing it.

Note: The ***mailTo*** command/option can only mail ZIP archives that are physical files. It cannot mail ZIP archives from STDIN, STDOUT, or special files (named pipes, sockets).

Configuring Required Options

To email an archive, each of the three options ***mailTo***, ***mailFrom***, and ***mailServer*** must be specified.

To avoid having to specify these three options on the command line, you can use the ***configuration*** command to configure values for ***mailFrom*** and ***mailServer*** for use by default. Then you need only specify ***mailTo*** on the command line. All the ***mail…*** options are configurable. (To tell PKZIP to mail an archive, you must include ***mailTo*** on the command line even if a value for the option is configured.)

Specifying a Mail Server

The ***mailServer*** option specifies the SMTP server to use. The server specified for ***mailServer*** must be available without a proxy server and must allow email to be forwarded from the machine on which you run PKZIP.

Set the name or IP address of the server into ***mailServer*** as a sub-option. You can either do this on the command line, as in the preceding examples, or you can configure ***mailServer*** to use a specified server by default. For example:

```
pkzipc -config -mailserver=mail.abc.com
```

If necessary, you can specify a user name and/or passphrase. This tells PKZIP to try plain-text or login authentication to connect to the server. Prefix the passphrase with a colon (:), and use an *at* sign (@) to separate user/passphrase information from the server address like this: `user:passphrase@server`. For example:

```
pkzipc -config -mailserver=john:mypassword@mail.abc.com
pkzipc -config -mailserver=:mypassword@mail.abc.com
```

Note the colon before the passphrase.

The following command line creates and sends `data.zip` with the message text specified in ***mailBody***. Set off the message text in quotes:

```
pkzipc -add -mailTo=john.public@abc.com -mailSubject="Latest
sales" -mailBody="Here are the sales figures I promised." data.zip
*.doc
```

Sending to Multiple Recipients

To send an archive to multiple email recipients, use ***mailTo*** multiple times or use it to specify a file that lists recipients. The following command line uses ***mailTo*** multiple times to send to multiple recipients. Each receives a message listing all other recipients who appear in the TO list:

```
pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com
-mailSubject="Latest sales" -mailBody="Here are the sales figures
I promised" data.zip *.doc
```

Sending to a List of Recipients

The ***mailTo*** option can take the name of a list file as a sub-option. In the file, list addresses of recipients one to a line. On the command line, prefix the file name with the ***listchar*** character (`@` by default). The message is sent to every address in the file:

```
pkzipc -add -mailto=@addresses.txt -mailserver=mail01
-mailfrom=sam.adams@wash.com files.zip *.doc
```

Sending Encrypted Attachments

Use ***mailTo*** with its ***recipient*** sub-option to send an archive to the same recipients for whom you encrypt it. For example:

```
pkzipc -add -recipient=tom.jefferson@wash.com
-recipient=sam.adams@wash.com -mailTo=recipient -mailserver=mail01
-mailfrom=sam.adams@wash.com files.zip *.doc
```

The command line above uses the ***recipient*** option to encrypt the archive for specified recipients. It uses ***mailTo*** with the ***recipient*** sub-option to send the archive to those same recipients.

For the ***mailTo recipient*** sub-option to work, the recipients' certificates used to encrypt must contain email addresses. PKZIP alerts you with a warning message for any recipient for whom PKZIP cannot find an email address.

The ***recipient*** sub-option of ***mailTo*** can be used only when ***mailTo*** is used as an option with another command such as ***add***: the ***mailTo recipient*** sub-option cannot be used when ***mailTo*** is used as a standalone command.

If you use the ***recipient*** option (not the ***mailTo recipient*** sub-option) to specify a file that lists the names of certificate holders, you do not need to list recipients on the command line. In this case, using ***mailTo*** with the ***recipient*** sub-option encrypts for, and sends to, all the certificate holders in the list, using the email addresses associated with their certificates.

```
pkzipc -add -recipient=@addresses.txt -mailto=recipient -
mailserver=mail01
-mailfrom=sam.adams@wash.com files.zip *.doc
```

The ***recipient*** option is available only in SecureZIP.

Specifying Text in a File

If the text of the subject or body of the message is more than a few words or contains quotes, you can put the text in a file and specify the file in the sub-option. For example:

```
pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com
-mailSubject=@subject_text.txt -mailBody=@body_text.txt data.zip
*.doc
```

Sending Copies

You can use ***mailCC*** and ***mailBCC***, respectively, to specify recipients to receive copies (CC) and blind copies (BCC) of messages. You can specify recipients' addresses directly, or you can specify a file containing a list of addresses.

Each recipient in the following command line receives a message showing all ***mailTo*** names in the TO list and the ***mailCC*** recipient in the CC list:

```
pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com
-mailSubject="Latest sales" -mailBody="Here are the sales figures
I promised"
-mailCC=rich.smith@abc.com -mailBCC=bill.cody@abc.com data.zip
*.doc
```

To send copies or blind copies to multiple recipients, either use ***mailCC*** or ***mailBCC*** multiple times, or list recipients in a file. Prefix the file name with the list character:

```
pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com
-mailSubject="Latest sales" -mailBody="Here are the sales figures
I promised"
-mailCC=rich.smith@abc.com -mailCC=bill.cody@abc.com
-mailBCC=@address_list.txt data.zip *.doc
```

Sending Split Archives

If you use the ***span*** option with ***mailTo*** to create and mail a split archive, PKZIP sends each segment of the split archive in a separate mail message. This is useful when your recipient's email server has size limits on file attachments on individual emails.

Hiding the TO List

If you do not want recipients to see names of other recipients in the TO list, use the ***mailOptions*** option with either the ***each*** or the ***undisclosed*** sub-option.

The ***each*** sub-option causes each ***mailTo*** recipient to receive a message showing only his own name in the TO list. All ***mailTo*** recipients see all names in the CC list. Any ***mailCC*** and ***mailBCC*** recipients receive a copy of each message to each ***mailTo*** recipient:

```
pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com
-mailOptions=each -mailSubject="Latest sales" -mailBody="Here are
the sales figures I promised" -mailCC=rich.smith@abc.com -
mailBCC=bill.cody@abc.com data.zip *.doc
```

The ***undisclosed*** sub-option works just like the ***each*** sub-option except that the message that each recipient receives displays `Undisclosed` in the TO field instead of the recipient's name.

The ***each*** option causes PKZIP to generate a distinct mail message for each recipient, showing only that recipient's address in the TO field. The ***undisclosed*** sub-option requires PKZIP and the mail server to do less processing and so sends a bit faster.

Including Instructions on How to Unzip

The ***instructions*** sub-option of ***mailOptions*** causes PKZIP to include a small, additional attachment explaining how to unzip a ZIP file.

```
pkzipc -add -mailTo=john.public@abc.com -mailSubject="Plans"
-mailOptions=instructions plans.zip *.doc
```

The ***instructions*** and ***each*** sub-options of ***mailOptions*** can be set together, separated by a comma:

```
… -mailOptions=each,instructions …
```

Using a ReplyTo Address

With the ***mailReplyTo*** option, you can specify an alternate email address for recipients to use to reply to the message instead of the ***mailFrom*** address. For example:

```
pkzipc -add -mailTo=john.public@abc.com -mailFrom=jane.doe@xyz.com
-mailSubject="Plans"  -mailreplyTo=jane.doe@myplace.net plans.zip
*.doc
```

6 Working with Digital Signatures and OpenPGP Keys

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 (version 3) 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) standard. PKZIP will authenticate signatures in OpenPGP files and validated by PGP keyrings 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 Server 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. You can add key usage flags to the public key to designate it for use in encrypting files, authentication of the key's holder, or both.

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 Server uses public-key certificates in PKCS#7 format. The intermediate CA certificates are maintained independently from the ROOT certificates.

Using X.509 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

By and large, a certificate is valid if, and only if, it is trusted, not expired, and not revoked. The table below explains the significance of each possibility:

Status	*What it Means*
Trusted	The certificate can be trusted to indicate who the files really come from
Not Trusted	The certificate cannot be trusted to indicate who the files really come from. The files may not really be signed by the person that the certificate says signed them.
Expired	The current date does not fall within the date range for which the certificate is valid. A certificate is valid only for a certain period. If the files were signed or encrypted while the certificate was valid, there is probably no problem. Otherwise, the certificate should be treated as Not Trusted. (A certificate may also show as expired if the date on your computer is incorrect.)
Not Expired	The current date is within the valid range of dates for the certificate

Status	What it Means
Time nested	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.
Not time-nested	The period of validity of the certificate extends past the dates when the issuer certificate is valid. This condition does not necessarily mean that the selected certificate is fraudulent. For example, it may inadvertently have been given too long a period of validity when it was issued. On the other hand, if the issuer certificate was expired when the selected certificate supposedly begins to be valid, the situation may be worth investigating more closely.
Revoked	The certificate has been canceled by either the issuer or the owner. A certificate might be revoked for a variety of reasons: Perhaps the owner lost the private key or someone else gained access to it; perhaps the issuer determined that the certificate was fraudulently obtained. In general, you should not trust files that were signed with a Revoked certificate.
Not Revoked	The certificate has not been revoked

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:

Option	Description
KeyUsage	Check the purpose for which the certificate is designated (encryption or signing).
TimeNesting	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.
TimeValid	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:

```
(W76) 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. On UNIX, use the PKCertTool utility to import a CRL. (See "PKCertTool Commands and Options" later in this chapter.)

Making an Online Certificate Status Protocol (OCSP) Request

The Internet Engineering Task Force (IETF) created the Online Certificate Status Protocol (OCSP) as an alternative method for determining whether a certificate is valid. Many Certificate Authorities maintain OCSP responders that allow you to check the validity of a certificate.

To test whether a certificate signing a ZIP archive is valid using OCSP, type:

```
pkzipc -test -crl=ocsp signed.zip
```

The OCSP responder will reply to this request with `good`, `revoked` or `unknown`.

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 Import 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 Mozilla Network Security Services (NSS) certificate stores. For PKZIP to access a certificate that you used the Mozilla Firefox browser to install, you must export the certificate from Firefox 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.

Setting Up Stores for Digital Certificates on UNIX/Linux

Unlike Windows, UNIX and Linux do not have a single standard facility for storing digital certificates or a standard way to import certificates and convert them into a form that PKZIP can use. To address this, PKZIP provides a utility program—PKCertTool—to set up and manage certificate stores on UNIX/Linux for use with PKZIP.

Note: To digitally sign files or to enable other people to strongly encrypt files specifically for you as a recipient, you need SecureZIP and your own personal digital certificate or an organizational certificate such as an SSL certificate. Visit the PKWARE Web site for information on the type of certificate you need (RSA format, 1024-bit minimum) and how to get one.

Setting Up the Certificate Stores

The PKWARE utility PKCertTool sets up the PKZIP certificate stores and imports certificates into them, along with any certificate revocation lists you want PKZIP to use.

A certificate revocation list (CRL) is a list of certificates that have been revoked by their issuing certificate authority. Before using a certificate, PKZIP can check CRLs in its stores to be sure the certificate is not listed as revoked. (See the ***crl*** option.) CRLs are published by certificate authorities and can be included in files that contain certificates, though this is rare in practice.

PKCertTool sets up the following certificate stores:

Store	*Description*
ROOT	A store for certificates used to validate other certificates. These certificates are “trusted” by the users of the system. They are the certificates at the top of a certificate chain and do not derive from any “more trusted” antecedent certificates.
CA	*CA* stands for *Certificate Authority*. Certificates in this store are used to validate other certificates. The certificates generally do not belong to particular users and are not used for encryption or authentication. They are intermediate certificates in a certificate chain that derives from some root certificate. They enable a certificate to be traced back to its root.
AddressBook	A store for certificates used to encrypt files for other people. Certificates in this store contain only public keys; they do not contain private keys.

Store	Description
MY	A store for personal certificates with their respective private keys. Private keys are used to sign files and to decrypt files encrypted specifically for the user with the associated public key. Each user should have his own personal store, accessible only to him, to ensure that only that user can use a certificate's private key to sign or decrypt files. (Private keys in the MY store are encrypted using PKCS#8 format and PKCS#5 version 2.)

PKCertTool sets up the certificate stores as tables in a SQL database. PKCertTool creates databases and tables as necessary and manages all database interactions. You just need to specify certificates and keys and the stores to put them in. If you add a certificate without specifying a store, PKCertTool determines the appropriate store for you, based on the certificate.

See "Migrating Certificates from a PKZIP 6.x Store," later in this chapter, if you have certificates in a PKZIP version 6 certificate store.

PKZIP and PKCertTool can import certificates, CRLs, and keys in the following file formats:

Format	Description
PEM	Contains one or more certificate(s) and/or private key(s). Can also contain a CRL. Common file extensions: `.pem`, `.cer`, `.key`
PKCS#12	Can contain one or (in theory) more certificates and both their public and private keys. In practice, a PKCS#12 file usually contains only multiple certificates with a single private key. Common file extensions: `.pfx`, `.p12`
PKCS#7	Can contain one or more certificates and their public keys and CRLs; does not contain private keys. Common file extensions: `.p7`, `.p7b`, `.p7c`

You must use the ***add*** command to tell PKCertTool what certificates and keys to import. PKCertTool copies the existing certificates and keys from their specified location and adds them to the appropriate stores. If the stores do not already exist where you tell PKCertTool to look for them, PKCertTool automatically creates the necessary database and/or tables.

Setting Up Stores for One User or Shared Environments

In most cases, you will be running PKZIP Server on your system, with all your relevant certificate stores in your `$HOME` directory. In some situations, you may need to access a certificate store that exists in a shared directory on your system. You can use PKCertTool to set up these stores or recognize them for use with PKZIP.

To create a single certificate store in `$HOME`, run ***PKCertTool*** with your end-user account. PKCertTool will create `$HOME/certificates.db`.

If you have certificates stored in a shared directory, the system administrator should set up environment variables that point the system to the appropriate certificate store(s). The system administrator can also run PKCertTool to create a database containing the ROOT, AddressBook, and CA certificate stores as shared stores, accessible to all users. See the sections, "Shared or Separate AddressBook Stores"

and “Setting Environment Variables for Certificate Stores” later in this chapter for more information.

Certificates that include a private key can be safely added to the AddressBook store: PKZIP does not add private keys to, or read private keys from, the AddressBook store. PKZIP adds private keys only to the MY store.

After the shared stores are created, each user must run PKCertTool to create a database and set up a MY store in his $HOME directory for his personal certificates and their private keys.

Once certificate stores are set up, PKCertTool needs to be run again only to add new certificates and CRLs. PKZIP accesses certificates in the stores as needed.

Shared or Separate AddressBook Stores

You can set up a single, shared AddressBook store for multiple users, or different AddressBook stores for different users.

If certificates containing public keys are placed in a shared AddressBook store, all users can access them. Alternatively, users can use PKCertTool to create their own AddressBook stores in the same (unshared) database with their MY store. Other users cannot access public key certificates in this AddressBook store.

PKZIP uses the first store (of the appropriate type) that it finds. If a user points PKZIP to an unshared AddressBook store, that is the only AddressBook store that PKZIP searches.

A user who wants to add a certificate to a shared AddressBook store for other users to access should ask the administrator of the shared store to add the certificate.

Locating Certificate Store Databases

You can specify your own names for certificate database files, and you can locate the databases anywhere you want. By default, PKCertTool names any certificate database it creates `certificates.db`.

If you do not tell PKCertTool where to look for a certificate database when you add certificates, PKCertTool will take the following actions, in order:

1. Search locations (if any) specified by the following environment variables:

- `$ROOT_CERTIFICATES` for the database containing the ROOT store
- `$CA_CERTIFICATES` for the database containing the CA store
- `$ADDRESS_BOOK_CERTIFICATES` for the database containing the AddressBook store
- `$MY_CERTIFICATES` for the database containing the MY store

See the section “Setting Environment Variables for Certificate Stores,” later in this chapter.

2. Locate `$HOME/certificates.db`, where `$HOME` is the user’s home directory.
3. Create `$HOME/certificates.db`.

PKCertTool Commands and Options

PKCertTool is a separate program from PKZIP. It has the following commands for operating on certificates:

- ***add*** Add certificates and certificate revocation lists (CRLs) to a store
- ***list*** List information about certificates and CRLs

- ***del*** Delete certificates and CRLs
- ***keys*** List public key hashes for all private keys in a certificate store database
- ***export*** Export certificates and CRLs
- ***view*** Display information about the certificates and CRLs in a certificate file.
- ***upgrade*** Upgrade certificate store database

Each PKCertTool command can be used with one or more options. Most of the options are not required. When entering a PKCertTool command or option, prefix it with a hyphen in the command line as you do with a PKZIP command or option.

The PKCertTool *add* Command

Usage:

```
pkcerttool -add [-store <store name>] [-database <database
   file>] [-all] [-f <friendly name>] [-passin <passphrase>]
   [-passout <passphrase>] <certfile> [<keyfile>]
```

Command / Option	*Description*
-add	Tells PKCertTool to add certificates and any CRLs signed by those certificates to a store. A CRL in a PKCS#7-format file is added automatically if the certificate used to sign it is added unless a newer CRL for that certificate already exists in the store. CRLs are added to the same store as the accompanying certificates used to sign them.
-store <store name>	The store to which to add certificates. Valid store names are: MY AddressBook ROOT CA If no store is specified, PKCertTool picks the most appropriate one based on the certificate. End-entity certificates in PKCS#7 files and in PEM files that do not include private keys are added to the AddressBook store. CA certificates in such files are added to the Root store if they are self-signed or to the CA store if they are not. NOTE: PKWARE recommends using a PKCS#7 file to add certificates to the AddressBook store.
-database <database>	The location of the database containing the store to use. If this parameter is omitted, PKCertTool uses the first database (that PKCertTool can write to) found by searching the places listed above in the section “Locating Certificate Store Databases.”
-all	Adds all certificates in the file specified in the <certfile> argument. Also adds all CRLs in a PKCS#7 file except that • An older CRL is not added if a newer CRL already exists in the store • A CRL is not added if its signature cannot be validated If this parameter is omitted, PKCertTool adds only the first certificate found in the file.

Command / Option	Description
<certfile>	The location of a file containing one or more certificates to add. The location can be either a file that contains certificates or a directory that contains such files. A certificate file can be in PEM, PKCS#7, or PKCS#12 format. Multiple certificates can be added from a single file. When the name of a directory, PKCS#7 file, or PEM file is used, PKCertTool adds only the first certificate found if -all is not specified. With certificates in a PKCS#12 file, PKCertTool adds only the end-entity certificate if -all is not specified.
<keyfile>	The path name of a PEM file that contains the private key for a certificate in a PEM format <cert file>. Use <keyfile> to specify the location of a private key if the <certfile> does not itself contain this key. <keyfile> cannot be used with -all or with a PKCS#7-format <certfile>: a PKCS#7 file cannot contain a private key.
-f <friendly name>	Sets a friendly name for a certificate—for example, *My 2003 Cert*. A friendly name can be used to reference a certificate in the <certfile> argument of the PKCertTool -list and -del commands. A friendly name is useful to distinguish multiple certificates that have the same common name. For example, you might give a different friendly name to a newer, renewed personal certificate to tell it from older, expired certificates that you keep to decrypt files encrypted using those certificates. The friendly name is added only to the first certificate in a file. In PKZIP, a friendly name that you assign with PKCertTool can be used with the PKZIP -recipient and -certificate options.
-passin <passphrase>	The passphrase used to decrypt a private key specified by <keyfile> or used to decrypt a PKCS#12 file specified by <cert file>. If -passin is not used, PKCertTool prompts for a passphrase needed to decrypt a private key.
-passout <passphrase>	The passphrase to use to encrypt a private key when it is stored in the certificates database. If -passout is not used, PKCertTool prompts for a passphrase to use to encrypt a private key.

Examples

The following command line adds all certificates and accompanying CRLs to the AddressBook store:

```
pkcerttool -add -all -store AddressBook mycerts.p7c
```

The following command line adds the certificate from `mycert.pfx` to the store. It uses `MyPassPhrase` to decrypt the PKCS#12 file, and it uses `MyStorePassPhrase` to encrypt the private key in the store:

```
pkcerttool -add -passin MyPassPhrase -passout MyStorePassPhrase
mycert.pfx
```

The PKCertTool *list* Command

Usage:

```
pkcerttool -list [-store <store name>] [-database <database
   file>] [-v] [-pemout] [-passin <passphrase>] [-crl]
   [-thumbprint <thumbprint>] [<certificate name>|<certfile>]
```

Command / Option	Description
-list	Lists information about the certificates and CRLs in the specified store
-store <store name>	The store for which to list certificates. Valid store names are: MY AddressBook ROOT CA If this argument is omitted, PKCertTool lists certificates in the MY store
-database <database>	The location of the database containing the store to list. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section "Locating Certificate Store Databases."
<certificate name>	The common name of a certificate to list. Alternatively, you can use an email address contained in the certificate, or a certificate's friendly name as set with the -f option of the -add command. Any of these can be used instead of <certfile> to reference a certificate.
<certfile>	The location of a file containing one or more certificates. A certificate file can be in PEM, PKCS#7, or PKCS#12 format. The -list command lists information about the PKZIP store copy of the *first* certificate listed in the file. (Use the PKCertTool -view command to see which certificate is listed first.)
-v	Displays a *verbose* (more detailed) listing of information about the certificate

-crl	Lists any certificate revocation lists in the store. If used with the -v option, produces a display like this:
	`--- CRL  1 ---`
	`<Issuer Certificate Name>`
	`Last Update:`
	`  Thu Oct 14 09:42:33 2004`
	`Next Update:`
	`  Sat Oct 15 09:42:33 2005`
	`Version: <version number>`
	`Revoked Serial Numbers (<count of serial number>):`
	`  <First serial number>`
	`  <Second serial number>`
	`----------`
	`  1 CRL`
	If -crl is used without the -v option, the display omits the CRL version number and the serial numbers of revoked certificates.

Command / Option	Description
-thumbprint <thumbprint>	Identifies a particular certificate by its *thumbprint*, that is, by the value listed as *SHA-1 Hash of Certificate* when the certificate is viewed in PKCertTool using the -view or -list command with the -v option. (See example after this table.) The -thumbprint option is useful to distinguish certificates that have the exact same common name and friendly name in a store. Such identical listings can come about when, for example, root and CA certificates are imported from a Windows system, or when a renewed personal certificate is installed without specifying a unique friendly name. A thumbprint is not case-sensitive and can be truncated. You need enter only enough of the thumbprint to match the certificate(s) you want. The -list command lists all certificates matching both a specified thumbprint and specified common or friendly name. A thumbprint can be entered with or without spaces. Set off the thumbprint with quotes if it contains spaces. For example: pkcerttool -list -v -thumbprint "25 28 Y0 YY" "John J. Adams" or: pkcerttool -list -v -thumbprint 2528Y0YY "John J. Adams"
-passin <passphrase>	The passphrase used to decrypt a PKCS#12 file specified by <certfile>. If -passin is not used, PKCertTool prompts for a passphrase needed to decrypt a private key.
-pemout	Prints out the certificate(s) in PEM format

Example

The following command line gives a verbose listing of certificates in the MY store:

