PKZIP®

Trusted Zip Solutions

Users Manual

- PKZIP Server
- SecureZIP™ Server

Table of Contents

1 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 the premium edition of PKZIP Server. SecureZIP Server has all the functionality of PKZIP Server and additional features as well. 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. Add-on modules providing additional functionality are available for both programs. Certain add-on modules are available only for SecureZIP Server, not for PKZIP Server.

Features Added to PKZIP by SecureZIP on Windows

On Windows, SecureZIP adds the following features to the set provided by PKZIP:

- Email and FTP integration: Options to create and transfer archives by email or FTP directly from the command line. See Chapter 5, on sending an archive.
- 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.”
- Strong encryption using a digital certificate instead of a password: This kind of encryption is both more convenient and more secure than password-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, certificate-based 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.

Features Added to PKZIP by SecureZIP on UNIX/Linux

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

- Email and FTP integration: Options to create and transfer archives by email or FTP directly from the command line. See Chapter 5, on sending an archive.

- 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.”

- Strong password-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).

- Strong encryption using a digital certificate instead of a password: This kind of encryption is both more convenient and more secure than password-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, certificate-based 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.

Add-On Modules for PKZIP Server and SecureZIP Server

Several optional add-on modules are available for PKZIP Server and/or SecureZIP Server. The modules are purchased and licensed separately from the base Server products.

Enhanced Data Processing Module: This module, available for PKZIP Server on both Windows and UNIX/Linux, adds these features (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, on sending an archive.

- 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.”

Directory Integration module: This module, offered for both Windows and UNIX/Linux versions of the product, requires SecureZIP Server.

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

About This Manual

This manual describes the command-line features of both PKZIP Server and SecureZIP Server for both Windows and UNIX/Linux. In general, references to PKZIP or PKZIP Server in the text should be taken to apply equally to SecureZIP Server. If a feature requires SecureZIP Server or an optional add-on module, 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. Chapter 2 provides an overview of program features and includes brief tutorials. See in particular the section “Understanding Commands and Options” for an explanation of how commands and options work.

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

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

Entering License Keys

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

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

1. (UNIX) If you want to share the license key among multiple users of the system, log on as `root`.

   Running the ***enterlicensekey*** command (see next step) creates a file `license.ini` (if it does not exist already) that must be accessible to all users of a license. Unless you log on as `root`, this file is created in your home directory; if you log on as `root`, the file is created in the PKZIP installation directory instead, where the `pkzipc` executable is placed. Multiple users can share a license only if the `license.ini` file is in the installation directory.

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.

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.

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>"

Sharing a License (UNIX)

To share a site license for PKZIP or an add-on module among multiple users of a UNIX system, put the `license.ini` file in the same directory as the PKZIP executable file `pkzipc` if it is not there already. The default location of this directory (the installation directory) is:

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

The `license.ini` file is created in the installation directory when you first run the ***enterlicensekey*** command (see "Entering License Keys," above) if you are logged on as `root`. Make sure that all users who are to share the license can read the file and do not have their own `license.ini` in their home directories.

Your Work Environment: The Command Line

In PKZIP Server, your work area is a character-based command line. 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***.

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*`

To Run the Program as Root

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

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 `autoexec.bat` file, which is typically located in the root directory (`C:\`). To add the PKZIP installation directory to your search path, follow the steps in the appropriate section below.

Windows 98/Me

1. Open a Windows Command Prompt window.

2. Type the following:

   edit c:\autoexec.bat

   The contents of your autoexec.bat file will appear in a text editor. Find the line in your autoexec.bat that starts with the PATH command followed by paths separated by semicolons. For example:

   PATH C:\DOS;C:\WINDOWS;C:\NETWORK;C:\NAV

3. At the end of this line, add a semicolon (if one is not there already) and the path, in quotes, to the folder where PKZIP Server is installed. (The quotes are necessary because the path contains a space.)

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

   ;"c:\program files\pkware\pkzipc"

4. Press Alt+F+X. You will be prompted to save the file before exiting. Enter Y to save.

5. Close any open applications and restart your PC

   You can now access PKZIP Server from any directory without specifying a path.