```
   pkcerttool -list -v

-----------
MY
-----------

--- Certificate     1 ---
John J. Adams
Subject:
   O=VeriSign, Inc.
   OU=VeriSign Trust Network
   OU=www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98
   OU=Persona Not Validated
   OU=Digital ID Class 1 - Microsoft Full Service
   CN=John J. Adams
   E=john.adams@acme.com
Issuer:
   O=VeriSign, Inc.
   OU=VeriSign Trust Network
   OU=www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98
   CN=VeriSign Class 1 CA Individual Subscriber-Persona Not
Validated
SerialNumber:
   1M M0 UJ 41 H3 24 U3 J3 8J 42 YH JJ 77 Y0 65 3H
NotBefore:
   Tue Sep  4 19:00:00 2001
NotAfter:
   Thu Sep  5 18:59:59 2002
SHA-1 Hash of Certificate:
   25 28 Y0 YY 37 YU J9 O1 6J YY 9Y 1H 79 0J 97 2Y
   1M 73 23 Y2
Public Key Hash:
   31 Y2 98 Y3 76 PU U2 J3 HH 25 U6 PY 9Y 6J O3 0U
   73 61 1H 8M
```

```
--- Certificate     2 ---
Thawte Freemail Member
Subject:
   CN=Thawte Freemail Member
   E=todda@acme.com
Issuer:
   C=ZA
   S=Western Cape
   L=Durbanville
   O=Thawte
   OU=Certificate Services
   CN=Personal Freemail RSA 1999.9.16
SerialNumber:
   2F 52 03
NotBefore:
   Thu Sep  7 12:26:30 2000
NotAfter:
   Fri Sep  7 12:26:30 2001
SHA-1 Hash of Certificate:
   M3 L6 J5 PM L4 74 M1 15 P6 PL 22 J3 11 94 30 L7
   LP 9Y 85 J6
Public Key Hash:
   01 6J 30 JJ 8Y 12 P5 18 YJ 9P 7U 8H 52 MP PY 94
   P4 81 4H 42

------------------
    2 certificates
```

The following command line gives a verbose listing of the certificates matching both the specified certificate name and thumbprint and in the MY store:

```
pkcerttool -list -v -thumbprint 2528Y0YY "John J. Adams"

-----------
MY
-----------

--- Certificate     1 ---
John J. Adams
Subject:
   O=VeriSign, Inc.
   OU=VeriSign Trust Network
   OU=www.verisign.com/repository/RPA Incorp. by Ref.,LIAB.LTD(c)98
   OU=Persona Not Validated
   OU=Digital ID Class 1 - Microsoft Full Service
   CN=John J. Adams
   E=john.adams@acme.com
Issuer:
   O=VeriSign, Inc.
   OU=VeriSign Trust Network
   OU=www.verisign.com/repository/RPA Incorp. By Ref.,LIAB.LTD(c)98
   CN=VeriSign Class 1 CA Individual Subscriber-Persona Not
Validated
SerialNumber:
   1M M0 UJ 41 H3 24 U3 J3 8J 42 YH JJ 77 Y0 65 3H
NotBefore:
   Tue Sep  4 19:00:00 2001
NotAfter:
   Thu Sep  5 18:59:59 2002
SHA-1 Hash of Certificate:
   25 28 Y0 YY 37 YU J9 01 6J YY 9Y 1H 79 0J 97 2Y
   1M 73 23 Y2
Public Key Hash:
   31 Y2 98 Y3 76 PU U2 J3 HH 25 U6 PY 9Y 6J 03 0U
   73 61 1H 8M

------------------
    1 certificates
```

The following command line uses the `pemout` option to print the certificates in the root store in PEM format:

```
hrvsrv-szgw2# pkcerttool -list -store root -pemout

PKCertTool(tm) Version 1.40
Portions copyright (C) 2001-2008 PKWARE, Inc. All Rights Reserved.
Build Version ($BuildRev: 1025 $)

------------------------------------------------------------------
Certificates in ROOT in the database: /usr/local/certificates.db
------------------------------------------------------------------
-----BEGIN CERTIFICATE-----
RJJWAWLLAkynAwJBAnJVAOP4uB6+R/698+StFaSh/pSotfTbRAOGLSqGSJb3WQEB
BQUARLJxJWAeBnNVBARTF29tYXNyWJ1AeRW3RJ5wa3WhLRUuY29tRB4XWTA4RTJx
RjJyRjLORVoXWTA5RTJxRjJyRjLORVowJjEnRB4GA1UEAxRXb21hL3J2LXN6A3Ly
LnBrW2FyAS5jb20wnnEJRAOGLSqGSJb3WQEBAQUAA4JBWwAwnnEKAoJBAQLkLOOk
49RrSJvRxnkuGPtRj8+WUYAklFbT9A4OTPo8NLaqSWRGQtsUsLGy+SBLuEeqYxoW
Or6TR4n4ThwHrWHnpyN/30Y/+JpJPOGU2roR+qUAnRX6RKJ/4keHP+huLnOPAOo6
LwNUJKuLpwx7nuXW3HljV6lRbnl4nVAWJAYJTRnlqUrrnRql5bAGTxBtHX8R13XY
nR2bbaUytX4aRRLVTXonkpsOXHVFFGuJOWLnOGLLo1/qxVRqVo5RARRoSJKfoEF+
8QnpGbLOG2WjL+Sq1WyY2onSqnuL+u35JQlJ49Xrw73JAbrAJb8vlHnLTteN4tL+
LejBLfAWKJKLhVtBAnRBAAGjnAAwnYOwHQYWVROOBBYEFFy5WQbqLwBwULvH5qqP
q8vaQtSURF4GA1UWJwRXRFWAFFy5WQbqLwBwULvH5qqPq8vaQtSUoSakJWAJRSAw
HnYWVQQWExWvbWFALnYtL3pnWAJuLGt3YXJlLRNvbYJVAOP4uB6+R/698+StFaSh
/pSotfTbRAwGA1UWEwEB/wQLRAAwWQYJKoAJhvLNAQEFBQAWnnEBAJQyOJLLWulG
1EARv+YetAQLAB14jR2LJh+rUV/TRLO3L5Y8Sunj7Epq19JPnOV4nLRAKASYrFNp
6epA4n+qbtAayap3y8Qf6WT3B+/YnN9rQjRLyvGAwJpXbKbBLRxwNeuBrt7R4fSJ
UK+2jyfnuAAsRB7TWntK6XLoPnHYJ9t9vkru6Wq+OvR1SHQA56Oqj0yAlA+HR1n+
OErLe+kWGWY9lQER+UnsYtVfanHAyrNXvAE8UoUyo2pJRQLjNAoAHNJHahs7WRNR
y6VJR4eLRkQ4OOeASR8p7J6KpbbpbrKrTJAoRRBNUuQ5AuFleYsAVpU39qRwRwrt
1WNuqLeehQA=
-----END CERTIFICATE-----
```

The PKCertTool *del* Command

Usage:

```
pkcerttool -del [-store <store name>] [-database <database
   file>] [-passin <passphrase>] [-crl] [-thumbprint
   <thumbprint>] <certificate name>|<certfile>
```

Command / Option	*Description*
-del	Deletes a specified certificate and any associated CRL from a PKZIP store. If a specified certificate has an associated CRL—that is, has been used to sign a CRL—the CRL is deleted with the certificate. If for some reason the CRL cannot be deleted, the certificate is not deleted either.
-store <store name>	The store from which to delete a certificate. Valid store names are: MY AddressBook ROOT CA If this argument is omitted, PKCertTool looks for the certificate in the MY store. PKCertTool warns if the certificate is not found.
-database <database>	The location of the database containing the store from which to delete a certificate. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section “Locating Certificate Store Databases.”

Command / Option	Description
-passin <passphrase>	The passphrase used to decrypt a PKCS#12 file specified by <certfile>. If -passin is not used, PKCertTool prompts for a passphrase needed to decrypt a private key.
-thumbprint <thumbprint>	Identifies a particular certificate by its *thumbprint*, that is, by the value listed as *SHA-1 Hash of Certificate* when the certificate is viewed in PKCertTool using the -view or -list command with the -v option. (See example below.) The -thumbprint option is useful to distinguish certificates that have the exact same common name and friendly name in a store. Such identical listings can come about when, for example, root and CA certificates are imported from a Windows system, or when a renewed personal certificate is installed without specifying a unique friendly name. A thumbprint is not case-sensitive and can be truncated. With -del, if a truncated thumbprint matches multiple certificates, the first certificate found is selected. A thumbprint can be entered with or without spaces. Set off the thumbprint with quotes if it contains spaces. For example: pkcerttool -del -thumbprint "25 28 Y0 YY" "John J. Adams" or: pkcerttool -del -thumbprint 2528Y0YY "John J. Adams"
<certificate name>	The common name of a certificate to delete. Alternatively, you can use an email address contained in the certificate, or a certificate's friendly name as set with the -f option of the -add command. Any of these can be used instead of <certfile> to reference a certificate.
<certfile>	The location of a file containing one or more certificates. A certificate file can be in PEM, PKCS#7, or PKCS#12 format. The -del command deletes the PKZIP store copy of the *first* certificate listed in the file. (Use the PKCertTool -view command to see which certificate is listed first.) The -del command does not delete certificates from files or delete the files themselves.
-crl	Deletes from the store *only* a CRL signed by the specified certificate; does not delete the certificate itself.

Example

The following command line deletes the certificate for John J. Adams from the AddressBook store:

```
pkcerttool -del -store AddressBook "John J. Adams"
```

The following command line deletes from the AddressBook store a CRL signed by the certificate identified by *<issuer name>*. Only the CRL is deleted, not the certificate:

```
pkcerttool -del -store CA <issuer name> -crl
```

The PKCertTool *keys* Command

Usage:

```
pkcerttool -keys [-database <database file>]
```

Command	Description
-keys	Lists the public key hashes of all private keys in a certificate store database.

Command	Description
	This command is for troubleshooting. You can compare its output to the output of -list -v to find the certificates for which you have private keys.
-database <database>	The location of the database. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section "Locating Certificate Store Databases."

Example

```
   pkcerttool -keys
-----------
Private keys in /home/george/certificates.db
-----------

--- PrivateKey     1 ---
   01 6J 30 JJ 8Y 12 P5 18 YJ 9P 7U 8H 52 MP PY 94
   P4 81 4H 42

--- PrivateKey     2 ---
   1M M0 UJ 41 H3 24 U3 J3 8J 42 YH JJ 77 YO 65 3H
   2U 8P 20 YY

--- PrivateKey     3 ---
   31 Y2 98 Y3 76 PU U2 J3 HH 25 U6 PY 9Y 6J 03 OU
   73 61 1H 8M
------------------
    3 private keys
```

The PKCertTool *export* Command

Usage:

```
pkcerttool -export [-store <store name>] [-database <database
   file>] [-passin <passphrase>] [-passout <passphrase>]
   [-crl] [-thumbprint <thumbprint>] [-all|<certificate name>]
   <output file>
```

Command	Description
-export	Exports certificates and CRLs from a specified store.
-store <store name>	The store from which to export certificates. Valid store names are: MY AddressBook ROOT CA If this argument is omitted, PKCertTool exports certificates from the MY store
-database <database>	The location of the database. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section "Locating Certificate Store Databases."
-passin <passphrase>	The passphrase to decrypt a private key in the store so that it can be exported. For use when exporting a certificate with its private key. Cannot be used with -all.

Command	Description
-passout <passphrase>	The passphrase to use to encrypt a private key in an exported PKCS#12 file. This is the passphrase that a user must supply to access the key in the exported PKCS#12 file. Cannot be used with -all.
-crl	Includes any CRL issued by the certificate when a single certificate is specified. Cannot be used with -all.
-thumbprint <thumbprint>	Identifies a particular certificate by its *thumbprint*, that is, by the value listed as *SHA-1 Hash of Certificate* when the certificate is viewed in PKCertTool using the -view or -list command with the -v option. (See example below.) The -thumbprint option is useful to distinguish certificates that have the exact same common name and friendly name in a store. Such identical listings can come about when, for example, root and CA certificates are imported from a Windows system, or when a renewed personal certificate is installed without specifying a unique friendly name. A thumbprint is not case-sensitive and can be truncated. With -export, if a truncated thumbprint matches multiple certificates, the first certificate found is selected. A thumbprint can be entered with or without spaces. Set off the thumbprint with quotes if it contains spaces. For example: ***pkcerttool -export -thumbprint "25 28 Y0 YY" "John J. Adams"*** or: ***pkcerttool -export -thumbprint 2528Y0YY "John J. Adams"***
-all	Exports all certificates and CRLs in the store
<certificate name>	The name of the certificate to export, in quotes. Cannot be used with -all.
<output file>	The name to give the output file. For information on the file format used, see the notes following this table.

Output file format: The format of the output file to which certificates are exported is determined by the following rules:

- If the ***-all*** or ***-crl*** option is used, the output file is saved in PKCS#7 format. No associated private key is exported even if one is present.
- If a single certificate that *does not* have a private key is specified (and the ***-crl*** option is not used), the certificate is saved in DER format.
- If a single certificate that *does* have a private key is specified (and the ***-crl*** option is not used), then the private key is exported and the output file is saved in PKCS#12 format if you specify a file name that has the extension `.p12` or `.pfx`. or no “dot” extension at all. Otherwise, the private key is *not* exported, and the output file is saved in DER format.

Examples

The following command line exports all certificates and CRLs in the MY store to a PKCS#7 file; no private keys are exported.

```
pkcerttool -export -all mycerts.p7
```

The following command line exports from the CA store the “GTZ Cybertrust” certificate, with any associated CRL from the same store, to a PKCS#7 file; no private key is exported.

```
pkcerttool -export -store CA "GTZ Cybertrust" -crl
cybertrust_ca.p7
```

The following command line exports from the MY store a personal certificate, with its private key, to a PKCS#12 file.

```
pkcerttool -export "George Washington" mycert.p12
```

The following command line exports from the AddressBook store a single certificate that has no private key to a raw CRL-format file.

```
pkcerttool -export -store addressbook "John Q. Public"
johnpublic.crl
```

The PKCertTool *view* Command

Usage:

```
pkcerttool -view [-ca] [-endentity] [-v] [-pemout]
   [-passin <passphrase>] <certfile>
```

Command	*Description*
-view	Displays information about the certificates and CRLs in a certificate file. The information is similar to that produced by the PKCertTool -list command.
-ca	A filter to view only certificate authority (CA) certificates
-endentity	A filter to view only end entity certificates
-v	Displays a *verbose* (more detailed) listing of information about the certificates
-passin <passphrase>	The passphrase to decrypt a private key in a PKCS#12 certificate file so that it can be viewed
<certfile>	The location of a file containing one or more certificates. The certificate file can be in PEM, PKCS#7, or PKCS#12 format.
-pemout	Prints out the certificate(s) in PEM format

Examples

The following command line displays a verbose listing of information about the certificates and CRLs in a PKCS#7 file:

```
pkcerttool -view -v mycerts.p7
```

The following command line displays a verbose listing of information about the certificate in a PKCS#12 file:

```
pkcerttool -view -v -passin "mypassphrase" my_personal_certs.p12
```

The following command line prints out certificate `hrvsrv-szgw2.pkware.com.cert` in PEM format:

```
hrvsrv-szgw2# pkcerttool -view -endentity -pemout
/etc/partnergw/hrvsrv-szgw2.mycompany.com.cert
```

The output looks like this:

```
PKCertTool(tm) Version 1.40
Portions copyright (C) 2001-2008 PKWARE, Inc. All Rights Reserved.
Build Version ($BuildRev: 1025 $)

-----BEGIN CERTIFICATE-----
RJJWAWLLAkynAwJBAnJVAOP4uB6+R/698+StFaSh/pSotfTbRAOGLSqGSJb3WQEB
BQUARLJxJWAeBnNVBARTF29tYXNyWJ1AeRW3RJ5wa3WhLRUuY29tRB4XWTA4RTJx
RjJyRjLORVoXWTA5RTJxRjJyRjLORVowJjEnRB4GA1UEAxRXb21hL3J2LXN6A3Ly
LnBrW2FyAS5jb2OwnnEJRAOGLSqGSJb3WQEBAQUAA4JBWwAwnnEKAoJBAQLkLOOk
49RrSJvRxnkuGPtRj8+WUYAklFbT9A4OTPo8NLaqSWRGQtsUsLGy+SBLuEeqYxoW
Or6TR4n4ThwHrWHnpyN/3OY/+JpJPOGU2roR+qUAnRX6RKJ/4keHP+huLnOPAOo6
LwNUJKuLpwx7nuXW3HljV6lRbnl4nVAWJAYJTRnlqUrrnRql5bAGTxBtHX8R13XY
nR2bbaUytX4aRRLVTXonkpsOXHVFFGuJOWLnOGLLo1/qxVRqVo5RARRoSJKfoEF+
8QnpGbLOG2WjL+Sq1WyY2onSqnuL+u35JQlJ49Xrw73JAbrAJb8vlHnLTteN4tL+
LejBLfAWKJKLhVtBAnRBAAGjnAAwnY0wHQYWVR0OBBYEFFy5WQbqLwBwULvH5qqP
q8vaQtSURF4GA1UWJwRXRFWAFFy5WQbqLwBwULvH5qqPq8vaQtSUoSakJWAJRSAw
HnYWVQQWExWvbWFALnYtL3pnWAJuLGt3YXJlLRNvbYJVAOP4uB6+R/698+StFaSh
/pSotfTbRAwGA1UWEwEB/wQLRAAwWQYJKoAJhvLNAQEFBQAWnnEBAJQy0JLLWulG
1EARv+YetAQLAB14jR2LJh+rUV/TRLO3L5Y8Sunj7Epq19JPnOV4nLRAKASYrFNp
6epA4n+qbtAayap3y8Qf6WT3B+/YnN9rQjRLyvGAwJpXbKbBLRxwNeuBrt7R4fSJ
UK+2jyfnuAAsRB7TWntK6XLoPnHYJ9t9vkru6Wq+0vR1SHQA56Oqj0yAlA+HR1n+
0ErLe+kWGWY9lQER+UnsYtVfanHAyrNXvAE8UoUyo2pJRQLjNAoAHNJHahs7WRNR
y6VJR4eLRkQ4O0eASR8p7J6KpbbpbrKrTJAoRRBNUuQ5AuFleYsAVpU39qRwRwrt
1WNuqLeehQA=
-----END CERTIFICATE-----
```

The PKCertTool *upgrade* Command

Usage:

```
pkcerttool -database <database>
```

Command	*Description*
-database <database>	The location of the database. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section “Locating Certificate Store Databases.”

Example

The following command upgrades your default certificates database file to be in the new format, and save a copy of the original as `$HOME/certificates.db.old`.

```
pkcerttool -database
```

Exit Codes for PKCertTool

PKCertTool generates the exit codes described in the table below.

Code	*Meaning*
0	Success
1	Unknown option
2	Out of memory
3	Missing option or argument. For example, -f is used but no friendly name is supplied; or -add is used but no cert file is given.
4	Invalid database specification. A database specified (with a command other than -add) does not exist, or a directory is specified in place of a database.
5	Conflicting commands. For example, specifying both -add and -del.
6	Action failed. For example, PKCertTool cannot open a store, or cannot

	list a specified certificate, or cannot find any certificates to list.

Setting Environment Variables for Certificate Stores

PKZIP checks the environment variables listed below for the locations of certificate stores. If these variables are set, PKZIP uses the certificate databases specified by the variables instead of looking in the default location.

Environment Variable	*Certificate Store*
$ROOT_CERTIFICATES	ROOT
$CA_CERTIFICATES	CA
$ADDRESS_BOOK_CERTIFICATES	AddressBook
$MY_CERTIFICATES	MY

You can use PKCertTool to set up certificate databases having custom names and locations if you do not want to use the PKZIP defaults (see "Locating Certificate Store Databases," above). You can then set the environment variables to point to the databases you have set up.

Use command lines like the following to set the variables for sh-based shells (sh, ksh, bash, zsh). Replace the database names and paths with your own:

```
ROOT_CERTIFICATES=/usr/local/certificates/certificates.db
export ROOT_CERTIFICATES
CA_CERTIFICATES=/usr/local/certificates/certificates.db
export CA_CERTIFICATES
ADDRESS_BOOK_CERTIFICATES=/usr/local/certificates/certificates.db
export ADDRESS_BOOK_CERTIFICATES
MY_CERTIFICATES=/home/certificates.db
export MY_CERTIFICATES
```

With csh or tcsh, use command lines like these:

```
setenv ROOT_CERTIFICATES /usr/local/certificates/certificates.db
setenv CA_CERTIFICATES /usr/local/certificates/certificates.db
setenv ADDRESS_BOOK_CERTIFICATES
/usr/local/certificates/certificates.db
setenv MY_CERTIFICATES /home/certificates.db
```

Put the commands in users' login files (`.profile` for sh users, `.cshrc` for csh users) to set the variables each time users log on.

Migrating Certificates from a PKZIP 6.x Store

PKZIP version 6 used a different arrangement for storing certificates. On this arrangement, certificates were stored in PKCS#7, PKCS#12, and PEM files in directories in the file system, and the environment variables pointed to the directories.

To migrate certificates from a PKZIP version 6 store, follow these steps:

1. If the environment variables described in the preceding section are set, make a note of the settings and unset the variables.
2. To import shared certificates from the old databases into the new one, have an administrator with access to the directories containing the certificates run the PKCertTool command lines below.

If necessary, replace the default paths shown in the command lines with the actual paths to the old stores. (The values, if any, that you noted down for the environment variables should give these paths.)