Windows NT/Windows 2000/Windows XP

1. Close any open Command Prompt windows.

2. Go to “My Computer” on your desktop and right-click the My Computer icon.

3. Select Settings | Control Panel from the Start Menu.

4. In the Control Panel, double click the System icon. The System (Properties) dialog appears.

5. If you are using NT, select the Environment tab. If you are using XP, click the Advanced tab and then click the Environmental Variables button.

6. Select the PATH variable in the System (Environment) Variables or User (Environment) Variables boxes. If you are unable to locate the PATH variable, enter the following in the Variable box:

   path

7. In the Value box, enter (in quotes) the path to the folder where PKZIP Server is installed. (The quotes are necessary because the path contains a space.)

   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.

8. Click the **Set** button.

9. 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 an Command Prompt Window to run PKZIP Server.

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

Strong Encryption

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

Traditional PKZIP encryption is password-based and uses the **password** option. Strong encryption can use either a password or a certificate-based list of recipients. To do password-based strong encryption, you use the **password** and **cryptalgorithm** options. To do certificate-based strong encryption, you use the **recipient** option. You must also have a digital certificate to use certificate-based strong encryption.

You need version 6.0 or later of PKZIP (or ZIP Reader) to decrypt archives that were strongly encrypted using PKZIP. You may need the SecureZIP edition of PKZIP to strongly encrypt archives yourself.

Using Digital Certificates on Windows

- PKZIP does not work directly with Netscape certificate stores. For PKZIP to access a certificate that you used Netscape to install, you must export the certificate from Netscape and then install it in the Windows certificate stores (usually by double-clicking on the certificate file in Windows Explorer).

- When you install a certificate on your system, the level of security configured can affect what you may see when compressing files with digital certificates. The level of security, either low, 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 password (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

Digital certificates are used to work with digital signatures as well as to do strong encryption for a list of recipients. To apply or authenticate digital signatures, or to encrypt or decrypt files for recipients, PKZIP needs to access keys in the certificates used.

Unlike Windows, UNIX and Linux do not have a 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.

To digitally sign files or to enable other people to strongly encrypt files specifically for you as a recipient, you need the SecureZIP edition of PKZIP 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 into them the certificates you want PKZIP to use.

PKCertTool sets up the following 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 beginning 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 the certificates’ owners. Certificates in this store contain only public keys; they do not contain private keys.
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,” below, if you have certificates in a PKZIP version 6 certificate store.

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

Format	*Description*
PEM	Contains a single certificate and/or private key. 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 contains only a single certificate with a private key. Common file extensions: `.pfx`, `.p12`
PKCS#7	Can contain one or more certificates and their public keys; does not contain private keys. Common file extensions: `.p7`, `.p7b`, `.p7c`

You must 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.

Single-User and Multi-User Environments

In a multi-user environment, an administrator with write access to shared directories must run PKCertTool to set up shared certificate stores.

- **In a multi-user environment**

  In a multi-user environment, an administrator should run PKCertTool to create a database containing the ROOT, AddressBook, and CA certificate stores as shared stores, accessible to all users.

  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.

- **In a single-user environment**

In a single-user environment, you can run PKCertTool to set up all four stores in the same database.

Once certificate stores are set up, PKCertTool needs to be run again only to add new certificates. 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.

In a multi-user environment, 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 searches in the following places, listed in order:

1. 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,” below.

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

3. `/usr/local/certificates/certificates.db`. NOTE: By default, PKZIP does not create, read, or write to a MY store in this shared location. The MY store contains the user’s private keys.

PKCertTool Commands and Options

PKCertTool is a separate program from PKZIP. It has commands to add, list, and delete certificates, as well as a ***keys*** command to verify that private keys exist for certificates.

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

Command / Option	*Description*
–add	Tells PKCertTool to add certificates to a store
–store <store name>	Specifies 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. Certificates in PKCS#7 files and PEM files that do not include private keys are added to the AddressBook store if no store is specified. NOTE: PKWARE recommends using a PKCS#7 file to add certificates to the AddressBook store.
–database <database>	Specifies 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
<certfile>	The location of a file containing one or more certificates. 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 or PKCS#7 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>	Specifies the path name of a PEM file that contains the private key for a certificate listed in a PKCS#12-format <certfile>. 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.