```
pkcerttool -add -store ROOT -all /usr/local/certificates/ROOT
pkcerttool -add -store CA -all /usr/local/certificates/CA
pkcerttool -add -store AddressBook -all
/usr/local/certificates/AddressBook
```

3. Have each user run the command line below to import personal certificates and private keys.

   PKCertTool will likely prompt the user to enter a passphrase for each certificate in a PKCS#12 file to decrypt the private key. PKCertTool will prompt again for a passphrase to use to encrypt each private key to be added.

   ```
   pkcerttool -add -store MY -all $HOME/.certificates
   ```

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.

Support for using smart card and token certificates is available only on Windows, not on UNIX or Linux.

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.

If any of your recipients run an older PC with Windows 2000 or older, use the ***win2000*** sub-option so those recipients can extract files encrypted with AES for a recipient list. You may also need this option if your recipient uses a smart card to decrypt.

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 an older version of Windows 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. Realize that recipients using smart cards or running older versions of Windows 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 2000 or older 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 8):

```
CryptOptions = Smartcard, Win2000
```

To configure one of the sub-options off, prefix it with a hyphen:

```
pkzipc -config -cryptoptions=-smartcard
```

or, to configure both off:

```
pkzipc -config -cryptoptions=-smartcard,-win2000
```

Either option can also be turned off just for the current command line, to override a configured default setting:

```
pkzipc -add -cryptoptions=-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 -cryptoptions=smartcard,win2000,fastaes
```

Enabling PKCS#11 Support in Windows and UNIX

You may have certifcates stored in a Hardware Security Module (HSM) such as Gemalto Luna. The PKCS#11 standard provides an interoperable standard API for using keys from HSMs and Smart Cards. PKZIP offers PKCS#11 support to permit connections to these certificate stores.

Only certificates and corresponding private keys that are stored on HSM and satisfy the following requirements can be used:

- Only **RSA** keys are supported.
- Private Key MUST have the same **CKA_ID** or **CKA_LABEL** value as the corresponding Certificate.
- Consult with your device vendor if you have OpenPGP keys stored in the device.

To enable PKCS#11 support, you must create a ***pkcs11.cfg*** file first.

This file should have these fields, in this form; the values are defined by the PKCS#11 device setup. Consult with your device vendor if necessary:

```
[{"name":"Soft HSM","module":"libsofthsm2.so",
"slots":[{"id":1679438631,"pin":"quality3"}]}]
```

Field	*Description*
"name"	Provider name
"module"	File name of the shared library
"slots"	List of slots containing the certificates on the specified module, with ID number and Login pincode. If you do not designate the slots, PKZIPC will try to connect to all available slots using the default login.
"id"	ID number for the slot identified in the slots list.
"pincode"	PIN or passphrase for the slot identfied in the slots list.

Save this file to the PKZIP Server program directory.

Once the file is created, test the connection by typing:

```
pkzipc -listcert -pkcs11
```

You should see all available certificates, including those from the listed PKCS#11 device(s),

Note: An encrypted Pincode can be stored if policy requires it. Additional info on encrypting a PIN can be obtained from PKWARE Technical Support.

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 Server can also create OpenPGP-compliant files and sign files with OpenPGP keys. In this section, you’ll learn more about the OpenPGP standard, and how to use PKZIP Server 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 keys 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 a pair of OpenPGP keys (public and private) to identify yourself. When a friend comes to visit, display the public key. The friend can now sign your public key (often called “key signing”) and certify that this key represents you. Now everyone who trusts the person who signed your key can also trust that your key 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.

Algorithm	*Type*
RSA	Public-Key Signature or Encryption
Elgamal	Public-Key Encryption
DSA	Public-Key Signature
3DES	Symmetric-Key
CAST5	Symmetric-Key
IDEA	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

Use these commands to identify existing OpenPGP private and public key pairs for use in creating and signing OpenPGP files:

```
pkzipc -listcertificates
pkzipc -listcertificates=AddressBook
```

In Windows, SecureZIP searches for these public keyring files:

- ***%PK_OPENPGP_PUBLIC_RING%***

- ***<Documents>\PGP\pubring.pkr***
- ***<AppData>\gnupg\pubring.gpg***

If your keyring is installed in a different location than listed here, add these environment variables to your Windows system:

PK_OPENPGP_PUBLIC_RING=<PATH>\pubring.pkr

PK_OPENPGP_SECRET_RING=<PATH>\secring.skr

On UNIX systems, SecureZIP searches for these public keyring files:

- ***$PK_OPENPGP_PUBLIC_RING***
- ***$HOME/.pgp/pubring.pkr***
- ***$HOME/.gnupg/pubring.gpg***

Windows:

- ***%PK_OPENPGP_SECRET_RING%***
- ***<Documents>\PGP\secring.skr***
- ***<AppData>\gnupg\secring.gpg***

UNIX:

- ***$PK_OPENPGP_SECRET_RING***
- ***$HOME/.pgp/secring.skr***
- ***$HOME/.gnupg/secring.gpg***

Working with Recipient Groups

You may have multiple recipients that you regularly send ZIP or OpenPGP files to. SecureZIP allows you to create and configure groups of recipients (whether those recipients use OpenPGP or X.509 to identify themselves) to simplify sending encrypted files to such groups.

Creating a Group

To create a new group, you need to name the group. You can also include a description of the group. Use this syntax:

```
pkzipc -groupadd=<name> [groupdesc=<description>] [-
groupfile=<group file name>]<item>
```

You must add at least one item to a group when you create it. SecureZIP recognizes any value for the ***recipient*** command as a group item. These include Common Name, Friendly Name, email address, KeyID (for OpenPGP). You can also use an LDAP search filter to identify multiple items. See “Specifying Recipients” in Chapter 3 for more information.

Use ***groupdesc*** to (optionally) add a brief description of the group.

By default, the file holding all the group information is called ***group.xml***, and is stored in the same location as the SecureZIP configuration file. You can rename this file or point it to another folder either directly from the command line or (preferably) through your configuration file. See “Changing a Default Value” in Chapter 8 for more information.

Groups can also contain other groups. To add an individual recipient and the members of the ***test*** group to the existing ***multi*** group:

```
pkzipc -groupadd=multi,user1@example.com,test
```

When you send OpenPGP files to a group, only group members with OpenPGP keys will be able to decrypt the files.

Listing Groups

Grouplist displays a list (with any description) of all your existing groups:

```
pkzipc -grouplist
```

Use ***groupdetail*** to see the content of either a specified group, or all groups:

```
pkzipc -groupdetail=<group name>
```

If you have multiple group files, you can also optionally specify a group file to list:

```
pkzipc -grouplist -groupfile=<group file name>
pkzipc -groupdetail -groupfile=<group file name>
```

Removing Items from a Group

Use ***groupremove*** to remove one or more keys from a particular group:

```
pkzipc -groupremove=<name>,<item>
```

To remove an entire group:

```
pkzipc -groupremove=<name>
```

You may also specify a groupfile name with this option:

```
pkzipc -groupremove <name>,<item> -groupfile=<group file name>
```

Creating an OpenPGP file to Send to a Group

To create a new OpenPGP file named ***file.pgp*** and specify a group of recipients:

```
pkzipc -add -archivetype=pgp -recipient=group=<name> <file.pgp>
<file1>
```

You can optionally specify a groupfile in this command.

Configuring Other OpenPGP Settings

You can configure both the default hash and encryption algorithm and default signing key 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
```

7 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.

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

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

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

Viewing the Contents of a ZIP File

view

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

```
pkzipc -view test.zip
```

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

```
Viewing .ZIP: test.zip

  Length Method     Size  Ratio    Date     Time   CRC-32  Attr   Name
  ------ ------     ----  -----    ----     ----   ------  ----   ----
   8369B DeflatN    3084B 63.2% 06/01/2001  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%                                      2
```

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

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

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

Displaying a Brief View of a ZIP File

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

```
pkzipc -view=brief test.zip
```

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

```
Viewing .ZIP: test.zip

  Length Method     Size  Ratio    Date     Time  Name
  ------ ------     ----  -----    ----     ----  ----
   8369B DeflatN    3084B 63.2% 06/01/2001  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
```

Note: The above ***view*** list was generated from a DOS command line. In a UNIX ***view*** listing the “Attributes” row would be replaced by a "Mode" row, displaying the global, group and user permission octal values.

In Windows, the Attributes include:

Value	*Description*
s	System
a	Archive
h	Hidden
R	read only
W	Read only is not set
V	volume
D	directory
t	Temporary
c	Compressed
o	Offline
e	Encrypted
1	Sparse
2	Reparse

Value	Description
i	Not indexed

Displaying Security Information of the ZIP File

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

```
pkzipc -view=security test.zip
```

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

```
Viewing .ZIP: test.zip

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

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

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

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

Verifying the Encryption Type for an Archive

VerifyEncryption

Perhaps you've received an archive with one or more encrypted files. Before you can decrypt it, you need to know how it was encrypted. Use ***`VerifyEncryption`*** to determine the type of encryption used.

Whether you're creating an encrypted archive manually or with a script, you may also need to check whether an archive, or files in that archive, were encrypted, and how to decrypt them. This option will help as well.

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

```
-verifyEncryption=recipient test.zip
```

The results appear:

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

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

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

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

```
pkzipc -verifyEncryption=passphrase,recipient test.zip
```

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

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

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


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

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

Verifying Recipients Listed for a File

VerifyRecipient

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

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

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

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

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

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

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

#<file name>	Specifies a PKCS#7 file with its certificates

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

Renaming Files

rename

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

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

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

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

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

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

Sample results:

Name of original file	*Name of copy*
blue.txt	**green**.txt

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

The ***rename*** option is particularly useful when adding or extracting streamed data. (See "Adding Data from STDIN or Special Files" in Chapter 3 and "Extracting Data to STDOUT or Special Files" in Chapter 4.)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Translating End-of-Line Sequence

translate

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

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

Sub-Option	*Description*
none	Does not change line endings
dos	DOS/Windows (carriage return/newline)
mac	MacOS (carriage return)
unix	UNIX (newline)
ebcdic,nl	With ZDW files, substitute EBCDIC newline (0x15)
ebcdic,lf	With ZDW files, substitute EBCDIC linefeed (0x25)
ebcdic,crlf	With ZDW files, substitute EBCDIC carriage return/linefeed (0x0D25)
ebcdic,lfcr	With ZDW files, substitute EBCDIC linefeed/carriage return (0x250D)
ebcdic,crnl	With ZDW files, substitute EBCDIC carriage return/newline (0x0D15)

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

```
pkzipc -extract -translate=UNIX  test.zip
```

Converting File Names to a Short Format

shortname

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

```
pkzipc -add -shortname=dos save.zip
pkzipc -add -shortname save.zip
```

Or, abbreviated:

```
pkzipc -add -short save.zip
```

The option can be configured to be on by default.

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

Inserting a Timestamp in the Archive File Name

substitution

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

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

Note: See "Time Stamping Your Signed ZIP Archive" in Chapter 3 for information on using an independent Time Stamp Authority to securely establish when a file was created or modified.

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

Token	*Replaced by*
{id}	A job ID specified separately with the ***jobid*** option. For example, if run in 2006: pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc produces a ZIP file named: myJob2006.zip
{mm}	Month, 2-digit
{m}	Month, 1-digit (if possible)
{dd}	Day, 2-digit
{d}	Day, 1-digit (if possible)
{yyyy}	Year, 4-digit
{yy}	Year, 2-digit
{HH}	Hour, 2-digit, 24-hour format
{H}	Hour, 1-digit (if possible), 24-hour format
{hh}	Hour, 2-digit, 12-hour format
{h}	Hour, 1-digit (if possible), 12-hour format
{MM}	Minute, 2-digit
{M}	Minute, 1-digit (if possible)
{SS}	Second, 2-digit
{S}	Second, 1-digit (if possible)
{ampm}	a.m. or p.m. indicator to identify current 12-hour segment of the day

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

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

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

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

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

If the current date and time are August 09, 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 (Windows)

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.

Handling Warnings

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

Pausing on Warnings

warning

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

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

```
pkzipc -extract -warning save.zip *
```

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

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

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

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

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

To override a 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.

Ignoring Warnings

IgnoreWarnings

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

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

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

Multiple warning numbers can be specified, separated by commas:

```
-ignore=40,41,79
```

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

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

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

```
-ignore=42,-41
```

The ***IgnoreWarnings*** option can be used with ***add***, ***extract***, ***test***, and ***view***. See 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`.

- UNIX command line:

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

➢ Windows command line:

```
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, you will be prompted for each file that is added to set to ASCII, BINARY or if PKZIP should determine the best type. The following examples show the different uses for this option.

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

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

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

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

To prompt the type for each file:

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

Encoding an Archive to another Type

encode

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

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

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

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

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

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

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

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

```
pkzipc -encode=gz save.tar
```

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

```
pkzipc -archivetype=pgp -encode
```

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

Removing an Intermediate Archive

movearchive

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

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

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

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

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

Generate a List File

listfile

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

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

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

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

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

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

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

When used with ***add*** (though not with ***extract***), the ***listfile*** option takes account of other options—for example, the options ***path***, ***recurse***, and ***directories*** that specify path information to save with the added files. For example, the ***path*** option in the following command line causes full path names to be saved with added files, so this information is saved in the list file as well:

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

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

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

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

Logging Events

LogError, JobID, Log, LogOptions

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

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

On UNIX, log entries sent to the syslog are prefixed with a datestamp followed by `PKZIP[XX]`, where `XX` is the number of the process ID of the PKZIP invocation that generated the message. With this information, you can identify messages that relate

to PKZIP and determine which PKZIP messages belong to which process when more than one PKZIP process is running.

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

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

On Windows, the syslog is the Application event log. For each entry, the event log records the name of the source of the message (for example, *PKZIP*), the date and time, an application-specific event ID, and the text associated with that event ID.

If you uninstall PKZIPC, the Windows event log will include this message, followed by the event text:

"The description for Event ID (1) in Source (PKZIP) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. You may be able to use the /AUXSOURCE= flag to retrieve this description; see Help and Support for details. The following information is part of the event"

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

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

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

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

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

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

Sending Information to an SNMP Host

SnmpTrapHost

SNMP (Simple Network Management Protocol) is a standard protocol for monitoring and managing activity on a network. It provides for applications to send messages, called *SNMP traps*, to an SNMP receiver application on a network server to inform

the receiver application of selected events such as error and warning conditions. PKZIP can also send traps on application startup and shutdown.

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

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

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

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

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

where:

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

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

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

Kinds and Contents of SNMP Traps Sent

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

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

Table: SNMP Traps

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

The PKWARE MIB

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

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

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

Setting Execution Priority

priority

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

On Windows, priority levels are:

`Low`

`BelowNormal`

`Normal` (the default)

`AboveNormal`

`High`

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

The ***priority*** option can be used only to lower the priority on UNIX. On Windows, you must be logged on as an administrator to raise the priority level.

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

The following command line (Windows) sets priority to `low`:

```
pkzipc -add -priority=low mydocs.zip *.doc
```

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

8 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 : Static
Encode (OpenPGP) = Disabled
FIPSMode = Disabled
Hash (OpenPGP) = SHA-1
Level = Normal
Locale = Enabled
More = Disabled
NoArchiveExtension = Disabled
NoFix = Disabled
OptionChar = -
Priority = Normal
Shortname = None
Sort = None
Substitution = Disabled
Times = All
UsePGPName = Disabled
ZoneIdentifier = Disabled

ASCII = Disabled
Authenticate = Disabled
AVArgs = Disabled
AVScan = Disabled
Binary = Disabled
Certificate =
Certificate (OpenPGP) = Disabled
CryptAlgorithm = Traditional
CryptAlgorithm (OpenPGP) = AES (256-bit)
CryptOptions = Smartcard, Win2000, FastAES
Add = Add All Files
CD = Normal
Comp Method = Deflate
Encode = Disabled, UUE
Extract = Extract All Files
Hash = SHA-1
KeyPassphrase = Disabled
ListChar = @
Lowercase = Disabled
MoveArchive = Disabled
NoExtended = Disabled
OpenFile = Never (skip)
Passphrase = Disabled
Recurse = Disabled
Shred = None
Span = None, Auto-Detect
Test = All Files
UTF8 = Enabled
View = Normal
```

```
Embedded = Disabled
Error = None
FTP = Disabled
Header = Disabled
IgnoreWarnings = None
LDAP = Disabled
Log = stdout
LogError = stderr
LogOptions = None
MailBCC = Disabled
MailBody = Disabled
MailCC = Disabled
MailFrom = Disabled
MailOptions = None
MailReplyTo = Disabled
MailServer = Disabled
MailSubject = Disabled
MailTo = Disabled
GroupFile = Default
Recipient = Disabled
Recipient (OpenPGP) = Disabled
Sign = Disabled, Central directory and individual files
Silent = Copying
SnmpTrapHost = Disabled
Strict = Disabled : KeyUsage, TimeValid, TimeNesting
TS = Disabled
Temp = Disabled
VerifySigner = Disabled
VerifySigner (OpenPGP) = 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
        AltStream = None
        Attributes = Read-Only, Archive
        Before = Disabled
        Exclude = Disabled
        Include = Disabled
        Larger = Disabled, 0
        Mask = None
        Newer = Disabled
        Older = Disabled
        Overwrite = Always Overwrite
        Overwrite (ArchiveEach) = Increment
        Path = No Path Information
        Smaller = Disabled, 18,446,744,073,709,551,615
        Translate = None - No Conversion

Extraction Options
        After = Disabled
        AltStream = Native
        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. If an option has a default value of ***None***: PKZIP applies the option 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

Filter options listed in the display of default settings take separate defaults for compression and extraction. That is, you can make each filter behave differently when zipping and unzipping an archive. 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:

Compression Method Options	*Description*
deflate64	Sets the compression method to Deflate64
bzip2	Sets the compression method to BZIP2
dclimplode	Sets the compression method to DCL Implode
lzma	Sets the compression method to LZMA
ppmd	Sets the compression method to PPMd
store	Sets the compression method to Store (that is, no compression)

The options in the next table set both compression method and level:

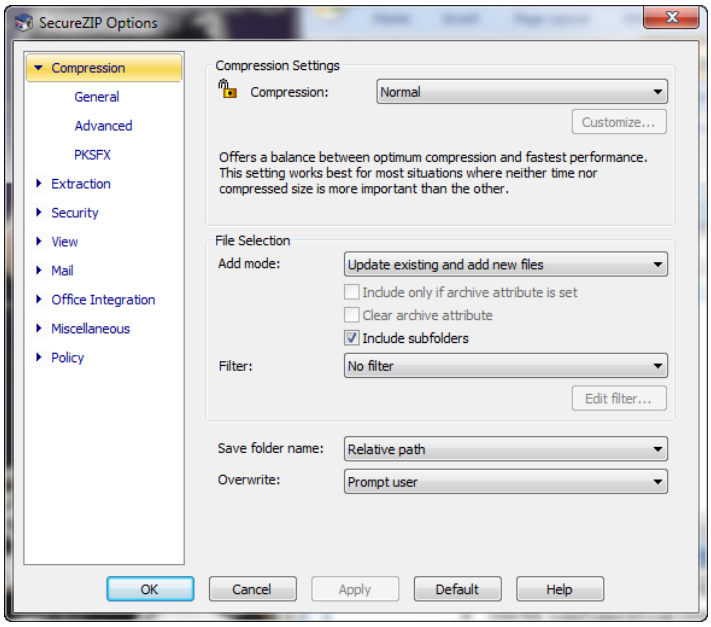
Option	*Description*
speed	Sets the compression method to Deflate—the initial PKZIP default method—and the level of compression to 1 (the lowest)
fast	Sets the compression method to Deflate and the level of compression to 2
normal	Sets the compression method to Deflate and the level of compression to 5. **Normal** is the initial default setting for compression method and level for PKZIP.
maximum	Sets the compression method to Deflate and the level of compression to 9
level=0	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

If you have PKZIP for Windows Desktop installed, you can use the graphical Options dialogs instead of the command line to change defaults:



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.