Command / Option	*Description*
–f <friendlyname>	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. Do not use –f to set a friendly name in the same command line with –all. A friendly name is useful to distinguish multiple certificates that have the same common name. For example, you might keep older, expired certificates to use to decrypt files that were encrypted using those certificates. In PKZIP, a friendly name that you assign with PKCertTool can be used with the PKZIP –recipient and –certificate options.
–passin <passphrase>	Specifies the passphrase used to decrypt a private key specified by <keyfile> or 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.
–passout <passphrase>	Specifies 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

NOTE: For readability, these examples use an en-dash hyphen. Substitute regular hyphens if you copy and paste these sample command lines for actual use.

The following command line adds all certificates 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

Command / Option	*Description*
–list	Lists information about the certificates in the specified store
–store <store name>	Specifies 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

Command / Option	Description
–database <database>	Specifies 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.”
<certfile>	Identifies the certificates about which to list information. The –list command can reference a certificate in a store by its common name, an email address contained in the certificate, or by a friendly name assigned to the certificate using the PKCertTool –add command.
–v	*Verbose*: Causes more information about the certificate to be listed.

Example

The following command line gives a verbose listing of certificates in the MY store, as shown in the accompanying output:

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 MO UJ 41 H3 24 U3 J3 8J 42 YH JJ 77 YO 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 YO YY 37 YU J9 O1 6J YY 9Y 1H 79 OJ 97 2Y
   1M 73 23 Y2
Public Key Hash:
   31 Y2 98 Y3 76 PU U2 J3 HH 25 U6 PY 9Y 6J 03 OU
   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 PKCertTool *del* Command

Command / Option	***Description***
–del	Deletes specified certificates from PKZIP store(s)
–store <store name>	Specifies the store from which to delete certificates. 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>	Specifies the location of the database containing the store from which to delete certificates. If this parameter is omitted, PKCertTool uses the first database found by searching the places listed above in the section “Locating Certificate Store Databases.”
<certfile>	Identifies the certificates to delete. The <certfile> argument can reference a certificate in a store by its common name, an email address contained in the certificate, or by a friendly name assigned to the certificate using the PKCertTool –add command. Alternatively, <certfile> can give 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 only deletes from the PKZIP stores copies of the certificates in a file; it does not delete files.

Example

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

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

The PKCertTool *keys* Command

Command	Description
–keys	Lists the public key hashes of all private keys in a certificate store database. 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>	Specifies 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 Y0 65 3H
   2U 8P 20 YY

--- PrivateKey     3 ---
   31 Y2 98 Y3 76 PU U2 J3 HH 25 U6 PY 9Y 6J 03 0U
   73 61 1H 8M
------------------
    3 private keys
```

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
   ```

Smart Card Compatibility (Win32)

On Windows, PKZIP can access certificates stored on smart cards. Files that are strongly encrypted using digital certificates and a recipient list can be decrypted using certificates on smart cards that 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.

Versions of PKZIP prior to 6.1 cannot decrypt files that are encrypted to be compatible with smart cards. To enable users of these earlier versions of PKZIP to decrypt files encrypted with a recipient list, PKZIP supplies a configurable option—**NoSmartCard**—for use when the **Recipient** option is set. Specifying the **NoSmartCard** option with the **Recipient** option turns off smart card compatibility and encrypts files so that they can be decrypted by previous versions of PKZIP, though not by smart cards.

The **NoSmartCard** option is needed only to do PKZIP 6.0-compatible certificate-based encryption (in other words, encryption using a recipient list). Password-based encryption works as in PKZIP 6.0. Smart cards are not used with PKZIP password-encrypted files.

In short, to enable users of earlier versions of PKZIP to decrypt your strongly encrypted files, either use the **NoSmartCard** option or use password-based encryption.

Creating the Tutorial Directory and Files

The PKZIP installation directory contains a batch file that creates a directory and several practice files for you to work with. These files enable you to do the tutorials in Chapter 2 without accidentally changing or deleting any of your other files.

See the appropriate section below—“Setting Up the Tutorial in Windows” or “Setting Up the Tutorial on UNIX”—for the platform you are using.

Setting Up the Tutorial in Windows

To create the tutorial directory and files in Windows:

1. From a Windows Command Prompt window, change to the directory where you installed PKZIP (`C:\program files\pkware\pkzipc` by default).