9 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, you 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 an "+" character (e.g., +file1.txt). For more information on using list files with PKZIP see “Compressing Files with a List File” in Chapter 3 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:

Column	*Purpose*
Name/Description	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.
Value(s)	Lists any sub-options or values associated with the command/option and specifies any initial default values
Example usage	Shows examples of the command/option used in a PKZIP command line
Used with	Identifies the item as a command or an option: a listing of *standalone* in this column means that the item is a command. For options, the column lists commands that the option can be used with. If *standalone* 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:

Name/Description	*Value(s)*	*Example usage*	Used with
204 Turns on PKZIP for DOS 204g compatibility Configurable	No sub-options. --- No default value.	***pkzipc -add -204 save.zip ******	add

Name/Description	*Value(s)*	*Example usage*	Used with
add Add files to an archive Configurable	**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 **archive** - Turn off archive attribute of all added files (prepares backup file set for incremental archiving) (Windows). **freshen** - Add only files that the archive already contains an older copy of **update** - Freshen files that are in the archive already and add any new ones **incremental** - Add only files that have the archive attribute on, and then turn off the archive attribute (Windows) **-incremental** - Add only files that have the archive attribute on, and do not turn off the archive attribute afterward (Windows) --- Default = **all**	***pkzipc -add save.zip *.doc*** ***pkzipc -add=freshen save.zip *.doc*** ***pkzipc -add=incremental save.zip *.doc*** ***pkzipc -add=-incremental save.zip *.doc*** Outputs the archive to STDOUT instead of to a file: ***pkzipc -add -noarchiveextension -silent=normal - *.txt***	standalone
after Process files that have the specified date or a later one Configurable separately for add and extract operations.	Any date in format specified in Country-Settings or the ***locale*** option. For example, the US date format is: mmddyy or mmddyyyy --- No default value.	For compression: ***pkzipc -add -after=09152003 save.zip *.doc*** For extraction: ***pkzipc -ext -after=09152003 save.zip *.doc***	add, extract, delete, test, view, delete, console

Name/Description	*Value(s)*	*Example usage*	Used with
altconfig Creates or updates an alternate configuration file containing alternate, specified defaults when used with the *configuration* command; loads the specified alternate configuration file when used in a command line with any command other than *configuration*.	Path and name of alternate configuration file to create, update, or load	Create or update an alternate configuration file secure.xml with specified defaults. File is created if it does not exist already, or updated if it does: ***pkzipc -config -altconfig=c:\secure.xml -cryptalgorithm=aes,256 -sign=all -certificate="John Public"*** Use the default settings specified in alternate configuration file secure.xml when adding files to archive foo.zip: ***pkzipc -add -altconfig=c:\secure.xml -passphrase foo.zip *.doc***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
AltStream Controls adding and extracting of Mac resource forks and NTFS alternate streams Configurable	**none** - Alternate streams will be skipped. **native** - Native alternate streams will be processed. **all** - All alternate streams are processed. Non-native alternate streams are extracted using @@MAC@RESOURCE-FORK@ or @@NTFS@STREAM@ in the file name. Defaults: Add = none Extract = native	Create an archive that includes alternate streams associated with *.txt files: ***pkzipc -add -altstream=all test.zip *.txt*** Extracts an archive and skips any alternate streams: ***pkzipc -extract -altstream=none test.zip***	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
archivedate Sets the modification date of the archive file. Configurable **Note:** The archivedate option is the same as the older **zipdate** option, which is now deprecated.	**newest** - Sets date of the archive to the date of the newest file in the archive **oldest** - Sets date of the archive to the date of the oldest file in the archive **retain** - When updating, keeps the date the archive had before it was updated. When creating a new archive, behaves the same as **none** **none** - Sets the date of the archive to the date of its latest modification --- Default = **none**	***pkzipc -add=update -archivedate=retain save.zip *.txt***	add, delete, fix, header, comment, sfx
archiveeach Creates a separate archive for each of multiple files specified in a single command line. Can be used with *archivetype* and *encode* to create .tar.gz archives.	**<destination>** - Directory in which to create the archives By default, archives are created in the current directory.	Creates a separate ZIP archive for each file in the current directory: ***pkzipc -add -archiveeach *.**** Creates the archives in a specified destination: ***pkzipc -add -archiveeach=C:\newzips *.**** Creates .tar.gz archives: ***pkzipc -add -archiveeach -archivetype=tar -encode=gz C:\data*.****	add

Name/Description	*Value(s)*	*Example usage*	Used with
archivetype 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. If you wish to create an OpenPGP file, always use **archivetype=pgp**.	**bzip2** - Specifies the Bzip2 archive type.* **zip** - Specifies the .ZIP archive type. (default) **gzip** - Specifies the GZIP archive type.* **pgp** – Specifies the OpenPGP archive type. **tar** - Specifies the TAR archive type. **uue** - Specifies the UUENCODED archive type.* **xxe** - Specifies an XXENCODED archive type.* * 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 encode option to encode this archive as the single-file archive type that you want.	Creates a TAR archive named myfile.foo.tar ***pkzipc -add -archivetype=tar myfile.foo*** Extracts from a BZIP2 archive on STDIN ***pkzipc -extract -archivetype=bzip2 -noarchiveextension -silent=input -*** Creates an OpenPGP archive encrypted for Bob Smith called myfile.pgp ***pkzipc -add -archivetype=pgp -recipient="Bob Smith" myfile.pgp *.txt***	add, extract, test, view
ascii Set the internal attribute bit (ASCII/Binary) to ASCII. Configurable	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. ---------- No default value.	***pkzipc -add -ascii="*.txt" save.zip **** ***pkzipc -add -ascii save.zip ******	add

Name/Description	*Value(s)*	*Example usage*	Used with
attributes Stores files with the specified file attribute information in the archive file. Configurable separately for add and extract operations. (Windows)	**hidden** - select hidden files. **system** - select system files. **readonly** - select read-only files. **archive** - select files with the archive bit set. **all** - select all types of files. **none** - do not select files that have hidden, system, or read-only attributes; overrides the default attributes setting in configuration file. **<hex value>** -The hex value of an attribute to be selected, or the logical OR of multiple hex values ---------- Default = readonly, archive	***pkzipc -add -attributes=system,hidden save.zip ******	add, extract
authenticate Verifies that an archive is signed.	No sub-options. ---------- No default value.	Extracts the archive if it was signed ***pkzipc -extract -authenticate signed.zip***	extract
avargs Specifies any command line arguments to use when running the anti-virus program given in **avscan** Configurable	<**command line**> - A command line that runs an anti-virus program	***pkzipc -extract -avscan= f-prot.exe -avargs="%e /silent /nomem /noboot" myfiles.zip***	extract
avscan Turns on virus scanning: runs the specified anti-virus program using the anti-virus command line arguments in **avargs** Configurable	<executable> - The name of the anti-virus program executable—with path, if necessary	***pkzipc -extract -avscan= f-prot.exe -avargs="%e /silent /nomem /noboot" myfiles.zip***	extract

Name/Description	*Value(s)*	*Example usage*	Used with
before Process files that are older than a specified date. Configurable separately for add and extract operations.	Any date in format specified in Country-Settings or the locale option. For example, the US date format is one of the following: mmddyy mmddyyyy --- No default value	For compression: ***pkzipc -add -before=09152003 save.zip *.doc*** For extraction: ***pkzipc -extract -bef=09152003 save.zip *.doc***	add, extract, delete, test, view, print, console
binary Treats the files to be added as binary files: sets the internal ASCII/Binary attribute bit of the files to binary. Configurable	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.	***pkzipc -add -binary="*.exe" save.zip **** ***pkzipc -add -binary save.zip***	add
bzip2 Compress files using the BZIP2 method. **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.	No sub-options Default compression level: 5	To compress files using the bzip2 algorithm and level 9 compression: ***pkzipc -add -bzip2 -level=9 save.zip doc1.txt*** To compress files using the default compression level (level 5): ***pkzipc -add -bzip2 save.zip *.doc***	add

Name/Description	*Value(s)*	*Example usage*	Used with
cd Encrypt file names and other metadata in a ZIP archive's central directory. Requires that *passphrase* and or *recipient* options also be used. Uses strong encryption; does not work with traditional ZIP encryption. Encrypting file names produces an archive that requires PKZIP or SecureZIP version 8.0 or later to open it. Configurable	**encrypt** - Encrypt file names and the archive's central directory **normal** - Do not encrypt file names; produces a normal ZIP file. Use to override a configured default setting that would otherwise encrypt file names. ---------- Default = encrypt	***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***	add

certificate Specifies the certificate to use to digitally sign a .ZIP file. Configurable	**<Name>** - The common name of the subject of the certificate (that is, the *cn* 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: cn= If the certificate name contains a space, enclose the certificate name in quotation marks ("My Name"). <**Email address**> - The email address of the certificate (that is, the *e* field in a string representation of a certificate); optionally, precede with: e= The specified certificate must exist in the MY certificate store. If more than one certificate in the MY store has the specified name, the first certificate is used. **#<file name>** - Specifies a PKCS#12 file that contains the certificate you want to use. 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. The **certificate** option can be used	***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***	add, delete, comment, sign, header

Name/Description	*Value(s)*	*Example usage*	Used with
	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.		
comment Include a text comment for files within an archive file. When you run the command, PKZIP prompts you to enter the comment. Configurable	**all** - All files already in the archive and all files added to or updated in the archive are commented **unchanged** - Only files that are not changed in the archive are commented **add** - Only newly added files and versions of files are commented **freshen** - Only newly added versions of files already in the archive are commented **update** - Only newly added files and versions of files already in the archive are commented (the opposite of **unchanged**) **none** - No comments added ---------- Default = **none** Value if used on command line without a sub-option = **add**	***pkzipc -add -comment=all save.zip *.doc***	add, standalone

Name/Description	*Value(s)*	*Example usage*	Used with
configuration Defines default values for PKZIP commands and options	**<command or option>** - Any configurable command or option **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. **silent** - Suppresses list of configured settings that is ordinarily displayed after a command or option is configured. ---------- No default value.	***pkzipc -config -extract=freshen*** To see the current configuration values, type: ***pkzipc -config*** To open the Configuration dialogs of the GUI product for use in setting configuration defaults: ***pkzipc -config=gui*** Configures ***overwrite*** option and suppresses display of settings afterward: ***pkzipc -config=silent -overwrite=never*** Configures ***silent*** option and suppresses display of settings afterward: ***pkzipc -config=silent -silent***	standalone
console Extracts files to the screen (standard output) instead of to disk	No sub-options. ---------- No default value.	***pkzipc -console save.zip *.txt***	standalone
crl Tells PKZIP to use static CRL checking when verifying certificates. Warns if a certificate to be used for digital signing, encryption, or authentication is listed as revoked in an accessible CRL (certificate revocation list). Use with ***strict*** option to prevent the certificate from being used. Configurable	**ocsp** - Check the Online Certificate Status Protocol for revocation status **static** - Check static CRL **none** - Turn off CRL check ---------- No default value.	***pkzipc -add -certificate="John Adams" -crl test.zip*** ***pkzipc -add -recipient="John Q. Public" -crl -strict test.zip *.doc*** ***pkzipc -extract -crl test.zip*** ***pkzipc -test -crl=static,ocsp signed.zip***	add, comment, delete, extract, header, listfile, print, test, view

Name/Description	*Value(s)*	*Example usage*	Used with
cryptalgorithm Encrypts files using the specified encryption algorithm. Configurable	The encryption algorithm to use. The ***listcryptalgorithms*** command lists the strong encryption algorithms available to you. Specify a strong encryption algorithm as it is listed in the output from the ***listcryptalgorithms*** command. Default = Traditional PKWARE encryption Value if used on command line without a sub-option = The strongest algorithm available on the system	Encrypt all files added with 128-bit AES using the specified passphrase: ***pkzipc -add -cryptalgorithm=aes,128 -passphrase save.zip *.doc*** Encrypt all files added with 3DES using the certificate named "My friend": ***pkzipc -add -cryptalgorithm=3DES,168 -recipient="My friend" save.zip *.doc*** Override a configured strong encryption setting to use traditional encryption just for the current command line: ***pkzipc -add --cryptalgorithm -passphrase save.zip *.doc*** Create an OpenPGP archive called myfile.pgp using CAST5: ***pkzipc -add -archivetype=pgp -cryptalg=cast5 -myfile.pgp *.txt***	add
cryptoptions Controls advanced encryption compatibility options. 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. Configurable	**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. **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.	To configure FastAES on : ***pkzipc -config -cryptoptions=FastAES*** To turn off **smartcard** in a command line: ***pkzipc -add -cryptoptions=-smartcard -recipient="John Q. Public" test.zip*** To configure both sub-options off: ***pkzipc -config -cryptoptions=-smartcard,-win2000*** To configure both sub-options on: ***pkzipc -config -cryptoptions=smartcard,win2000***	Add Windows: Extract, Test

Name/Description	*Value(s)*	*Example usage*	Used with
	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. --- –smartcard and win2000 sub-options are on by default. FastAES is not.		
dclimplode Instructs PKZIP to use the data compression library compression scheme. Configurable	**ascii** - use with ASCII files. **binary** - use with BINARY or unknown data files. 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. --- No default value	***pkzipc -add -dclimplode=ascii,4096 text.zip *.txt***	add
default Reset the original defaults in the configuration file for all commands and options	No sub-options No default value.	To reset all defaults: ***pkzipc -default***	standalone

Name/Description	*Value(s)*	*Example usage*	Used with
deflate64 Compress files using the Deflate64 method. Configurable **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.	No sub-options. No default value.	To compress files using Deflate64 algorithm and level 9 compression: ***pkzipc -add -deflate64 -level=9 save.zip doc1.txt*** To compress files using the normal, default compression level (level 5): ***pkzipc -add -deflate64 save.zip *.doc***	add
delete Remove (delete) files from an archive	**<files>** -Names or file name pattern of files to delete No default value.	For individual files: ***pkzipc -delete save.zip doc1.txt*** For a specific file pattern: ***pkzipc -delete save.zip *.doc***	standalone
directories When adding, includes matching files in subdirectories and stores directory path names; when extracting, recreates saved directory paths. Configurable **Note:** Using this command is the same as combining the *path* and *recurse* commands.	**current** - Store the path from the current directory. **root** or **full** - Store the entire path beginning at the root of the drive; also referred to as "full" path. **specify** or **relative** - Store path information relative to the specified directories, for their subdirectories **none** - No path information stored --- Default = **none** when used with **add**; **full** when used with **extract** Value if used on command line without a sub-option = **current**	Compression example (assumes you are in \wp): ***pkzipc -add -directories=root save.zip docs**** The path stored is wp/docs/. ***pkzipc -add -directories=current save.zip docs**** The path stored is: “docs\”. Extraction: ***pkzipc -extract -directories save.zip **** Note: UNIX users should use the *include* option or place quotation marks around wildcard designations to bypass automatic wildcard expansion by the shell, which may restrict your pattern search. See “Using Wildcards with PKZIP on UNIX” in Chapter 2.	add, extract

embedded 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. Configurable	**arj** - Extract the contents of lone archives embedded in ARJ archives, without prompting **-arj** - Do not extract the contents of lone archives embedded in ARJ archives, and do not prompt **BinHex** - Extract the contents of lone archives embedded in BinHex archives, without prompting **-BinHex** - Do not extract the contents of lone archives embedded in BinHex archives, and do not prompt **bzip2** - Extract the contents of lone archives embedded in BZIP2 archives, without prompting **-bzip2** - Do not extract the contents of lone archives embedded in BZIP2 archives, and do not prompt **cab** - Extract the contents of lone archives embedded in CAB archives, without prompting (Windows only) **-cab** - Do not extract the contents of lone archives embedded in CAB archives, and do not prompt (Windows only) **gzip** - Extract the contents of lone archives embedded in GZIP archives, without prompting **-gzip** - Do not extract the contents of lone archives embedded in GZIP archives, and do not prompt **lzh** - Extract the contents of lone archives embedded in LZH archives, without prompting	To extract an embedded archive from a ZIP file without prompting: ***pkzipc -extract -embedded=zip outerarchive.zip*** To suppress the prompt and not extract archives embedded in ZIP files: ***pkzipc -extract -embedded=-zip outerarchive.zip***	extract, console, print

Name/Description	*Value(s)*	*Example usage*	Used with
	-lzh - Do not extract the contents of lone archives embedded in LZH archives, and do not prompt **rar** - Extract the contents of lone archives embedded in RAR archives, without prompting (Windows only) **-rar** - Do not extract the contents of lone archives embedded in RAR archives, and do not prompt (Windows only) **uue** - Extract the contents of lone archives embedded in UUENCODED archives, without prompting **-uue** - Do not extract the contents of lone archives embedded in UUENCODED archives, and do not prompt **xxe** - Extract the contents of lone archives embedded in XXENCODED archives, without prompting **-xxe** - Do not extract the contents of lone archives embedded in XXENCODED archives, and do not prompt **zip** - Extract the contents of lone archives embedded in ZIP archives, without prompting **-zip** - Do not extract the contents of lone archives embedded in ZIP archives, and do not prompt ---------- Disabled by default. When used, a sub-option must be set.		

Name/Description	*Value(s)*	*Example usage*	Used with
encode As an option, used with *add*, creates an archive and converts it to the archive type specified by the sub-option. As a standalone command, converts a specified existing archive. Configurable **Note:** PKZIP creates two files when the ***encode*** option is invoked: an intermediate archive of the type specified for the ***add*** command (ZIP, by default), and an archive of the type specified for the ***encode*** option. Use the ***movearchive*** option with ***encode*** to remove (delete) the intermediate archive.	**bzip2** - Creates a BZIP2 file **gzip** - Creates a GZIP file **uue** - Creates a UUENCODED file **xxe** - Creates an XXENCODED file ---------- Default value = **uue** Value if used on command line without a sub-option = **uue**	Add files to save.zip and encode to UUE: ***pkzipc -add -encode save.zip **** Add files to a TAR archive and encode to a GZIP archive: ***pkzipc -add -encode=gz save.tar*** Encode the archive as a GZIP archive and delete the intermediate archive created by the *add* command: ***pkzipc -add -encode=gz -movearchive save.tar **** As a command, creates save.tar.gz from existing archive save.tar: ***pkzipc -encode=gz save.tar***	add
enterlicensekey Prompts for a product license key	None	***pkzipc -enterlicensekey***	standalone
error Designates warning conditions, by warning number, to treat as error condition 73 (*Warning configured as an error*) Configurable	**<warning number>** - One or more warning numbers, separated by commas. To override a warning number configured for the option (and thus *not* treat that warning as an error), precede the number with a hyphen.	***pkzipc -extract -error=42,43 files.zip*** ***pkzipc -extract -error=42,-43 files.zip***	add, extract, test, view

Name/Description	*Value(s)*	*Example usage*	Used with
exclude Exclude files from being compressed or extracted. Configurable separately for add and extract operations. **Note:** You must specify a sub-option (for example, ***file pattern*** or ***list file name*** preceded by an appropriate list character "@") with the ***exclude*** option.	The file(s) or file pattern (for example, **.doc*) being excluded. No default value **Note**: If you are adding hidden or system files in Windows to an archive with ***attr=all*** , you can still use ***exclude*** hidden folders like the contents of the Windows Recycle Bin. In this case, specify ***-exclude $RECYCLE.BIN/* .****.	Compression example: ***pkzipc -add -exclude="*.doc" save.zip*** Extraction example: ***pkzipc -extract -exclude="*.txt" save.zip*** Setting exclude default: ***pkzipc -config -exclude="*.txt"*** **Note**: When you use the ***exclude*** option with the ***configuration*** command, PKZIP prompts you to configure the ***exclude*** default for ***add*** and/or ***extract*** operations.	add, extract, delete, test, view, print, console
extract Extracts files from an archive file Configurable	**all** - Extracts all files in an archive file **freshen** - Extracts only files in the archive that are newer versions of files that already exist in the target directory **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 Default = **all**	***pkzipc -extract save.zip*** ***pkzipc -extract=update save.zip***	standalone
fast 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 Configurable	No sub-options. No default value.	***pkzipc -add -fast save.zip *.doc*** ***pkzipc -config -fast***	add

filetype Processes files with the specified file type information in the .ZIP file. Configurable separately for add and extract operations. (UNIX) Note: A "-" before a filetype sub-option tells PKZIP to exclude the specified filetype(s) regardless of the default configuration setting (for example, *-filetype=-hidden* will exclude hidden files regardless of the default configuration setting).	**block** - Include/exclude block device files. **char** - Include/exclude character device files. **directory** - Retain directory information. **hidden** - Include/exclude file names that have a dot "." in the first position of the file name (for example, .profile) **hlink** - Include/exclude hard linked files. These are linked files that are not symbolic links. They are files with a link count > 1. **pipe** - Include/exclude pipe files. These are files having a file mode starting with "p" (for example, prwxrw-rw-). Adds the pipe specification or definition (name, permissions, times, and so on), not pipe data. **regular** - Include/exclude regular files. **slink** - Include/exclude symbolic links. These are files having a file mode starting with "l" (e.g. lrw-rw-rw-). **socket** - Include/exclude sockets. These are the items that the command `ls -l` lists with an "s" at the beginning of the permissions **all** - Include all of the above file types. **none** - Exclude all of the file types above. Follow this sub-option with one	***pkzipc -add -filetype=all save.zip*** Include only pipe files: ***pkzipc -add -filetype=none,pipe save.zip***	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
	or more file types to include just those types. Default = **regular**		
fipsmode Causes SecureZIP to use only algorithms that comply with the FIPS 140 standard to perform cryptographic operations. Use the commands **listcryptalgorithms** and **listhashalgorithms** with the **fipsmode** option to see lists of algorithms available with **fipsmode**. Configurable	**enabled** - Turns the option on **disabled** - Turns the option off (the default on UNIX) 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.	Turns on **fipsmode** for the current command line: ***pkzipc -add -recipient="John Public" -fipsmode save.zip *.doc*** Overrides a configured default setting of fipsmode=enabled and turns off **fipsmode** for the current command line: ***pkzipc -extract -fipsmode=disabled wedding_plans.zip *.**** Lists encryption algorithms available with **fipsmode**: ***pkzipc -listcryptalgorithms -fipsmode***	add, extract, test, listCryptAlg-orithms, listHashAlg-orithms With file name-encrypted (FNE) archives, also applies to: comment, delete, header, view
fix Attempts to repair a corrupt ZIP archive file	**<file name>** - The name of the ZIP archive to fix No default value.	***pkzipc -fix save.zip***	standalone
ftp Transfers an archive using FTP Configurable	Syntax (optional fields in brackets): **-ftp=** [username[:pwd [:account]]@]server /path where: **username** (optional) is the user account with which to log in if the FTP server requires a login **passphrase** (optional) is the passphrase associated with the user account. Colons are not allowed in the passphrase. **account** (optional) is for use only with FTP servers that require additional authentication. Do	***pkzipc -add -ftp=serv/home/thomas mydocs.zip *.doc*** ***pkzipc -add -ftp=me@serv/home/thomas mydocs.zip *.doc*** Standalone: ***pkzipc -ftp=serve/home/thomas mydocs.zip***	add, delete, header, comment, standalone

Name/Description	*Value(s)*	*Example usage*	Used with
	not specify the account for servers that do not require it. **server** is the FTP server name **path** is the path to the destination of the transferred file on the server. Use two slashes (server//path) to specify a full path; use one slash (server/path) to specify a path relative to the login destination directory.		
Groupadd Creates a new group or adds items to the existing group	<group name>[,<recipient 1>][,<recipient 2>]...	***pkzipc -groupadd=Test,0x1234567812345678,user1@mycompany.com***	Standalone
Groupdesc Provides description for the group, can only be used with –groupadd command	<group description>	***pkzipc -groupadd=Test,0x1234567812345678,user1@mycompany.com*** ***-groupdesc="My test group"***	groupadd
Groupdetail Lists detail information about group(s)	[<group name>]	List detailed information about group 'Test': ***pkzipc -groupdetail=Test*** List detailed information about all groups: ***pkzipc -groupdetail***	Standalone
Groupfile Specifies the alternative name of the group file to use Configurable		***pkzipc -groupadd=Test,0x1234567812345678,user1@mycompany.com*** ***-groupfile=<path>"my group file.xml"*** ***pkzipc -add -groupfile=<path>"my group file.xml" -recipient=group=Test test.zip *.txt***	groupadd
Grouplist Lists names and description of all groups		***pkzipc -grouplist***	Standalone