2. Run the `tutorial.bat` batch file: Type the following at the command prompt and press ENTER:

   tutorial.bat

   The batch file creates a new subdirectory called *tut* in the install directory and copies several practice files. It displays lines on screen like this:

```
.
.
.
Creating the Tut Directory
Copying the test files
Done.
```

3. Change to the tut directory created in the previous step. To do so, type the following and press ENTER:

   cd tut

4. To confirm that the files were created, type the following and press ENTER:

   dir

   You should see a list of files similar to the following:

```
07/10/2003  12:19 PM            9,108 black.tut
07/10/2003  12:19 PM            9,108 blue.fil
07/10/2003  12:19 PM            9,108 brown.doc
07/10/2003  12:19 PM            9,108 gold.tut
07/10/2003  12:19 PM            9,108 green.doc
07/10/2003  12:19 PM            9,108 orange.fil
07/10/2003  12:19 PM            9,108 pink.tut
07/10/2003  12:19 PM            9,108 purple.txt
07/10/2003  12:19 PM            9,108 red.txt
07/10/2003  12:19 PM            9,108 tan.txt
07/10/2003  12:19 PM            9,108 white.doc
07/10/2003  12:19 PM            9,108 yellow.doc
```

Setting Up the Tutorial on UNIX

To create a directory for the tutorials on UNIX:

1. Change to the directory where you installed PKZIP, such as /usr/local/bin/pkware/pkzipc.

2. Run the tutorial.sh file. Type the following and press **Enter**:

   sh tutorial.sh

   Running the tutorial.sh file creates a tutorial directory `tut` as a subdirectory of the install directory: for example, `/usr/local/bin/pkware/pkzipc/tut`. It copies several practice files for you to use with the tutorials.

3. Go to the directory that you created in the previous step: Type the following and press **Enter**:

   cd tut

4. To confirm that the files were copied, type the following and press **Enter**:

   ls -l

   A file list similar to the following appears:

```
-rw-r--r--   1 user    pkware     43326 Jul  1 02:51 black.tut
-rw-r--r--   1 user    pkware      4445 Jul  1 02:51 blue.fil
-rw-r--r--   1 user    pkware      5777 Jul  1 02:51 brown.doc
-rw-r--r--   1 user    pkware     43326 Jul  1 02:51 gold.tut
-rw-r--r--   1 user    pkware       582 Jul  1 02:51 green.doc
-rw-r--r--   1 user    pkware       582 Jul  1 02:51 orange.fil
-rw-r--r--   1 user    pkware     43326 Jul  1 02:51 pink.tut
-rw-r--r--   1 user    pkware       582 Jul  1 02:51 purple.txt
-rw-r--r--   1 user    pkware      4445 Jul  1 02:51 red.txt
-rw-r--r--   1 user    pkware      4445 Jul  1 02:51 tan.txt
-rw-r--r--   1 user    pkware      5777 Jul  1 02:51 white.doc
-rw-r--r--   1 user    pkware      5777 Jul  1 02:51 yellow.doc
```

Using Help

Besides the manual you are now reading, PKZIP provides online help for the PKZIP commands and options. The online help describes syntax and shows sample command lines.

You access the online help 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

To list the version of PKZIP that you are using, do the following:

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

pkzipc -version

Advanced users of PKZIP often require more detailed version information to assist in automating compression and extraction operations. PKZIP has the capability to return the 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 therefore not viewable. This functionality can be useful to verify PKZIP version numbers in the context of a .BAT file or shell script. The default action is to return the major number of the release. For example, for version 2.5 the return value will be 2. Using sub-options with the ***version*** option will return precise values of the release number to the shell. The available sub-options and the values they return are outlined in the following table:

Sub-Option	*PKZIP Returns*	*For example*
major	The major release number. For example, if the version number is 2.5, the value returned is <2>.	pkzipc -version=major
minor	The minor number of the release. For example, if the version number is 2.5, the value returned is <5>.	pkzipc -version=minor
step	The step or patch value. For example, if the version is 2.04.01, the step value returned is <1>.	pkzipc -version=step

Technical Support

For support, visit our Web site at:

www.pkware.com/support

2 The Basics

This chapter shows you how to use PKZIP to do basic compression and extraction operations. It also introduces some command line options and conventions. For example, you will learn how to specify the directory that you want to extract files to and how to specify just which files you want to compress or extract.

PKZIP gives you numerous commands and options that enable you to customize the operations that you use PKZIP to do. Some of these commands and options are discussed in this chapter. The rest are explained in later chapters. The present chapter gets you started. Its tutorials introduce you to major features of PKZIP and enable you to practice using PKZIP on files in a special tutorial directory. See the section, “Creating the Tutorial Directory and Files,” in Chapter 1 to learn how to set up your tutorial workspace.

An Overview of What PKZIP Does

PKZIP does four main sorts of things: It compresses files, and it decompresses them to restore them to their original state. It also authenticates and encrypts files. All the various commands and options relate to how you do these things or to other things that you can optionally do in the process. For example, you have options for picking the files that you want to compress or encrypt and options for how you want to compress or encrypt them.

Supported Archive Types

When PKZIP compresses files, it adds the compressed files to an *archive*. An archive is a kind of file that can contain other files. When PKZIP decompresses files, it extracts them from the archive that contains them. This is why the commands to compress and decompress files are named ***add*** and ***extract***.

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 the kind that you will probably use most often.

PKZIP enables you to create and extract from 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 that you can tell PKZIP what type of archive to create:

- Specify a name for the archive file that uses the file name extension customarily 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*
ARJ	Extract only	.arj
BinHex	Extract only	.hqx
BZIP2	Create and extract	.bz2
CAB	Extract only (Not supported on UNIX)	.cab
GZIP	Create and extract	.gz
LZH	Extract only	.lzh
RAR	Extract only (Not supported on UNIX)	.rar
TAR	Create and extract	.tar
UUEncoded	Create and extract	.uue
XXEncoded	Create and extract	.xxe
ZIP	Create and extract	.zip, .jar

Using the Tutorials

To help you learn PKZIP, this chapter contains several tutorials. These tutorials progress from simple to complex features. Each tutorial is preceded by a brief explanation of the topic(s) that the tutorial represents. For each tutorial, you will be asked to perform a specific PKZIP task. The results of what you type are reviewed every step of the way.

Conventions Used in the Tutorials

This manual was written for several operating system and platform versions of the command line implementation of PKZIP. To avoid confusion, we have tried to minimize references to specific operating systems or platforms. There may be instances in the tutorials where it is necessary to refer to an operating system or platform convention. Since PKZIP' s command line interface is virtually identical between the various operating system/platform versions, the examples used in this manual are, for the most part, universal. However, keep in mind that what you see on your screen might differ slightly from what you see in the examples.

Typically, in command line examples, we only reference what you would actually type at the command prompt, as in the following command line example:

```
pkzipc -add test.zip text.doc
```

Using Sample Files and Directories

The tutorials in this manual take you through compressing and extracting "specific" files from a "specific" tutorial directory. Feel free to use your own files if you choose to practice on them. However, keep in mind that the results that are displayed are specific to the files that these tutorials use as samples.

Note: These tutorials assume that you have installed PKZIP in the default installation directory and that you can access PKZIP from any directory without specifying a file path.

Preparing Your Workspace

To follow the tutorials in this manual step-by-step, you must create a working directory. Consider it a temporary directory to be used only for the tutorials. This helps to ensure that your permanent directories and files are not deleted or damaged while you are practicing with PKZIP. See the section, “Creating the Tutorial Directory and Files,” in Chapter 1 to learn how to set up your tutorial workspace.

Entering Commands: Syntax

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, act on in an archive (for example, to delete), or 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 include multiple file names and/or patterns, separate them with spaces.

The only elements that are always required in a command line are the name of the executable ***pkzipc*** and some 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

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 pkzipc –fast –add zipfile.zip pkzipc –add zipfile.zip –fast
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

Compressing and Adding Files to an Archive

To compress one or more files into an archive file, you use the ***add*** command.

For example, to compress a file called test.txt into an archive file called temp.zip, type the following:

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

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

In the tutorials that follow, you will learn about other commands and options that you can use and about other ways to specify files to operate on.

Archive File Naming Conventions

Conventionally, archive files (or *archives* for short) are given a name with a *file (name) extension* (the last part of the name, after the dot) that indicates the kind of archive they are. Thus a .ZIP archive generally has a name of the form *myarchive.zip*, where the file extension is *.zip*. Similarly, a BZIP2 archive generally has a file extension of *.bz2*.