Name/Description	*Value(s)*	*Example usage*	Used with
Groupremove Removes items from a group or an entire group	<group name>[,<recipient 1>][,<recipient 2>]...	Remove selected recipient(s) from the group: ***pkzipc -groupremove=Test,0x1234567812345678*** Remove entire group: ***pkzipc -groupremove=Test***	Standalone
hash Sets the hashing algorithm to use when signing an archive. Use **listhashalgorithms** to list hashing algorithms available. Configurable **Note:** This option requires SecureZIP.	The hashing algorithm to use, as listed in the output from the ***listhashalgorithms*** command. Default = **sha1**	***pkzipc -add -certificate="John Smith" -hash=sha1 save.zip *.doc***	add, delete, comment, header
header Creates a comment for a ZIP archive file in the header area of the file Configurable	**<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 **<comment>** - The literal comment to be used ---------- No default value.	To include literal text: ***pkzipc -add -header save.zip *.doc*** Note: PKZIP prompts for the header text To include an existing file: ***pkzipc -add -header=@text.doc save.zip *.doc***	add standalone
help Displays help screen for PKZIP	**<command or option>** - Any command or option for which help is desired. No default value.	***pkzipc -help*** Display help for the add command: ***pkzipc -help=add***	standalone

Name/Description	*Value(s)*	*Example usage*	Used with
id Preserve the user ID (UID) and/or group ID (GID) on extraction. Configurable (UNIX) Note: The user who extracts files with preserved UID and GID information must have the same UID as is archived in the .ZIP file or root (superuser) file privileges.	**userid** - retain the user ID on extraction. **groupid** - retain the group ID on extraction. **all** - retain both the user ID and group ID on extraction. **none** - retain neither the user ID or group ID on extraction. No default value.	***pkzipc -extract -id=userid save.zip***	extract
ignorewarnings Allows warnings to be ignored. Ignored warnings will still be displayed, but they will not affect the program's return code. Configurable	A comma separated list of warnings to be turned on or off	***pkzipc -config -ignore=59*** ***pkzipc -config -ignore=40,41,79*** ***pkzipc -extract -ignore=7 test.zip***	
include Include files to compress or extract. Configurable separately for add and extract operations. **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.	The file(s) or file pattern (for example, *.doc) being included. No default value.	Compress only .doc files: ***pkzipc -add -include="*.doc" save.zip*** 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: ***pkzipc -config -include="*.txt"*** Note: When you use the *include* option with the **configuration** command, PKZIP prompts you to specify whether to configure the option for *add* and/or *extract* operations.	add, extract, delete, test, view, print, console
jobid Specifies a job ID with which to prefix PKZIP log entries in the system logging facility	<**ID**> - The job ID to use	***pkzipc -add -logerror=syslog -logoptions=start -jobid=my_id2 mytestlog.zip bookmark.htm***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
keyfile 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. Configurable	<**file name**> - The name and location of the file	***pkzipc -add -certificate=#mycert.pem -keyfile=mykey.key save.zip *.doc***	add, extract, test, sign, view
keypassphrase 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. Configurable	<**passphrase**> - The passphrase, in quotes	***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***	add, extract, test, sign, view
Keyserver Defines a connection to an OpenPGP Key Server that will be used to find keys Configurable **Note**: This option is only available in versions that have certificate-based strong encryption.	Syntax is [http://][https://][[userid:password@]server[:port]	***pkzipc -config -keyserver=http://user:pass@192.168.0.1*** ***pkzipc -config -keyserver=http://user:pass@192.168.0.1:11371***	config

Name/Description	*Value(s)*	*Example usage*	Used with
larger Process only those files whose size is greater than (in bytes) or equal to a specified file size. Configurable separately for add and extract operations.	Numerical value that indicates a minimum desired file size. Acceptable notation: K - 1024B M - 1024K G - 1024M T - 1024G No default value.	Add only files that are at least 5000 bytes in size: ***pkzipc -add –larger=5000 save.zip **	add, extract, test, view, delete, print console
ldap Specifies an LDAP directory for PKZIP to search before looking in local stores for certificates containing public keys for certificate-based encryption (see the recipient option). **Note:** The **ldap** option must appear *before* the **recipient** option when the two options are used together in a command line. Configurable **Note:** The **ldap** option is available only with SecureZIP.	Syntax (optional fields in brackets): **-ldap=** [[userid :passphrase@] server[:port]] [,ssl] /ldap_base where: **userid** (optional) is the user account with which to log in if the LDAP server requires a login **passphrase** (optional) is the passphrase associated with the user account **server** (optional) is the LDAP server name or TCP/IP address **port** (optional) is the TCP/IP port to use. The default is 389 if no port is specified. **ssl** (optional) will use SSL encryption and change the default port to 636. Windows only. **ldap_base** (required) is the name of the entry that PKZIP should use as the base or root of the LDAP search for certificates, analogous to a root folder or directory in a file system	***pkzipc -add -ldap=john_p:mysecret @192.172.0.1 :389/cn=users,dc=xyz, dc=com -recipient="Mary Samplename" save.zip *.doc*** ***pkzipc -add -ldap=/cn=users,dc=xyz, dc=com -recipient=e= mary.samplename@xyz.com save.zip *.doc***	add

Name/Description	*Value(s)*	*Example usage*	Used with
	The query string format for ldap_base can vary between LDAP implementations. Check with your LDAP or network administrator for the format to use. See "Accessing Recipients in an LDAP Directory" on page 39.		
level Sets the level of compression. 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. Configurable	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. Default = level 5 (normal)	***pkzipc -add -level=9 save.zip *.doc***	add
license Displays the product license information for PKZIP	No sub-options. No default value.	***pkzipc -license***	standalone
links Specify that linked files be followed or preserved in a .ZIP archive. Configurable (UNIX) Note: Following a link results in a larger .ZIP file size since two copies of file data are compressed as though each link is a separate file.	**hlink** - Hard links are followed (stored) rather than preserved. **-hlink** - Hard links are preserved **slink** - Symbolic links are followed (stored) rather than preserved. **-slink** - Symbolic links are preserved **all** - Symbolic and hard links are followed rather than preserved. **none** - Symbolic and hard links are preserved. Default = **none**	Compress regular and hard linked files and duplicate link and file data for each hard linked file added: ***pkzipc -add -links=hlink save.zip***	add

Name/Description	*Value(s)*	*Example usage*	Used with
listcertificates Lists digital certificates in a certificate store. This will display both X.509 certificates and OpenPGP keys. The OpenPGP keys will be listed with (OpenPGP) following the name. If an OpenPGP key contains more than one userid, multiple certificates will display.	**my** - Lists personal certificates in the MY store **addressbook** - Lists public certificates in the AddressBook store **ca** - Lists intermediate, certificate authority certificates in the CA store **root** - Lists trusted certificates in the Root store Default = **my**	***pkzipc -listcertificates*** ***pkzipc -listcertificates =addressbook***	standalone
listchar Set the list character to the specified ASCII character. Prefixing a file name with the list character identifies it as a list file. Configurable	Any character in the printable ASCII range. Must not be the same as OptionChar and must not be "-". default = @	***pkzipc -config -listchar=+***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
listcryptalgorithms Displays a list of the strong encryption algorithms available for use with the **cryptalgorithm** option. With **fipsmode** on, it lists only FIPS-validated algorithms. When OpenPGP is enabled through configuration or –archivetype=pgp, CAST,128 will also be listed. **Note:** This option is only available in versions that have strong encryption.	None	***pkzipc -listcryptalgorithms***	standalone

Name/Description	*Value(s)*	*Example usage*	Used with
listfile 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.	Requires a name for the list file No default value.	Create a list file of files that the command line minus the **listfile** option would add to myarchive.zip: ***pkzipc -add=update -listfile=mylist.txt myarchive.zip **** 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 ***pkzipc -extract -listfile=list.txt save.zip***	add, extract
listhashalgorithms Displays a list of the hash algorithms available to the **hash** option. With **fipsmode** on, it lists only FIPS-validated algorithms.	None	***pkzipc -listhashalgorithms***	standalone
listsfxtypes Display a list of the types of SFX files that can be created with PKZIP	No sub-options. No default value.	***pkzipc -listsfxtypes***	standalone
listsponsors Available only for SecureZIP Partner Lists PKWARE PartnerLink sponsors defined to the system	No sub-options. No default value.	***pkzipr -listsponsors***	standalone
locale 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. Configurable	**enable** - Turns the option on **disable** - Turns the option off Default = **enable**	Configure the option to be off by default: ***pkzipc -config -locale=disable*** Turn the option off for the current command line ***pkzipc -add -locale=disable test.zip *.doc***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version

Name/Description	*Value(s)*	*Example usage*	Used with
log Causes PKZIP to log normal messages and specifies where to write them. Also causes SNMP traps for normal operations to be sent. Configurable	**stdout** - (Default) Writes messages to standard output **stderr** - Writes messages to standard error **syslog** - Writes messages to the system logging facility <**file name**> - Writes messages to a specified file. Each command line logged overwrites the file. **snmp** - If **SnmpTrapHost** is set, sends an SNMP trap for each normal operation that generates a message to STDOUT Default = **stdout**	***pkzipc -add -log=syslog -logoptions=start -jobid=my_id2 mytestlog.zip bookmark.htm*** Sends SNMP traps: ***pkzipc -add -log=snmp -logoptions=start -jobid=my_id2 -snmptraphost=nmsnode1 mytestlog.zip bookmark.htm***	All commands
logerror Causes PKZIP to log messages for errors and warnings and specifies where to write them. Also causes SNMP traps to be sent for error and warning conditions. Configurable	**stdout** - Writes messages to standard output **stderr** - (Default) Writes messages to standard error **syslog** - Writes messages to the system logging facility <**file name**> - Writes messages to a specified file. Each command line logged overwrites the file. **snmp** - If **SnmpTrapHost** is set, sends an SNMP trap for each error or warning condition Default = **stderr**	***pkzipc -add -logerror=syslog -logoptions=start -jobid=my_id2 mytestlog.zip bookmark.htm*** Sends SNMP traps: ***pkzipc -add -logerror=snmp -logoptions=start -jobid=my_id2 -snmptraphost=nmsnode1 mytestlog.zip bookmark.htm***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
logoptions Determines whether an entry appears in the system logging facility when PKZIP starts and/or stops, regardless of **log** or **logerror** settings. Also sends SNMP traps for these events if **snmptraphost** is set. Configurable	**none** – No extra messages will be logged. **start** - Puts an entry in the system logging facility each time PKZIP starts. The entry contains the current command line. Also sends an SNMP trap if **snmptraphost** is set. **stop** - Puts an entry in the system logging facility each time PKZIP stops. Also sends an SNMP trap if **snmptraphost** is set. **security** - Causes additional security information to be logged. **all** - includes Start, Stop and Security entries in system log.	***pkzipc -add -log=syslog -logoptions=start,stop -jobid=my_id2 mytestlog.zip bookmark.htm*** ***pkzipc -add -logoptions=start,stop -snmptraphost=nmsnode1 mytestlog.zip bookmark.htm***	All commands
lowercase Extracts file name(s) in lower case regardless of how it was originally archived. Configurable	**disable** – Do not change the case of extracted files. **archive -** extracts file name(s) such that only path information stored in the archive becomes lower case. **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. ---------- default = **archive**.	***pkzipc -extract –lowercase=full save.zip MixedCaseExtract/*** Extracts all files from save.zip using lower case letters to a directory named mixedcaseextract. That directory's name will have lower case letters. ***pkzipc -extract –lowercase=archive save.zip MixedCaseExtract/*** Extracts all files from save.zip using lower case letters to a directory named MixedCaseExtract. That directory's name will have mixed case letters.	extract

Name/Description	*Value(s)*	*Example usage*	Used with
lzma Compress files using the LZMA method. Configurable **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.	No sub-options. ---------- No default value.	***pkzipc -add -lzma save.zip doc1.txt***	add
mailBCC Specifies the email address of a recipient to be blind-copied (that is, sent a copy without appearing in any list of recipients) Configurable	<**email address**> - The email address of the recipient <**file name**> - Name of a file that contains a list of email addresses, one address to a line The file must be prefixed with the list character (@ by default) defined with the **listchar** option, to identify the file as a list file.	***pkzipc -add -mailTo=john.public@abc.com -mailTo=jane.doe@abc.com -mailSubject="Latest sales" -mailBody="Here are the figures I promised" -mailCC=rich.smith@abc.com -mailBCC=bill.cody@abc.com data.zip *.doc***	add, delete, header, comment, mailto
mailBody Specifies text for the message body of a message Configurable	<**body text**> - The message body text to use, set off by quotes <**file name**> - Name of a file that contains the text to use for the message body. The file name must be prefixed with the list character (@ by default) defined with the **listchar** option.	***pkzipc -add -mailTo=john.public@abc.com -mailSubject="Latest sales" -mailBody="Here are the figures I promised" data.zip *.doc*** ***pkzipc -add -mailTo=john.public@abc.com -mailSubject=@subject_text.txt -mailBody=@body_text.txt data.zip *.doc***	add, delete, header, comment, mailto

Name/Description	*Value(s)*	*Example usage*	Used with
mailCC Specifies the email address of a recipient to receive a copy of the message. Use this option once for each recipient to receive a copy. Configurable	<**email address**> - The email address of the recipient <**file name**> - Name of a file that contains a list of email addresses, one address to a line. The file name must be prefixed with the list character (@ by default) defined with the **listchar** option,	***pkzipc -add -mailTo=john.public @abc.com -mailSubject="Your data" -mailCC=rich.smith@abc .com data.zip *.doc*** ***pkzipc -add -mailTo=john.public @abc.com -mailSubject="Your data" -mailCC=@listfile.txt data.zip *.doc***	add, delete, header, comment, mailto
mailFrom Specifies the email address of the sender of the message. The address must be one that the SMTP server allows. This option and the **mailTo** and **mailServer** options are required: They must be configured or explicitly appear on the command line to send mail. Configurable	<**email address**> - The email address of the sender	***pkzipc -add -mailTo=john.public @abc.com -mailFrom=jane.doe @abc.com -mailSubject="Your data" -mailCC=rich.smith@abc .com data.zip *.doc***	add, delete, header, comment, mailto
mailOptions For use with the **mail…** options: Hides names in the TO list of a mail message; includes instructions on how to unzip Configurable	**each** - Causes each **mailTo** recipient to receive the message with only his own name appearing in the TO list. Each **mailCC** and **mailBCC** recipient receives a copy of each message to each **mailTo** recipient. **undisclosed** - Works just like the **each** sub-option except that the message that each recipient receives displays *Undisclosed* in the TO field instead of the recipient's name. **Undisclosed** is faster than **each**,	***pkzipc -add -mailTo=john.public @abc.com -mailTo=jane.doe@abc.com -mailOptions=each -mailSubject="Your data" -mailCC=rich.smith@abc .com data.zip *.doc*** Uses default value: ***pkzipc -add -mailTo=john.public @abc.com -mailTo=jane.doe@abc.com -mailOptions -mailSubject="Your data" -mailCC=rich.smith@abc .com data.zip *.doc***	add, delete, header, comment, mailto

Name/Description	*Value(s)*	*Example usage*	Used with
	which must get recipient names. **instructions** - Causes PKZIP to include a small, additional attachment explaining how to unzip a ZIP file **all** - Turns on both of the sub-options described above **none** - (Default) Turns off any sub-options Default = **none**		
mailReplyTo Specifies an alternate email address for recipients to use instead of the **mailFrom** address to reply to the message Configurable	<**email address**> - The email address to use for replies	***pkzipc -add -mailTo=john.public @abc.com -mailFrom=jane.doe @xyz.com -mailSubject="Plans" -mailreplyTo=jane.doe @myplace.net plans.zip *.doc***	add, delete, header, comment, mailto
mailServer Specifies the name or IP address of the SMTP server. This option and the **mailTo** and **mailFrom** options are required: They must be configured or explicitly appear on the command line to send mail. Configurable	<**server**> - The name or IP address of the server, for example, `mail01`, or `mail.abc.com` <**user:pass@server**> - Tells PKZIP to try plain-text authentication to connect to the server, using the supplied user name and passphrase. The user name and passphrase are both optional. For example, either of these works: mailserver=user@mail.abc.com Note the colon prefixing the passphrase: mailserver=:pass@mail.abc.com	***pkzipc -add -mailto=tom.jefferson @wash.com -mailfrom=sam.adams @wash.com -mailserver=mail01 files.zip *.doc*** ***pkzipc -add -mailto=tom.jefferson @wash.com -mailfrom=sam.adams @wash.com -mailserver=sama@mail01 files.zip *.doc*** ***pkzipc -add -mailto=tom.jefferson @wash.com -mailfrom=sam.adams @wash.com -mailserver=sama:passwd @mail01 files.zip *.doc***	add, delete, header, comment, mailto

Name/Description	*Value(s)*	*Example usage*	Used with
mailSubject Specifies text for the Subject line of an email message header If this option is omitted, the text *Sending <archive name>* is used. Configurable	<**subject text**> - The message subject text to use, set off by quotes <**file name**> - A file name that contains the message subject text. The file must be prefixed with the list character (@ by default) defined with the **listchar** option.	***pkzipc -add -mailTo=john.public@abc.com -mailSubject="Your data" -mailCC=rich.smith@abc.com -mailOptions=instructions data.zip *.doc*** ***pkzipc -add -mailTo=john.public@abc.com -mailSubject=@subject_text.txt -mailBody=@body_text.txt data.zip *.doc***	add, delete, header, comment, mailto
mailTo Specifies the email address of a recipient. Use multiple times to specify multiple recipients. Use with **cryptalgorithm** to apply certificate-based encryption to attached files for all recipients. This option and the **mailFrom** and **mailServer** options are required: Values for these options must be configured or be specified on the command line to send mail.	<**email address**> - The email address of the recipient <**file name**> - A file name that contains a list of email addresses, one address to a line. The file must be prefixed with the list character (@ by default) defined with the **listchar** option. **recipient** - Uses email addresses associated with certificates configured for the **recipient** option (as distinct from the **recipient** sub-option of **mailTo**). This sub-option can be used only when **mailTo** is used as an option with another command such as **add**.	As a command: ***pkzipc -mailto=tom.jefferson@wash.com -mailfrom=sam.adams@wash.com files.zip*** As an option: ***pkzipc -add -mailto=tom.jefferson@wash.com -mailto=ulysses.grant@wash.com -mailfrom=sam.adams@wash.com files.zip *.doc*** ***pkzipc -add -mailto=@to_list.txt -mailfrom=sam.adams@wash.com files.zip *.doc*** ***pkzipc -add -mailto=recipient -mailserver=mail01 -mailfrom=sam.adams@wash.com files.zip *.doc*** Encrypts files: ***pkzipc -add -recipient=tom.jefferson@wash.com -recipient=sam.adams@wash.com -mailTo=recipient -mailserver=mail01 -mailfrom=sam.adams@wash.com files.zip *.doc***	add, del, header, comment, standalone

Name/Description	*Value(s)*	*Example usage*	Used with
mask Strips file attributes that the attribute option would otherwise cause to be stored or set for extracted files Configurable (Windows) **Note:** You can only mask attributes that are specified using the attributes option.	**hidden** - hidden attributes. **archive** - archive attribute. **system** - system attributes. **readonly** - read-only attributes. **none** - no attributes (turns off attribute mask in the PKZIP Configurations Settings file for this instance only). **all** - all attributes **<hex value>** -The hex value of an attribute to be masked, or the logical OR of multiple hex values ---------- Default (add) = **none** Default (extract) = **all** Value if used on command line without a sub-option (add and extract) = **all**	***pkzipc -add -attributes=all -mask=hidden save.zip*** ***pkzipc -extract -mask=none save.zip*** ***pkzipc -config -mask=hidden***	add, extract
mask Specifies a permissions mask for files added or extracted. When extracting, you can use the option with the permission option to explicitly strip permissions that would otherwise be set. Configurable (UNIX only)	**<octal value>** - A value in octal which represents the permissions which should NOT be restored. For example, to prevent files from being extracted with any execute permissions, specify a mask value of 111.	***pkzipc -extract -mask=111 save.zip***	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
maximum 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 Configurable	No sub-options. --- No default value.	***pkzipc -add -maximum save.zip *.doc*** ***pkzipc -config -maximum***	add
messagedigest Display one or more message digests for files inside an archive.	**All** – Calculates and displays the message digest for all of the algorithms. **None** – Don't display any message digests or checksums. This is useful for displaying only one. **CRC32** – Calculates and displays CRC32 checksum **MD5** – Calculates and displays MD5 message digest. **SHA1** – Calculates and displays SHA-1 message digest. **SHA256** – Calculates and displays SHA-256 message digest. **SHA384** – Calculates and displays SHA-384 message digest. **SHA512** – Calculates and displays SHA-512 message digest. Prefix an algorithm with - to indicate it should not be used --- Default = all	Shows the message digest using all available hash algorithms for all files inside archive.zip ***pkzipc -messagedigest archive.zip*** Shows the message digest using all available hash algorithms for only file.doc inside archive.zip ***pkzipc -messagedigest archive.zip file.doc*** Uses the sha256sum program to verify that file.doc inside archive.zip is the same as file.doc in the current directory. ***pkzipc -messagedigest=none,sha256 -silent=banner archive.zip file.doc | sha256sum --check***	Standalone

Name/Description	*Value(s)*	*Example usage*	Used with
more Pauses after one screen of output and prompts to continue. Configurable	The number of rows of information you want to define as a screen --- Default = one screen of information	***pkzipc -view -more=22 save.zip*** ***pkzipc -config -more***	All commands
move Removes (deletes) files from the source drive after adding them to an archive.	No sub-options. --- No default value.	***pkzipc -add -move save.zip *.doc***	add
movearchive 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, or to be transferred by FTP. Configurable	No sub-options. --- No default value.	***pkzipc -add -encode=gzip -movearchive myfiles.tar***	add
mt Use multiple threads when creating archive.	The number of threads to use, defaults to the number of processing cores on the system.	Create test.zip using multiple threads. ***pkzipc -add -mt test.zip *.doc*** Create test.zip using up to 3 threads. ***pkzipc -add -mt=3 test.zip *.doc***	add
namesfx Specify a file name when converting to a self-extracting file.	**<file name>** - File name for the SFX file --- No default value.	***pkzipc -sfx -namesfx=test.exe docs.zip***	sfx

Name/Description	*Value(s)*	*Example usage*	Used with
newer Selects files that are no older than a specified interval Configurable separately for add and extract operations **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. **Note:** To specify an explicit date, see **after**..	**<numeric value>** A number of days, hours, minutes, or seconds defining the interval, plus a suffix identifying the kind of units used: Suffixes: **d** - Days (default) **h** - Hours **m** - Minutes **s** - Seconds --- No default value.	Add files no older than 24 hours: ***pkzipc -add -newer=24h save.zip **** Add files no older than five days: ***pkzipc -add -newer=5d save.zip **** ***pkzipc -add -newer=5 save.zip ****	add, extract, test, view, print, console
noarchiveextension Suppresses adding a file name extension to the specified archive file name Configurable **Note:** This option is identical to **nozipextension**, which is now deprecated.	No sub-options. --- No default value.	***pkzipc -add -noarchiveextension file.ibm *.doc***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
noextended Suppress the storage of extended attribute information (excluding file permission attributes Configurable	No sub-options. --- No default value.	***pkzipc -add -noextended save.zip ****	add
nofix Suppress the attempt to fix any problems PKZIP encounters in extracting from an archive Configurable	No sub-options. --- No default value.	***pkzipc -add -nofix save.zip *.doc***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version

Name/Description	*Value(s)*	*Example usage*	Used with
normal 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 **fast** option, all files are compressed. Configurable	No sub-options. --- No default value.	***pkzipc -add -normal save.zip*** ***pkzipc -config -normal***	add
nosmartcard **Note:** This option is deprecated. Instead of setting **nosmartcard**, turn off the **smartcard** sub-option of **cryptoptions**. Turns off smart card compatibility when set in conjunction with the **recipient** option. Set this option to enable users of versions of PKZIP prior to 6.1 to decrypt files encrypted using the **recipient** option. **Note:** Smart cards cannot decrypt files encrypted using a recipient list if this option is set. Configurable	No sub-options. --- No default value.	***pkzipc -add -recipient="Thomas Francis, Jr." nosmartcard save.zip *.doc***	add
nozipextension **Note:** This option is deprecated. Use the option **noarchiveextension** instead. Suppress PKZIP's adding of an identifying file extension to an archive file name Configurable	No sub-options. --- No default value.	***pkzipc -add -nozipextension file.ibm *.doc***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
older Selects files that are older than a specified interval Configurable separately for add and extract operations **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. **Note:** To specify an explicit date, see **before**.	**<numeric value>** A number of days, hours, minutes, or seconds defining the interval, plus a suffix identifying the kind of units used: Suffixes: **d** - Days (default) **h** - Hours **m** - Minutes **s** - Seconds ---------- No default value.	Adds files older than 24 hours: ***pkzipc -add -older=24h save.zip **** Adds files older than five days: ***pkzipc -add -older=5d save.zip **** ***pkzipc -add -older=5 save.zip ****	add, extract, test, view, print, console
OpenFile Determines whether to include files that are open for write access in another application **Note**: This option is not needed in UNIX and Linux systems, as this is the default behavior.	**Never -** PKZIP does not include any open files. A warning will appear if a matching file is open **All -** PKZIP includes all matching open files without prompting first. A message noting each open file is included in the standard output. **Prompt -** PKZIP notifies you when a matching file is open, and asks whether to add the open file or skip it. ---------- Default = ***Never***	***pkzipc -add -OpenFile=never test.zip *.bmp*** ***pkzipc -add -OpenFile test.zip *.bmp*** ***pkzipc -add -OpenFile=prompt test.zip *.bmp***	add

Name/Description	*Value(s)*	*Example usage*	Used with
optionchar Specifies the prefix character used to identify a command or option as such on the command line **Note:** On Windows, the "/" (slash) character can also always be used. Configurable	Any valid single character. --- Default = - (hyphen)	***pkzipc -optionchar=+ +add save.zip *.doc*** ***pkzipc +config -optionchar=+***	All commands
overwrite Specifies whether to overwrite existing files with files being added or extracted. By default, PKZIP prompts before overwriting when extracting but not when adding. Configurable	**prompt** - Prompt every file individually on whether to overwrite a file that has the same name as the one being added or extracted **all** - Overwrite all files that have the same name **increment** - Increment file name to make it unique. **never** - Never overwrite a file that already exists in the target directory or archive --- Value if used on command line without a sub-option = **all**.	***pkzipc -extract -overwrite=all save.zip*** ***pkzipc -add -overwrite=prompt save.zip***	add, extract
owner Changes files' associated UID and/or GID to a specified UID or GID Configurable (UNIX only)	**<owner>:<group>** - Specifies files' owner and group **<owner> -** Specifies files' owner; leaves group unchanged **<group> -** Specifies files' group; leaves owner unchanged	***pkzipc -extract -owner=jon:eng test.zip*** ***pkzipc -extract -owner=jon test.zip*** ***pkzipc -extract -owner=jon test.zip*** ***pkzipc -add -owner=0:0 test.zip***	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
passphrase Protects an archive with passphrase-based encryption PKZIP prompts for a passphrase if none is specified with the option. Configurable	**<passphrase>** - The passphrase that must be supplied to extract and decrypt the files <**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. --- No default value.	To include a passphrase in the command: ***pkzipc -add -passphrase=beowulf save.zip*** To have PKZIP prompt for a passphrase after you type the command: ***pkzipc -add -passphrase save.zip*** To have PKZIP get the passphrase from a file: ***pkzipc -add -passphrase=@secret.txt save.zip*** To extract passphrase-protected files from an archive: ***pkzipc -extract -passphrase=beowulf9 save.zip***	add, extract, test, print, console
path Stores or restores directory path names for files within a .ZIP file By default, PKZIP does not store path information Configurable **Note:** UNIX users should use the include option or place quotation marks around wildcard designations to bypass automatic wildcard expansion by the shell, which may restrict your pattern search. See “Using Wildcards with PKZIP on UNIX” in Chapter 2.	**current** - Store the path from the current directory. **root** or **full** - Store the entire path beginning at the root of the drive; also referred to as "full" path. **specify** or **relative** - Store path information relative to the specified directories, for their subdirectories **none** - No path information stored --- Default = **none** when used with **add**; **full** when used with **extract** Value if used on command line without a sub-option = **current**	Assuming you are in "/temp": ***pkzipc -add -path=root save.zip docs/**** (the complete path is stored including "temp/docs/"). ***pkzipc -add -path=current save.zip docs/wp/**** (the path stored is "docs/wp").	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
permission Restores and sets extra permissions when extracting files. Normally, the setuid, setgid, and sticky bit permissions are not restored when extracting archives. Using this option restores them. Configurable (UNIX) **Note:** PKZIP restores any read, write, and execute permission attributes by default. The permission option is necessary only if you wish to restore additional attributes (such as *setuid*, *setgid*, and *sticky* bits) or different ones.	Octal mode value. --- No default value.	Preserve all permissions and other attributes on extraction: ***pkzipc -extract -permission save.zip*** Preserve and/or attempt to modify all permissions and other attributes on extraction: ***pkzipc -extract -permission=4111 save.zip***	extract
pgpPublicKey Specify the file containing PGP public keys.	File containing PGP public keys.	***pkzipc -add -pgpPublicKey=pgpkeys.pkr test.pgp -recipient="John Smith" *.doc***	add
pgpSecretKey Specify the file containing PGP secret keys.	File containing PGP secret keys.	***pkzipc -extract -pgpSecretKey=pgpkeys.skr test.pgp***	add extract
pkcs11 Enables access to encryption and decryption keys from secure storage devices (PKCS#11). These include Hardware Security Modules (HSM) or Smart Cards. **Note**: This command only appears if you have altconfig configured to point to the PKCS#11 device.	No sub-options. --- No default value.	List all available certificates, including those on PKCS#11 devices. ***pkzipc -listcert -pkcs11***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
ppmd Compress files using the PPMd method. Configurable **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.	No sub-options. --- No default value.	***pkzipc -add -ppmd save.zip doc1.txt***	add
preview Prints out messages to preview the results of a set of commands or options without actually performing the tasks	No sub-options. --- No default value.	***pkzipc -add -preview save.zip***	add, delete, header, sfx, comment
print Print a file within a .ZIP file. (Windows)	**<print device>** - The print device use, for example, "lpt1". --- Default = the default printer on your system.	***pkzipc -print=lpt1 save.zip readme.txt*** Uses default printer if no printer is specified.	standalone
priority Sets the execution priority of PKZIP with regard to other programs running on the same system Configurable, but must be included on the command line to take effect **Note:** Only users with appropriate rights or privileges can raise priority of execution.	**<priority level>** - Windows: Low BelowNormal Normal AboveNormal High UNIX: A value in the range 0-39 Default = Normal (Windows) Default = 20 (UNIX)	Windows: ***pkzipc -add mydocs.zip *.doc -priority=normal*** UNIX: ***pkzipc -add mydocs.zip *.doc -priority=10***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
recipient Specifies one or more recipients for certificate-based encryption. The option can appear more than once on the command line to specify multiple recipients. Configurable **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. **Note:** This option is available only with SecureZIP.	**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. **<Friendly name>** - The friendly name associated with the certificate. This is often the same as the common name of the subject. **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. **f=<ldap filter>** - An LDAP filter to use to filter a search for certificates on an LDAP server that you are accessing with the ***ldap*** option. **@<file name>** - Specifies a text file which contains the names of recipients, one on each line. **#<file name>** - Specifies a PKCS#7 or PKCS#12 file that contains certificates of the recipients you want to list. --- Default = **cn=**	***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.p7b save.zip *.doc*** ***pkzipc -add -recipient=#recipients.p12 save.zip *.doc***	add

Name/Description	*Value(s)*	*Example usage*	Used with
recurse Search subdirectories for files to compress Use with path to store path information for files in subdirectories. Tip: You can use ***directories*** to combine the functionality of **recurse** and **path**. Configurable **Note:** UNIX users should use the include option or place quotation marks around wildcard designations to bypass automatic wildcard expansion by the shell, which may restrict your pattern search.	No sub-options. --- No default value.	***pkzipc -add -recurse save.zip ****	add
rename Uses regular expressions to rename files as they are added or extracted.	**@<list file>** - A list file specifying replacement expressions, one on each line **<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	***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***	add, extract, test

Name/Description	*Value(s)*	*Example usage*	Used with
runafter Run or open a specified file after extraction by a self-extractor Configurable	**<file name>** - The file to run or open --- No default value.	Launch the file (for example, readme.txt) via the specified applications (for example, notepad.exe): ***pkzipc -add -sfx -runafter="notepad.exe readme.txt" test.exe **** Launch the file (for example, readme.txt) via the associated application (Windows only): ***pkzipc -add -sfx -runafter="${} readme.txt" test.exe **** Run the install script (for example. install.inf) (Windows only): ***pkzipc -add -sfx -runafter="${install.inf}" test.exe **** Run the install script (for example, install.inf) with the full short path pre-appended (for example, c:\program~1\temp) (Windows only): ***pkzipc -add -sfx -runafter="${install}%0install.inf" test.exe***	(add) sfx
RunContext Specifies that program is running in the context of the Current User or Local Machine This option is 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. (Windows)	**user** - program is running in the context of the Current User **machine** - program is running in the context of the Local Machine --- Default = user	***pkzipc -extract archive.zip*** Extracts archive in the context of the Current User ***pkzipc -extract -RunContext archive.zip*** Extracts archive in the context of the Current User ***pkzipc -extract -RunContext=user archive.zip*** Extracts archive in the context of the Current User ***pkzipc -extract -RunContext=machine archive.zip*** Extracts archive in the context of the Local Machine	extract

Name/Description	*Value(s)*	*Example usage*	Used with
sftp Transfers an archive using SSH File Transfer Protocol Configurable (UNIX only)	Syntax (optional fields in brackets): **-ftp=** [username[:pwd [:account]]@]server /path where: **username** (optional) is the user account with which to log in to the SFTP server **pwd** (optional) is the passphrase associated with the user account. Colons are not allowed in the passphrase. **account** (optional) is for use only with FTP servers that require additional authentication. Do not specify the account for servers that do not require it. **server** is the FTP server name **path** is the path to the destination of the transferred file on the server. Use two slashes (server//path) to specify a full path; use one slash (server/path) to specify a path relative to the login destination directory.	***pkzipc -add -sftp=serv/home/thomas mydocs.zip *.doc*** ***pkzipc -add -sftp=me@serv/home/thomas mydocs.zip *.doc*** Standalone: ***pkzipc -sftp=serve/home/thomas mydocs.zip***	add, delete, header, comment, standalone

Name/Description	*Value(s)*	*Example usage*	Used with
sfx 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. Configurable **Note:** For a listing of available self-extractors, use the **listsfxtypes** command.	**<no sub-option>** - Create a native command line self-extractor **win32_x86_g610** - Create a graphical Windows self-extractor that, when run, opens a dialog to let the user select a target extract folder --- Default = Create a native command line self-extractor for use in the command line environment of the operating system in which it was created	To create myfiles.exe: ***pkzipc -add -sfx myfiles *.doc*** To convert existing ZIP file myfiles.zip to self-extracting graphical Windows archive myfiles.exe: ***pkzipc -sfx=win32_x86_g myfiles.zip*** To convert existing ZIP file myfiles.zip to a self-extractor and specify a name for the self-extractor: ***pkzipc -sfx -namesfx=newname myfiles.zip*** (Converts myfiles.zip to newname.exe.)	add, standalone
sfxdestination Specifies a default target folder for files extracted from a self-extractor Configurable	**<path>** - Path to target folder --- No default value	***pkzipc -add -sfx -sfxdestination="My Documents\newstuff" mysfx *.doc***	add, sfx
sfxdirectories 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. Configurable	No sub-options --- No default value	***pkzipc -add -sfx -sfxdirectories -directories mysfx "docs*.*"***	add, sfx
sfxlogfile Creates an ASCII text error log (named pkerrlog.txt) in the destination directory on extraction Configurable	No sub-options --- No default value	***pkzipc -add -sfx -sfxlogfile test.exe ****	(add) sfx

Name/Description	*Value(s)*	*Example usage*	Used with
sfxoverwrite Specifies when a self-extractor overwrites files that have the same name as a file being extracted Configurable	**prompt** - (Default) The user is asked whether to overwrite files **always** - Files that have the same name in the destination folders are overwritten without prompting **update** - Only files that do not already exist or are newer than same-named files **freshen** - Only newer versions of files that already exist in the destination folders are extracted; the older files are overwritten without prompting **never** - Files are never overwritten --- Default = **prompt**	***pkzipc -add -sfx -sfxoverwrite=freshen mysfx *.doc***	add, sfx
Sfxtitle Specifies the title to use for the graphical interface (GUI) that a self-extractor presents to the user. This option only affects GUI self-extractors. (Command line self-extractors do not present a GUI.) Configurable	**Title** – The title to display	***pkzipc -add -sfx -sfxuitype=regularsfx -sfxtitle="My Self-Extractor" mysfx *.doc***	

Name/Description	*Value(s)*	*Example usage*	Used with
sfxuitype Specifies the type of graphical interface (GUI) that a self-extractor presents to the user. This option only affects GUI self-extractors. (Command line self-extractors do not present a GUI.) Configurable	**autosfx** - Presents a dialog that displays a bar to show progress extracting, and a Cancel button **easysfx** - (Default) Presents a dialog that enables the user to select a destination folder and to turn off any **runafter** option set **regularsfx** - Presents a dialog that enables the user to change the destination folder and other options before the archive is extracted ---------- Default = **easysfx**	***pkzipc -add -sfx -sfxuitype=regularsfx mysfx *.doc***	add, sfx
shortname Convert long file names of files added to an archive to WIN32-equivalent "short" file names Configurable	**dos** - Convert long file names to DOS-equivalent short file names (8+3) **none** - Do not convert file names ---------- Default = **none** Value if used on command line without a sub-option = **dos**.	***pkzipc -add -short=dos save.zip***	add

Name/Description	*Value(s)*	*Example usage*	Used with
shred Overwrites PKZIP temporary files and files deleted by PKZIP to prevent recovery of their data Configurable **Note:** The ***shred*** option is the same as the older ***wipe*** option, which is now deprecated.	**none** - turns off shredding: files are not overwritten **random** - Overwrites files once with random data **dod5220** - Overwrites files three times, to the DOD 5220.22-M specification **nsa** - Overwrites files seven times, to the NSA standard --- Default = **none** Value if used on command line without a sub-option = **random**	***pkzipc -add -move -shred=nsa myfiles.zip ****	add
sign 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. For maximum security, sign both the central directory and local files. Configurable **Note:** This option requires SecureZIP.	**cd** - sign central directory. **files** - sign files. **all** - sign both the central directory and files. **timestamp** – Sign files and apply a digital timestamp to the central directory. **none** - do not sign files (Used for turning signing off if it has been configured) --- Value if used on command line without a sub-option = **all**.	***pkzipc -add -certificate="John Smith" -sign=cd save.zip *.doc***	add, standalone

Name/Description	*Value(s)*	*Example usage*	Used with
silent 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. Configurable	**none** - Turns off the silent option; displays all messages **banner** - Suppresses printing the banner **copy** - Suppresses "Copy file" messages when updating archives **error** - Suppresses all error and warning outputs **fileheader** - Suppresses file headers when using the **console** command **input** - Suppresses all requests for input. If any operation requests input, an error is given **normal** - Suppresses all message outputs except warnings, errors, and prompts for input **output** - Suppresses all normal, error, and warning outputs **progress** - Suppresses "percent complete" messages **all** - Same as specifying both Input and Output. (Default if option is specified without a sub-option) ---------- No default value.	***pkzipc -add -silent save.zip *.doc*** ***pkzipc -config -silent***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version

Name/Description	*Value(s)*	*Example usage*	Used with
smaller Process only files that are smaller than or equal to a given file size, specified in bytes Configurable separately for add and extract operations.	Numerical value that indicates a maximum desired file size Acceptable notation: K - 1024B M - 1024K G - 1024M T - 1024G ---------- No default value.	***pkzipc -add -smaller=5000 save.zip **** In this example, PKZIP adds only files no larger than 5000 bytes in size.	add, extract, test, view, delete, print, console
snmptraphost Specifies an SNMP host machine running an SNMP receiver for PKZIP to send SNMP traps to. To specify traps to send, set sub-options for the ***log***, ***logoptions***, and ***logerror*** options. Configurable	Syntax (optional fields in brackets): **-snmptraphost=** -snmptraphost =[community@] host[:port] where: **community** (optional) is the community name; default is *public* **host** is the SNMP host name or IP address **port** (optional) is the port number. The default SNMP trap port is 162.	***pkzipc -add mydocs.zip *.doc -snmptraphost=nmsnode1*** ***pkzipc -extract backup1.zip -snmptraphost =private@hostxyz:20001***	All commands

Name/Description	*Value(s)*	*Example usage*	Used with
sort 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. Configurable **Note:** The **crc** and ***ratio*** sub-options do not work with the **add** command and **sort** option.	**crc** - sort by CRC value **date** - sort by file date of file **extension** - sort by file extension **name** - alphabetically sort files and folders together in one series by path name **natural** - sort in the order files occur in the archive **ratio** - sort by compression ratio **size** - sort by the original, uncompressed size of the file ("length" in display) **comment** - sort by file comment **none** - first alphabetically sort path names that contain folders and then separately sort file names that lack folder information. (The default.) ---------- Default = **none** Value if used on command line without a sub-option = **name**	***pkzipc -add -sort=date save.zip *.doc*** ***pkzipc -config -sort=date***	add, extract, test, view, delete, print, console

Name/Description	*Value(s)*	*Example usage*	Used with
span Forces PKZIP to create a split archive, even when creating the archive on non-removable media. Also formats or wipes removable media prior to writing an archive. On Windows (only), the option causes PKZIP to write an archive in segments if necessary to span multiple removable media. This option is available only for ZIP archives. **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. On UNIX, this option only splits an archive into segments on the hard drive; disk spanning and the sub-options are not available. Configurable	**Force** - Fully format media without checking for existing files **Format** - Fully format media before attempting to write to it **Quick** - Quick-format media before attempting to write to it **Wipe** - Delete contents of media before attempting to write to it **None** - Do not format or erase media before attempting to write to it **<segment size>** - Split archive into segments of predefined size (see choices below) or a specified size (in bytes) greater than 65535. Predefined sizes: **360** = 360KB floppy **720** = 720KB floppy **1.2** = 1.2MB floppy **1.44** = 1.44MB floppy **2.88** = 2.88MB floppy. **95.7** = 100MB ZIP disk **650** = 650MB CD-ROM **700** = 700MB CD-ROM Acceptable notation: K - 1024B M - 1024K G - 1024M T - 1024G ---------- Default = **none**	***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***	add

Name/Description	*Value(s)*	*Example usage*	Used with
speed 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. 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 Configurable	No sub-options. --- No default value.	***pkzipc -add -speed save.zip *.doc*** ***pkzipc -config -speed***	add
sponsor Available only for SecureZIP Partner Specifies a PKWARE PartnerLink sponsor. Used to determine certificates to use to authenticate a sponsor signature or to encrypt for as recipients. Configurable	**<sponsor name>** - The common name of the sponsor **<sponsor ID>** - The sponsor ID	***pkzipr -add myfiles.zip -sponsor="Example Corp" *.doc*** ***pkzipr -add myfiles.zip -sponsor=15 *.doc***	add, extract
store Sets the level of compression to 0 (no compression) on a scale of 0 - 9; stores the files in the archive without compressing them Configurable	No sub-options. --- No default value.	***pkzipc -add -store save.zip *.doc*** ***pkzipc -config -store***	add

Name/Description	*Value(s)*	*Example usage*	Used with
stream Gets file data to or from pipe or socket special files. Without this option, **add** gets only file definitions (name, permissions, times, and so on) but not data. The **extract** command used with this option extracts to special files instead of overwriting them with ordinary files. Use **add** with **filetype** to have PKZIP operate on special files. (UNIX)	No sub-options ---------- No default value	Save data from pipe mystream to file mystream in archive: ***pkzipc -add -filetype=pipe -stream data.zip mystream*** Extract file mystream to special file mystream (if such file exists) else to an ordinary file named mystream: ***pkzipc -extract -stream data.zip mystream***	add, extract
strict 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). Configurable	**KeyUsage** - Controls key usage checks **TimeNesting** - Controls time nesting checks **TimeValid** - Controls time validity checks ---------- No default value	***pkzipc -add -cryptalg -recipient="John Q. Public" -strict test.zip *.doc*** ***pkzipc -add -recipient="John Q. Public" -crl -strict test.zip *.doc***	add, delete, comment, header

substitution 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**, 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. Configurable	Available tokens. These are replaced by associated values on execution: **{archivename}** - (For use only with **extract**) Base name of archive, without the extension **{archiveext}** - (For use only with **extract**) The file name extension of the archive (without a leading dot) **{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 **{id}** - A job ID specified separately with the jobid option **{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)	This command line using tokens: ***pkzipc -add -substitution "Design Spec {yyyy}-{mm}-{dd}-{h}-{MM}-{SS}{ampm}.zip" plan.doc*** produces a ZIP file with a name like: Design Spec 2006-08-09-12-06-29am.zip This command line uses the *jobid* option to set a value for {id}: ***pkzipc -add -jobid=myJob -substitution {id}{yyyy}.zip *.doc*** and results in a ZIP file with a name like: myJob2006.zip Extracts all ZIP files in the current directory, each to a subdirectory named after the ZIP archive extracted there ***pkzipc -extract -substitution *.zip {archivename}***	add, extract

Name/Description	*Value(s)*	*Example usage*	Used with
	{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 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: ***pkzipc -add -temp=z:\public test.zip *.txt*** Updates the .ZIP file test.zip and uses the /temp directory location for temporary files: ***pkzipc -add -temp=/temp test.zip *.txt***	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**	***pkzipc -test save.zip***	standalone

Name/Description	*Value(s)*	*Example usage*	Used with
timeout Sets a number of seconds for PKZIP to wait for another process to send or be ready to receive (more) data on a socket or block device. Configurable (UNIX)	**<seconds>** -The period of the timeout in seconds --- Default value if used without a specified number of seconds: 30 seconds	***pkzipc -extract -noArchiveExtension -timeout=60 mysocket***	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
times Specifies that PKZIP should restore the extended time fields, and/or other dates stored in the archive. Configurable	**access** - restores the time of last access to file(s) on extraction. **modify** - restores the time of last modification to files on extraction. **create** - restores the time of creation to files on extraction (Windows). **all** - all file times are restored. **none** - file times are not restored. --- Default = **all**	***pkzipc -extract -times=access save.zip***	extract

Name/Description	*Value(s)*	*Example usage*	Used with
translate 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. 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. Configurable	**none** - no translation is performed. **dos** - translates text files so that lines end with a return/newline pair (Windows default) **mac** - translates text files so lines end with a single carriage return **unix** - translates text files so lines end with a single newline **ebcdic,nl** - With ZDW files, substitute EBCDIC newline (0x15) **ebcdic,lf** - With ZDW files, substitute EBCDIC linefeed (0x25) **ebcdic,crlf** - With ZDW files, substitute EBCDIC carriage return/linefeed (0x0D25) **ebcdic,lfcr** - With ZDW files, substitute EBCDIC linefeed/carriage return (0x250D) **ebcdic,crnl** - With ZDW files, substitute EBCDIC carriage return/newline (0x0D15) **remove** - Remove end of line marks --- Default = **none** Value if used on command line without a sub-option = native operating system compatibility translation.	***pkzipc -extract -translate=unix save.zip*** ***pkzipc -add -translate=unix scripts.zip *.pl***	add, extract, console, print

Name/Description	*Value(s)*	*Example usage*	Used with
ts Contact a Time Stamp Authority (TSA) with the supplied URL to apply a digital timestamp to the archive. Configurable	Syntax (optional fields in brackets): **ts=** [username[:pwd @]server [:port]/]page where: **username** (optional) is the user account with which to log in if the FTP server requires a login **pwd** (optional) is the passphrase associated with the user account. Colons are not allowed in the passphrase. **server** is the TSA server name **port** is the TCP/IP port to use. No sub-options	Add files and digitally sign and timestamp the archive test.zip using the "My Name" certificate: ***pkzipc -add -sign=timestamp -certificate="My Cert" -ts=http://tsa.example.com/tsa test.zip *.txt*** Digitally sign and timestamp archive the archive test.zip using the "My Name" certificate ***pkzipc -sign=timestamp -ts=http://tsa.example.com/tsa -certificate="My Name" test.zip***	sign
usePGPName Derive the file name of a single file OpenPGP archive from the archive name, ignoring the original file name within the archive Configurable	No sub-options. No default value.	***pkzipc -extract -usePGPName sample.txt.pgp***	extract, test
utf8 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 Configurable	No sub-options. No default value.	***pkzipc -add -utf8 test.zip *.****	add, comment

Name/Description	*Value(s)*	*Example usage*	Used with
VerifyEncryption Verify that the encryption within the archive matches one or more of the specified encryption criteria.	**none** - Files are not encrypted **pkware** - Files are using PKWARE encryption **aex** - Files are using AE-x encryption **passphrase** - Files are using strong passphrase encryption **recipient** - Files are using strong recipient encryption **nopassphrase** - Files are not using passphrase based encryption **norecipient** - Files are not using recpient based encryption	***pkzipc -verifyEncryption=none test.zip*** Verify that none of the files in test.zip are encrypted. ***pkzipc -verifyEncryption=passphrase,recipient test.zip*** Verify that files are using either passphrase or recipient based encryption. ***pkzipc -verifyEncryption=nopassphrase test.zip "-include=*.txt"*** Verify that none of the *.txt files in test.zip are using passphrase based encryption.	standalone
verifyrecipient Verifies that an archive is encrypted for one of the X.509 certificates or OpenPGP keys specified.	**#<file name>** - Specifies a PKCS#7 file that contains the certificates. **cn=<common name>** - The Common Name (CN) field of the subject of the certificate. The “cn=” prefix is optional. **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. **f=<ldap filter>** - An LDAP filter to use to filter a search for certificates on an LDAP server that you are accessing with the ***ldap*** option. **kid<OpenPGP keyid>** - Specifies the keyid of an OpenPGP key.	***pkzipc -verifyEncryption=recipient -verifyRecipient="My Friend" test.zip*** Verify the archive is encrypted for recipients using the certificate for 'My Friend' ***pkzipc -verifyEncryption=recipient -verifyRecipient=bob.smith@pkware.com -verifyRecipient="My Friend" test.zip*** Verify the archive is encrypted for recipients using the certificates for 'My Friend' or 'Bob Smith' ***pkzipc -verifyEncryption=recipient -archiveType=pgp -verifyRecipient=31C47555 test.pgp*** Verify the archive is encrypted for the OpenPGP keyID 0x31C475555	verifyEncryption

Name/Description	*Value(s)*	*Example usage*	Used with
	<short OpenPGP keyid> - 8-digit hex string specifying a short OpenPGP keyid **<long OpenPGP keyid>** - 16-digit hex string specifying a long OpenPGP keyid **<X.509 public key hash>** - 20-digit hex string specifying the SHA1 hash of the public key for an X.509 certificate ---------- Default = **e**.		
verifysigner 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 or, using the *ldap* option, on LDAP. The option can appear more than once in the same command line, to specify multiple certificates. Configurable	**cn=<Common name>** - The Common Name (CN) field of the subject of the certificate. The "cn=" prefix is optional. **<Friendly name>** - The friendly name associated with the certificate. This is often the same as the common name of the subject. **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. **f=<ldap filter>** - An LDAP filter to use to filter a search for certificates on an LDAP server that you are accessing with the ***ldap*** option. **@<file name>** - Specifies a text file which contains a list of certificates, one on each line. **#<file name>** - Specifies a	***pkzipc -extract -verifysigner="Thomas Jones, Jr." save.zip *.doc*** ***pkzipc -extract -verifysigner="cn=Thomas Jones, Jr." save.zip *.doc*** ***pkzipc -extract -verifysigner=e=john.public@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 -***	extract

Name/Description	*Value(s)*	*Example usage*	Used with
	PKCS#7 or PKCS#12 file that contains the certificates themselves. --- No default value.	***verifysigner=#recipients.p7b save.zip *.doc*** ***pkzipc -extract -verifysigner=#recipients.p12 save.zip *.doc***	
version Gives information about the version of the release. Displays complete version information; also returns to the shell particular version numbers specified by sub-options.	**major** - Returns the major release number. For example, if the version number is 12.10.1054, the value returned is 12. **minor** - Returns the minor number of the release. For example, if the version number is 12.10.1054, the value returned is 10. **step** - 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. **product** - Returns the build number of the product. For example, if the product version is SecureZIP Server 12.10.0003, the value returned is 3. --- Default = **major**	The command line: ***pkzipc -version*** outputs two lines like the following after the usual header information: ***Program File Version (pkzipc): 12.30.1062*** Product Version: 12.30.0004 The **minor** sub-option outputs just the minor version number, for example, 10: ***pkzipc -version=minor***	standalone

Name/Description	*Value(s)*	*Example usage*	Used with
view Displays information about the files in an archive—for example, the compressed size of a file Configurable	**brief** - present information in the most compact manner. **detail** - present information in the most detailed manner **normal** - present information in the normal manner. **security** - Display encryption and signature information --- Default = **normal**	***pkzipc -view save.zip***	standalone
warning Pauses after every specified warning and prompts whether to continue. If no warning is specified, pauses after every warning. Configurable	**<warning number>** - One or more warning numbers, separated by commas. To override a warning number configured for the option (and thus *not* pause and prompt on that warning), precede the number with a hyphen --- No default value.	***pkzipc -extract -warning=43 save.zip **** ***pkzipc -extract -warning save.zip **** ***pkzipc -extract -warning=-43 save.zip ****	add, extract, test, view

Name/Description	*Value(s)*	*Example usage*	Used with
wipe Overwrites PKZIP temporary files and files deleted by PKZIP to prevent recovery of their data Configurable **Note:** This option is deprecated. Use the functionally identical **shred** option instead.	**none** - turns off shredding: files are not overwritten **random** - Overwrites files once with random data **dod5220** - Overwrites files three times, to the DOD 5220.22-M specification **nsa** - Overwrites files seven times, to the NSA standard --- Default = **none** Value if used on command line without a sub-option = **random**	***pkzipc -add -move -wipe=nsa myfiles.zip ****	Add
zipdate **Note:** This option is deprecated. Use the functionally identical option **archivedate** instead. Set the file modification date of the archive file. Configurable	**newest** - set to the date of the newest file within the archive file. **oldest** - set to the date of the oldest file in the archive file. **retain** - retain the original date of the archive file (the date when the file was created). **none** - disable the file date in the configuration file and set the archive date as the last modification date. --- Default = **none**	***pkzipc -add=update -zipdate=retain save.zip *.txt***	add, delete, fix, header, comment, sfx

Name/Description	*Value(s)*	*Example usage*	Used with
zoneidentifier Specifies that any Zone Identifier information for the archive should be copied to all files extracted from the archive. Configurable (Windows)	**enable** – Indicates the Zone Identifier should be copied from the archive. **disable** – Indicates the Zone Identifier should not be copied from the archive. ---------- Default = enable	***pkzipc -extract -zoneidentifier archiveFromInternet.zip***	extract

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.

Error	*Potential Cause(s)*
(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.

Error	*Potential Cause(s)*
(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 removable media on which you are creating these files.
(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. ***(e.g., pkzipc -add=freshen /temp/test.zip *.txt)***
(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.

Error	*Potential Cause(s)*
(E14) Can't create archive: XXX.	PKZIP is not able to create the archive file. Verify that the destination directory is not full, 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 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.
(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.

Error	*Potential Cause(s)*
(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.
(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 *extract* command in the command line.
(E76) Cannot open alternate config file: <file name>	The *altconfig* 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.

Error	*Potential Cause(s)*
(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 - *nnnnn*, contact your system administrator	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. 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.
(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

Error	*Potential Cause(s)*
(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
(E93) Timestamp failed	PKZIP unable to connect to Internet. Timestamp server URL is incorrect. Problem with signed file. Each file in the archive must be signed with the same hash algorithm. Be sure to sign each file in the archive. Problem with time stamp server. Try again later.
(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 3.
(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.
(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.
(E160) Unable to read sponsor list: XXX	Unable to find or read master file with information about PKWARE PartnerLink sponsors
(E161) Unable to read sponsor distribution package	Unable to find or read the distribution package for a PKWARE PartnerLink sponsor

Error	*Potential Cause(s)*
(E162) Unknown sponsor	A sponsor referenced on the command line Is not in the master list of PKWARE PartnerLink sponsors
(E163) Invalid sponsor: <sponsor>	The sponsor distribution package for this PKWARE PartnerLink sponsor failed internal integrity checks. There may be a problem with the signature on the package.
(E164) Archive not signed by an authorized sponsor	SecureZIP Partner cannot extract the archive because it is not signed by a valid PKWARE PartnerLink sponsor.
(E165) Sponsor required but not specified	The SecureZIP Partner command line directed to create an archive, but no PKWARE PartnerLink sponsor recipient was specified (with the **sponsor** option).
(E166) Unable to create ZIP archive for multiple sponsors	The **sponsor** option was specified more than once on the SecureZIP Partner command line. An archive cannot be created for multiple sponsors.
(E167) Unable to update ZIP archive using file name encryption	SecureZIP Partner cannot update an archive created with encrypted file names.
(E168) Sponsor not licensed for SecureZIP Partner: <sponsor>	SecureZIP Partner cannot create archives for this PKWARE PartnerLink sponsor. The sponsor is licensed only for extracting archives with Reader, not for using SecureZIP Partner.
(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.
(E204) The specified certificate does not meet the minimum requirements for signing when FIPS mode is enabled	In FIPS mode, RSA and DSA keys must be at least 2048 bits in size in order to sign with them.
(E205) The specified certificate does not meet the minimum requirements for encrypting when FIPS mode is enabled.	In FIPS mode, RSA keys must be at least 2048 bits in size in order to encrypt for them.
(E252) Only the super user may enter license keys!	You must run PKZIP as root, or be logged on as root, to use the **enterlicensekey** command.

Error	Potential Cause(s)
(E253) This program is not licensed for use on Windows Server platforms. Please contact PKWARE to obtain an appropriate server product for this machine.	You are using the PKZIP for Windows Desktop command line interface on Windows Server. Upgrade your license to PKZIP/SecureZIP 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:

PKZIP Warning	Potential Cause(s)
(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.
(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.

PKZIP Warning	*Potential Cause(s)*
(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.
(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.

PKZIP Warning	*Potential Cause(s)*
(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 ***ListCryptAlgorithms*** command to view a list of the encryption algorithms that PKZIP can use
(W47) No recipients specified. Recipients will not be used.	You specified the ***recipient*** 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 ***passphrase*** and did not also specify ***cryptalgorithm***, 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.

PKZIP Warning	*Potential Cause(s)*
(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.
(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 *cd* 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 (*passphrase*, *recipient* options), encryption algorithm (*cryptalgorithm* option).
(W63) You must specify -passphrase or -recipient to encrypt files!	You specified *-cryptalgorithm* or *-cd=encrypt* but did not specify either the *recipient* or *passphrase* option. Files are not encrypted unless one of these options is used.
(W68) Must specify MailTo, MailFrom and MailServer to email the archive.	You tried to email an archive without specifying all three options MailTo, MailFrom and MailServer. Values for all three must be specified on the command line or configured for use by default.

PKZIP Warning	*Potential Cause(s)*
(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
(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 - *nnnnn*, 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.

PKZIP Warning	*Potential Cause(s)*
(W85) Warning! Error shredding file: XXX	You used the *shred* 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 *fipsmode* 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 *fipsmode* 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 3. 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 almost every platform in use today, from mainframe to mobile. You can send a .ZIP file to anyone with confidence, no matter what type of computer they have.

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.

When we talk about information in this context, we're not comparing the amount of information contained in a news article compared to an entertainment show on television. Think about it more in terms of combinations of letters, numbers and punctuation.

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'.

*sh*e se**lls** sea *sh*e**lls** by the sea *sh*ore

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 (you can find a lengthy description and a link to Huffman's original paper at Wikipedia), 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.

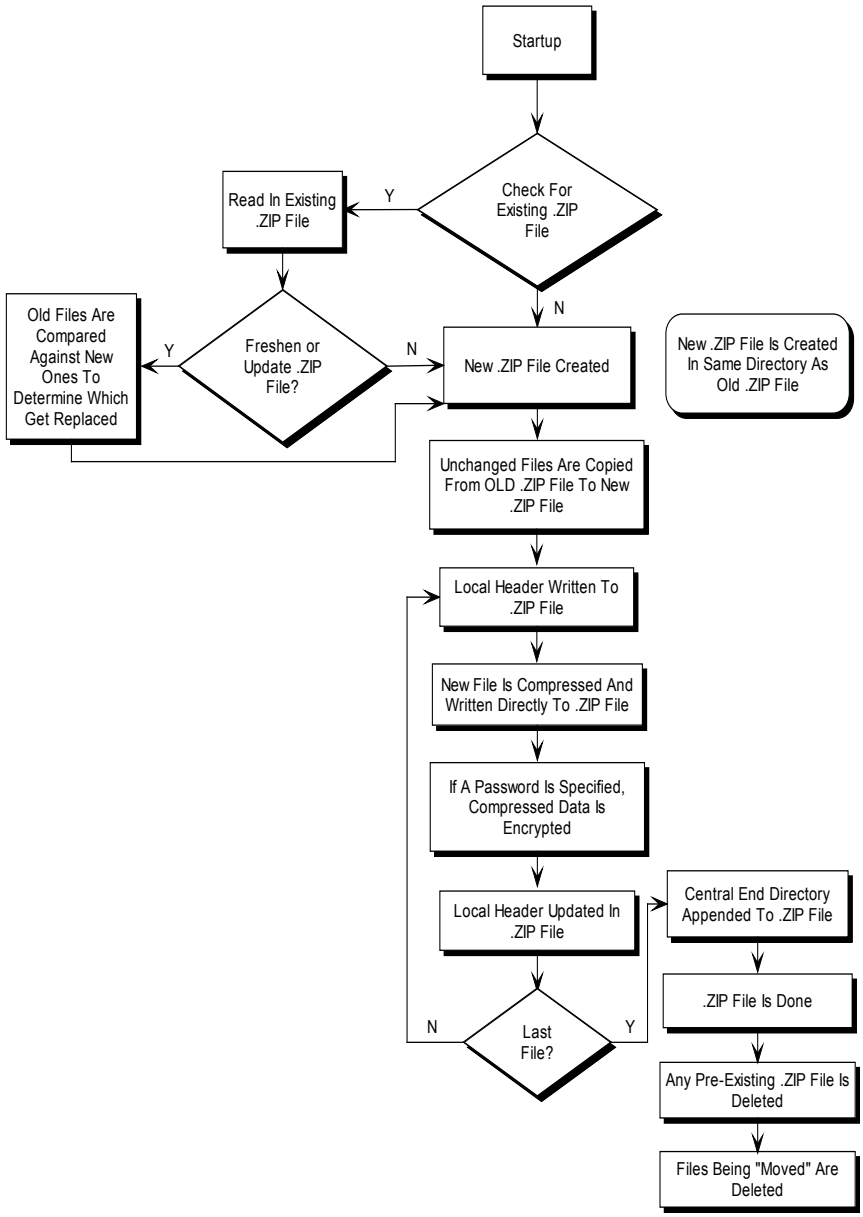
PKZIP is the second generation of these programs. PKZIP can not only compress and archive files, but also stores a great deal of vital information about the files. PKZIP even stores directory structures.

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.

E Tips for Scripting PKZIP on UNIX

There are a few general rules when scripting PKZIP on UNIX systems.

- Use the ***altconfig*** option in your script to explicitly set the location of the configuration file.
- Make sure that only the script can read the configuration file if it contains any sensitive information such as passphrases.
- Make sure that the configuration file specifies a temporary directory, even if PKZIP will never need to create temporary files.

  PKZIP uses configuration files to determine what defaults are set for options and sub-options. If a configuration file is not explicitly set in your script with the ***altconfig*** option, PKZIP looks for a configuration file in the current directory. A malicious user could put a configuration file in the directory where the script runs and thereby change the behavior of PKZIP. This could result in the wrong files (or even no files at all) being compressed or extracted. It could cause the extracted files to have different permissions after they are extracted, or it could even cause an SFX you create to ask the user to run some program after the SFX is run. This program could be a Trojan horse created by the malicious user and added by the configuration file.

  Just as it is dangerous not to explicitly specify a configuration file, it is also dangerous to let anyone change the file. Allowing a user to change the file creates the same risk as allowing a user to create one from scratch. Similarly, the directory containing the file should be protected so that the file cannot be removed and then replaced with a new one.

  To specify a configuration file for your script, use the ***altconfig*** option. This option can be used to create, update, or just read a custom configuration file (for example, one intended for use specifically with your script).

  The following command creates an alternate configuration file `newconfig.xml` in the current directory. The ***default*** option used with the ***config*** command initializes default settings to their original values.

  ```
  pkzipc -altconfig=newconfig.xml -config -default
  ```

F McAfee eBusiness Server Command Options

If you are transitioning from the McAfee eBusiness Server (EBS), you can use SecureZIP Server eBusiness Edition in OpenPGP Mode to run many of your existing EBS scripts with minimal editing. The commands include ***decrypt***, ***encrypt***, and ***sign***.

You can do this if you're using the legacy PGP.exe application as well, See "Using Legacy PGP Mode" later in this appendix.

Using OpenPGP Mode

To enable OpenPGP Mode:

1. Install SecureZIP
2. Copy or Link pkzipc.exe to the program name ebs.exe.

   To copy and rename pkzipc.exe to ebs.exe:

   copy pkzipc.exe <path/>ebs.exeTo use a symbolic link for pkzipc.exe:

   mklink <path/>ebs.exe <path/>pkzipc.exe
3. If you have the McAfee eBusiness Server in your PATH, either remove the PATH statement altogether, or replace the pointer to the McAfee ebs.exe program with the PKWARE program defined in step 2.
4. Make sure any running scripts have the PATH set to use the ebs.exe program from step 2.

Name/Description	*Shortcut*	*Value(s)*	*Example usage*	*Used with*
armor Create ASCII armored file	-a	No sub-options. --- No default value.	***ebs --encrypt --armor save.pgp***	encrypt, sign

Name/Description	Shortcut	Value(s)	Example usage	Used with
authenticate Verifies that an archive is signed.		No sub-options. --- No default value.	***ebs --decrypt --authenticate signed.pgp***	decrypt
Conventional Trigger use of symmetric passphrase encryption	-c	No sub-options. --- No default value.	***ebs --encrypt --conventional save.pgp***	encrypt
conventional-passphrase Provide symmetric encryption passphrase		<passphrase>	***ebs --encrypt --conventional --conventional-passphrase <passphrase>***	encrypt
decrypt Specify decryption operation	**-d**	No sub-options. --- If no other command is entered, ***ebs*** will default to ***decrypt***.	***ebs --decrypt [--passphrase <passphrase>] [--preserve-name] save.pgp***	standalone
dry-run Prints out messages to preview the results of a set of commands or options without actually performing the tasks	**-n**	No sub-options. --- No default value.	***ebs --encrypt --dry-run save.zip***	encrypt
encrypt Specify encryption operation	**-e**	No sub-options. --- No default value.	***ebs --encrypt --conventional [--conventional-passphrase <passphrase>] save.pgp *.doc***	standalone
help Displays help screen	**-h**	**<command or option>** - Any command or option for which help is desired. No default value.	***ebs --help*** Display help for the decrypt command: ***ebs --help --decrypt***	standalone
output Sets OpenPGP output file name.	**-o**	**<filename>**	***ebs --decrypt --output save.pgp save.zip*** ***ebs --encrypt --output save.zip encrypt.pgp***	decrypt, encrypt , sign

Name/Description	Shortcut	Value(s)	Example usage	Used with
overwrite Specifies whether to overwrite existing files with files being added or extracted. By default, PKZIP prompts before overwriting when extracting but not when adding.	-ow	No sub-options. --- No default value.	***ebs --decrypt --overwrite save.zip***	encrypt, decrypt
passphrase Specify private-key passphrase	**-z**	**<passphrase>** - The passphrase. --- No default value.	***ebs --encrypt --passphrase beowulf9 save.zip***	encrypt, decrypt
preserve-name Ignore any internal file name and use OPGP filename when decrypted		No sub-options. --- Default = off.	***ebs --decrypt -preserve-name sample.txt.pgp***	decrypt
sign Specify signing operation.	**-s**	No sub-options. --- No default value.	***ebs --encrypt --sign --sign-with "John Smith <johns@example.com>" save.zip***	encrypt, standalone
signed-by Specifies the sender's key. Decrypt this file only if the file is signed with this key. The option can appear more than once in the same command line, to specify multiple keys.		**<email address>** - Email address of the person associated with the OpenPGP key pair. **User name** - The name of the person associated with this OpenPGP key pair. **UserID** - This value can contain a name, email address and comment; such as: Tom <tom@example.com>. **@<file name>** - Specifies a text file which contains a list of certificates, one on each line.	***ebs --decrypt --signed-by "john.public@nowhere.com" save.zip*** ***ebs --decrypt --signed-by "John Public" save.zip*** ***ebs --decrypt --signed-by "John Public <john.public@nowhere.com>" save.zip*** ***ebs --decrypt --signed-by "john.public@nowhere.com" save.zip*** ***ebs --decrypt --signed-by "0x12345678" save.zip*** ***ebs --decrypt --signed-by @recipients.txt save.zip***	decrypt

Name/Description	Shortcut	Value(s)	Example usage	Used with
		keyID - Long or short version of unique key identifier. --- No default value.		
sign-with Specifies the key to use to sign an OpenPGP file.		**<email address>** - Email address of the person associated with the OpenPGP key pair. **User name** - The name of the person associated with this OpenPGP key pair. **UserID** - This value can contain a name, email address and comment; such as: Tom <tom@example.com>. **keyID** - Long or short version of unique key identifier.	***ebs --encrypt --sign-with "john.public@nowhere.com" save.zip *.doc*** ***ebs --encrypt --sign-with "John Smith" save.zip *.doc*** ***ebs --encrypt --sign-with "Jon Public <john.public@nowhere.com>" save.zip *.doc*** ***ebs --encrypt --sign-with "0x12345678" save.zip *.doc***	encrypt
text Translate line endings to UNIX	**-t**	Default = **UNIX**	***ebs --decrypt -text save.zip*** ***ebs --encrypt --text scripts.zip *.pl***	decrypt, encrypt
user Specifies the people (recipients) permitted to decrypt your OpenPGP-encrypted file. You can include this option more than once to specify multiple users.	**-u**	**<email address>** - Email address of the person associated with the OpenPGP key pair. **User name** - The name of the person associated with this OpenPGP key pair. **UserID** - This value can contain a name, email address and comment; such as: Tom	***ebs --encrypt --user "John Smith" save.zip *.doc*** ***ebs --encrypt --user "john.public@nowhere.com" save.zip *.doc*** ***ebs --encrypt --user "Jon Public <john.public@nowhere.com>" save.zip *.doc*** ***ebs --encrypt --user "john.public@nowhere.com" save.zip *.doc*** ***ebs --encrypt --user "0x12345678" save.zip *.doc***	encrypt

Name/Description	Shortcut	Value(s)	Example usage	Used with
		<tom@example.com>. **@<file name>** - Specifies a text file which contains a list of certificates, one on each line. **keyID** - Long or short version of unique key identifier. ---------- No default value	***ebs --encrypt --user @recipients.txt save.zip *.doc***	
version Gives information about the version of the release. Displays complete version information; also returns to the shell particular version numbers specified by sub-options.		No sub-options. ---------- No default value.	The command line: ***ebs --version*** outputs two lines like the following after the usual header information: ***Program File Version (pkzipc): 14.30.1181*** Product Version: 1.00.0047	standalone
wipe Overwrites PKZIP temporary files and files deleted by PKZIP to prevent recovery of their data	**-w**	No sub-options. ---------- No default value.	***ebs --encrypt --wipe myfiles.zip ****	decrypt, encrypt

Using Legacy PGP Mode

PKWARE offers support to users of the McAfee Legacy PGP application. This application supports the limited command set of PGP v2.63 described in the accompanying table. Other key differences between OpenPGP mode and Legacy PGP include:

- PGP mode commands only use the single-letter Command Switch, rather than the full command name.
- You can combine multiple commands with one switch. For example, to decrypt a PGP file and preserve the encrypted file's name, type:

  pgp -dp sample.txt.pgp

- Use ***+force*** to accept all requests from the program.

To enable Legacy PGP Mode:

1. Install SecureZIP

2. Copy or Link pkzipc.exe to the program name pgp.exe.

   To copy and rename pkzipc.exe to pgp.exe:

   copy pkzipc.exe <path/>pgp.exe

   To use a symbolic link for pkzipc.exe:

   mklink <path/>pgp.exe <path/>pkzipc.exe

3. If you have the McAfee eBusiness Server in your PATH, either remove the PATH statement altogether, or replace the pointer to the McAfee pgp.exe program with the PKWARE program defined in step 2.
4. Make sure any running scripts have the PATH set to use the pgp.exe program from step 2.

Name/Description	*Command Switch*	*Value(s)*	*Example usage*	*Used with*
armor Create ASCII armored file	-a	No sub-options. --- No default value.	***pgp –ea save.txt <userID> <userID>***	encrypt, sign
cypher Provide symmetric passphrase	-c	No sub-options. --- No default value.	***pgp –c save.txt [–z <passphrase>]***	encrypt
decrypt Specify decryption operation	**-d**	No sub-options. --- If no other command is entered, ***pgp*** will default to ***decrypt***.	***pgp –d save.txt.pgp [–z <passphrase>]***	standalone
encrypt Specify encryption operation	**-e**	No sub-options. --- No default value.	***pgp –e save.pgp <userID> <userID>***	standalone
+force Force YES to all responses		No sub-options. --- No default value.	***pgp –e +force save.pgp <userID> <userID>***	Encrypt, decrypt, sign
help Displays help screen	**-h**	No sub-options. No default value.	***pgp –h***	standalone

Name/Description	Command Switch	Value(s)	Example usage	Used with
outputfile Sets OpenPGP output file name.	**-o**	**<filename>**	***pgp –d save.txt.pgp –o new.txt*** ***pgp –e save.txt –o new.txt.pgp***	decrypt, encrypt , sign
passphrase Specify private-key or symmetric passphrase. If you specify the passphrase twice, the first item entered is assumed to be associated with the public key (for decryption) or the private key (for encryption). The second item entered is assumed to be the cypher passphrase for the file.	**-z**	**<passphrase>** - The passphrase. --- No default value.	***pgp -e save.txt -z beowulf9***	encrypt, decrypt
preserve-name Restores the original name of the encrypted file inside the archive. If this switch is not used, the decrypted file will use the archive filename minus ".pgp".	-p	No sub-options. --- Default = off.	***pgp -dp sample.txt.pgp***	decrypt
sign Specify signing operation.	**-s**	No sub-options. --- No default value.	***pgp –es save.txt –u <sign id> [<userid>]***	encrypt, standalone
text Considers all PGP plaintext files to be text files. Preserves the internal text structure and converts to local text conventions.	**-t**		***pgp -dt save.zip***	decrypt, encrypt

Name/Description	Command Switch	Value(s)	Example usage	Used with
user Specifies the person (recipient) permitted to decrypt your OpenPGP-encrypted file.	**-u**	**UserID** - This value can contain a name, email address and comment; such as: Tom <tom@example.com>. --- No default value	***pgp –es save.txt –u <sign id> [<userid>] *.doc***	encrypt
wipe Erase the original plaintext file after encryption. May also be used on its own for secure file deletion.	**-w**	No sub-options. --- No default value.	***pgp -ew myfiles.zip ****	decrypt, encrypt

Index

7

7Zip, 3, See also*: archives*

A

About This Manual, 1
anti-virus, 80
archive attribute, 31
archiveeach, 63
archives
  adding to existing, 255
  comments, 67
  convert to self-extracting, 75
  dates, 69
  deleting files, 254
  digital time stamping, 47
  extracting, 9
  fixing corrupt, 143
  freshen, 30, 80
  including open files, 52
  intermediate, 146
  moving files, 8, 72
  naming conventions, 5
  printing, 140
  self-extracting, 73
  sorting files, 70
  span and split, 61
  supported types, 2
  testing, 140
  update, 30
  verify signature, 84
  view contents, 8, 131
  writing to STDOUT, 48
archiving, 251, 252
ARJ, 3, See also*: archives*
attributes, 63
  extended, 66, 67
  setting internal, 145
authentication, 83, 102
avargs, 80
avscan, 80

B

backup, 31
  incremental, 31
binary data representation, 249
BinHex, 3, See also: *archives*
bzip2, 3, 59, 156
BZIP2. See also: *archives*

C

CAB, 3, See also: archives
CDR, 3, See also*: archives*
Certificate Authority (CA), 102, 103
certificate revocation list (CRL), 106
certificate stores, 39, 45
  UNIX, 109
certificates, 102, *See* digital certificates
changing defaults, 152
command line, 3
  changing command/option character, 161
  options, 14
  syntax, 4
commands, 13, 162
  abbreviating, 14
  changing character, 161
  default values, 29, 154
  difference from options, 13
  values, 14
Commands/Options
  204, 67, 162
  add, 29, 163
  after, 10, 163
  altconfig, 158, 164
  AltStream, 164
  archivedate, 69, 165, 229
  archiveeach, 63, 165
  archivetype, 60, 166
  armor, 257, 262
  ascii, 145, 166
  attributes, 63, 167
  authenticate, 167, 258
  avargs, 80, 167
  avscan, 80, 167
  before, 10, 168
  binary, 145, 168
  bzip2, 168
  bzip2, 59
  cd, 36, 169
  certificate, 43, 170
  comment, 67, 171
  configuration, 152, 172
  console, 94, 172
  conventional, 258, 262
  conventional-passphrase, 258
  crl, 106, 172
  cryptalgorithm, 33, 173
  cryptoptions, 124, 173
  dclimplode, 174
  decrypt, 258, 262

default, 174
deflate64, 175
delete, 175
directories, 54, 92, 175
embedded, 81, 176
encode, 145, 178
encrypt, 258, 262
enterlicensekey, 178
error, 142, 178
exclude, 12, 179
extract, 79, 179
fast, 56, 179
filetype, 64, 180
fipsmode, 37, 181
fix, 143, 181
freshen, 30, 80
ftp, 95, 181
groupadd, 182
groupdesc, 182
groupdetail, 182
groupfile, 182
grouplist, 182
groupremove, 183
hash, 44, 183
header, 68, 183
help, 20, 183, 258, 262
id, 92, 184
ignorewarnings, 142
ignorewarnings, 184
include, 12, 184
jobid, 147, 184
keyfile, 44, 185
keypassphrase, 44, 185
larger, 12, 186
ldap, 39, 186
level, 56, 187
license, 22, 187
links, 66, 187
listcertificates, 45, 188
listchar, 160, 188
listcryptalgorithms, 188
listcryptalgorithms, 33
listfile, 147, 189
listhashalgorithms, 189
listhashalgorithms, 45
listsfxtypes, 189
listsponsors, 27, 189
locale, 160, 189
log, 147, 190
logerror, 147, 190
logoptions, 147, 191
lowercase, 91, 191
lzma, 59, 192
mailbcc, 96, 192
mailbody, 96, 192
mailcc, 96, 193
mailfrom, 96, 193
mailoptions, 96, 193
mailreplyto, 96
mailserver, 96, 194
mailsubject, 96, 195
mailto, 96, 194, 195
mask, 69, 70, 196
maximum, 56, 197
messagedigest, 197
more, 198
move, 8, 72, 198
movearchive, 146, 198
mt, 198
namesfx, 75, 198
newer, 11, 199
noarchiveextension, 60, 86, 199
noextended, 66, 199
nofix, 199
normal, 56, 200
nosmartcard, 200
nozipextension, 199, 200
older, 11, 201
OpenFile, 52, 201
optionchar, 161, 202
output, 258, 263
overwrite, 130, 202, 259
owner, 92, 202
passphrase, 32, 203, 259, 263
path, 53, 203
permission, 70, 204
pgpPublicKey, 204
pgpSecretKey, 204
ppmd, 60, 205
preview, 143, 205
print, 140, 205
priority, 205
priority, 150
recipient, 206
recurse, 52, 207
rename, 50, 135, 207
runafter, 77, 208
RunContext, 208
security, 133
sftp, 209
sfx, 73, 210
sfxdestination, 75, 210
sfxdirectories, 76, 210
sfxlogfile, 76, 210
sfxoverwrite, 76, 211
sfxuitype, 76, 212
shortname, 138, 212
shred, 72, 213
sign, 45, 213, 259, 263
signed-by, 259
sign-with, 260
silent, 145, 214

smaller, 12, 215
snmptraphost, 215
snmptraphost, 148
sort, 70, 93, 216
span, 61, 217
speed, 56, 218
sponsor, 26, 218
store, 56, 218
stream, 50, 219
strict, 104, 219
substitution, 139, 220
temp, 144, 221
test, 140, 221
text, 260
timeout, 50, 222
times, 92, 222
translate, 138, 223
ts, 47, 224
update, 30, 80
usePGPName, 224
user, 260, 264
utf8, 62, 224
VerifyEncryption, 133, 225
VerifyRecipient, 134
verifysigner, 84, 225, 226
version, 20, 227, 261
view, 8, 131, 228
warning, 141, 228
wipe, 62, 229, 261, 264
zipdate, 69, 165, 229
zoneidentifier, 93, 230
comments, 67
compress. See also: *archives*
compressing, 29, 248
all files in a directory, 29
ASCII/BINARY internal attribute, 145
based on file type, 64
compression level, 55, 156
current directory, 6
digital certificate, 43, 45
directories, 54
encode, 145
file attribute information, 63
files in subdirectories, 52
filters, 10
freshen, 30
generate list file, 147
hash, 44
incremental archiving, 31
links, 66
list files, 57
listcertificates, 45
methods, 58, 59, 60, 156
only changed files, 30
only new files, 30
open files, 52
overview, 5
path information, 53
removable media, 61
removing file attributes, 70
removing file attributes, 69
selected files, 6
signature, 43, 45
single file, 6
update, 30
with attributes, 63
configuration file, 152
alternate, 158
contingency keys, 41
CRC, 254

D

dates, 69
environment variable, 160
dclimplode, 59, 156
defaults
changing, 155
changing with Options dialog, 157
resetting, 157
deflate64, 58, 156
deleting original files, 8, 72
shred, 72
digital certificates, 42, 43, 101, 102
hash, 44
listcertificates, 45
revocation lists, 106
root, 103
setting a default, 46
sign, 45
strict checking, 104
validity, 104
Windows, 107
digital signatures, 83, 101
attaching, 42, 104
authenticating, 83, 140
time stamping, 47
verifying, 84
DOS file names, 138

E

EBS, 257
armor, 257, 262
authenticate, 258
conventional, 258, 262
conventional-passphrase, 258
decrypt, 258, 262
encrypt, 258, 262
output, 258, 263
overwrite, 259
passphrase, 259, 263
sign, 259, 263

signed-by, 259
sign-with, 260
text, 260
user, 260, 264
eBusiness Edition, 19
email, 96–100
encode, 145
encrypting files, 31
file names, 36
passphrase, 32
recipient list, 33, 34
strong encryption, 15, 33
traditional ZIP encryption, 33
end-of-line characters, 138
entropy, 249
environment variables
date, 160
for certificate stores, 123
locale, 160
path (Windows), 24
PKSFXDATA, 74
time, 160
UNIX, 74
error messages, 231
treat warnings as, 142
extended attribute storage, 66
extracting
all files from an archive, 79
directory structure, 92
embedded files, 81
files only for display, 94
filters, 10
freshen, 80
from named pipe, 87
generate list file, 147
id, 92
list files, 94
lowercase, 91
new and existing files, 79
newer versions and new files, 80
only newer versions of files, 80
overriding default settings, 80
overview, 9
owner, 92
sorting, 93
times, 92
to named pipes, 89
to STDIN, 49
to STDOUT, 88
translate, 138
update, 10, 80

F

FastAES, 39, 124
Field Of Information theory, 249
file name encryption (FNE), 36
filetype, 50
filters, 10
FIPS mode, 37
format or wipe removable media, 62
freshen, 30, 80
ftp, 95

G

GnuPG. *See* OpenPGP
Groups
groupadd, 182
groupdesc, 182
groupdetail, 182
groupfile, 182
grouplist, 182
groupremove, 183
GZIP, 3, See also: archives

H

hash algorithms, 44
header comments, 68
help system, 20

I

IMG, 3, See also: *archives*
information content, 248
integrity test, 140
international characters, 62
Internet Explorer, 93
ISO, 3, See also: archives

J

JAR, 3, See also: *archives*

K

keyfile, 44
keypassphrase, 44
keys, 102

L

Legacy PGP Mode, 261
licenses, 22
entering keys, 21
sharing on Windows, 22
Lightweight Directory Access Protocol (LDAP), 39
links, 66
list files, 57, 94, 147
changing the list character, 160
listsponsors, 27
LZH, 3, See also: *archives*

lzma, 59, 156

M

mailBCC, 96
mailBody, 96
mailCC, 96
mailFrom, 96
mailOptions, 96
mailReplyTo, 96
mailServer, 96
mailSubject, 96
mailTo, 96
md5, 44
MIB file, 150
moving files, 8, 72

N

namesfx, 75
naming conventions, 5
noextended option, 66

O

OpenPGP, 3, 16, 42, 126, See also: *archives*
  archivetype, 60
  compared to X.509, 126
  keyrings, 127
  settings, 129
options, 13, 162
  abbreviating, 14
  changing character, 161
  combining, 14
  command line, 14
  default values, 29, 154
  difference from commands, 13
  values, 14
overwriting existing files, 130

P

Partner. See SecureZIP Partner
PartnerLink. *See* SecureZIP Partner
passphrase, 32, 246
password. *See* passphrase
PGP. *See* OpenPGP
PGP.exe, 261
PKCertTool, 109
  add, 112
  del, 117
  exit codes, 122
  export, 119
  keys, 118
  list, 114
  view, 121, 122
PKCS#11, 125
PKSFX, 73
PKSFXSDATA environment variable, 74
PKWARE, 21
PKZIP, 2, 248
  building a .ZIP archive, 252
  configuring, 152
  enterprise edition, 17
  for Windows Desktop, 157
  help, 20
  license information, 22
  license keys, 21
  Policy Manager, 41
  previewing command and option operations, 143
  scripting, 256
  sharing licenses, 22
  standard edition, 17
  support, 21
  v2.04 compatibility, 38, 67
  version information, 20
ppmd, 60, 156
Pretty Good Privacy (PGP). *See* OpenPGP
printing, 140
priority, 150
private key, 102, 103
  backup, 107
  keyfile, *44*
public key, 102, 103
  exporting, 107
Public-Key Infrastructure (PKI), 102

R

RAR, 3, See also: *archives*
Recipient Groups
  groupadd, 182
  groupdesc, 182
  groupdetail, 182
  groupfile, 182
  grouplist, 182
  groupremove, 183
recipient list, 33, 34
rename, 50, 135

S

scripting, 256
SecureLink. See SecureZIP Partner
SecureZIP Partner, 25, 26
  available commands, 28
  listsponsors, 27
  sponsor, 26, 218
selecting files, 10
self-extracting archives, 73
  command-line options, 77
  converting from ZIP, 75
  converting to, 75
  graphical interface, 76

logging messages, 76
options, 75
overwrite rules, 76
run program, 77
set-uid, 23
sfxdestination, 75
sfxdirectories, 76
sfxuitype, 76
sha256, 44
shred, 72
signing, 102
smart cards, 124
SNMP, 148
sorting files, 70, 93
spanning/splitting, 61
speed vs. size, 251
split sizes, 61
sponsor, 26
STDIN, 49
STDOUT, 48
store, 156
stream, 50
strict checking, 104
strong encryption, 15
subdirectories, 245, 246, 253
sub-options, 14
substitution, 139
suppressing screen output, 145
syntax, 4
options, 14

T

TAR, 3, See also: *archives*
Technical Support, 21
test, 140
text comments, 67
time
environment variable, 160
time stamping, 47
timeout, 50

U

UNIX, 256
running PKZIP as root, 23
wildcards, 23
update, 30, 80
extract, 10
utf8, 62
UUEncode, 3, 146, See also: *archives*

V

verifysigner, 84
version command, 20
viewing archive contents, 8, 131
virus scanning, 80

W

warning messages, 231, 238
ignore, 142
pause, 141
treat as error, 142
wildcards, 7, 23
Windows
file attributes, *31*
including open files in archives, 52
path, 24
Windows 2000, 124

X

X.509, 102
compared to OpenPGP, 126
XXEncode, 3, 146, See also: *archives*

Z

ZIP archives. *See* archives
zone identifier, 93