Different archive types are used for different things. For example, a BZIP2 archive is used to contain a file compressed using the BZIP2 compression algorithm.

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 sometimes uses this information to determine what sort of archive you want to create. Here are the rules PKZIP follows with respect to archive types and names when you use the ***add*** command create an archive:

- If you specify an archive with an extension—for example, *myarchive.zip* or *myarchive.bz2*, or *myarchive.exe*, PKZIP will create an archive of that name. Also, by default, PKZIP will use 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).

- If you specify an archive 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” on page 73.

- If you specify an archive 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, using a trailing dot suppresses the .zip extension, but the dot remains a part of the file name. For example, specifying "*myarchive.*" as the archive name results in a file called "*myarchive.*".

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

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

Compressing a Single File in the Current Directory

The simplest PKZIP task is compressing a single file from the current working directory, and storing the resulting .ZIP file in that same directory. In this scenario, you are not retrieving files to compress from other directories, nor are you placing the .ZIP file in a different directory.

Tutorial A

Compress a single file (red.txt) into an archive file in your current working directory.

Follow these steps:

1. From the command prompt, change to the PKZIP Tutorial (tut) directory that you created (see "Creating the Tutorial Directory and Files" in Chapter 1). To compress the file red.txt into the .ZIP file called test.zip, type the following command and press ENTER:

pkzipc -add test.zip red.txt

Your file starts to compress, and your screen looks like the following:

```
Creating .ZIP: test.zip
  Adding File: red.txt      Deflating    (62.1%), done.
```

The first three lines contain the PKZIP banner information (name of the product, date, version and so on).

The last two lines list the name of the .ZIP file you created, the file(s) that were compressed, and the percentage that the file was compressed.

You have now successfully compressed a file into an archive file. Before you continue with the next section, let's take a look at your ***tut*** directory.

2. View the contents of the *tut* directory by typing one of the following commands:

➢ From a UNIX based command prompt, type:

ls -l

A file listing similar to the following will appear:

```
-rw-rw-rw-   1 user    pkware      1030 June 30 2:50 black.tut
-rw-rw-rw-   1 user    pkware      4185 June 30 2:50 blue.fil
-rw-rw-rw-   1 user    pkware      5920 June 30 2:50 brown.doc
```

```
-rw-rw-rw-   1 user    pkware      1030 June 30 2:50 gold.tut
-rw-rw-rw-   1 user    pkware       591 June 30 2:50 green.doc
-rw-rw-rw-   1 user    pkware       591 June 30 2:50 orange.fil
-rw-rw-rw-   1 user    pkware      1030 June 30 2:50 pink.tut
-rw-rw-rw-   1 user    pkware       591 June 30 2:50 purple.txt
-rw-rw-rw-   1 user    pkware      4185 June 30 2:50 red.txt
-rw-rw-rw-   1 user    pkware      4185 June 30 2:50 tan.txt
-rw-r--r--   1 user    pkware      1702 June 30 2:59 test.zip
-rw-rw-rw-   1 user    pkware      5920 June 30 2:50 white.doc
-rw-rw-rw-   1 user    pkware      5920 June 30 2:50 yellow.doc
```

➢ From a DOS based command prompt, type:

dir

A file listing similar to the following will appear:

```
.              <DIR>        06-01-01 12:00a .
..             <DIR>        06-01-01 12:00a ..
ORANGE   FIL           561  06-01-01  4:50a orange.fil
YELLOW   DOC         8,369  06-01-01  4:50a yellow.doc
RED      TXT         8,369  06-01-01  4:50a red.txt
GREEN    DOC           591  06-01-01  4:50a green.doc
PURPLE   TXT           591  06-01-01  4:50a purple.txt
WHITE    DOC         8,369  06-01-01  4:50a white.doc
BROWN    DOC         8,369  06-01-01  4:50a brown.doc
PINK     TUT        30,155  06-01-01  4:50a pink.tut
TEST     ZIP         1.702  06-01-01  4:50a test.zip
TAN      TXT         8,369  06-01-01  4:50a tan.txt
BLACK    TUT        30,155  06-01-01  4:50a black.tut
BLUE     FIL         8,369  06-01-01  4:50a blue.fil
GOLD     TUT        30,155  06-01-01  4:50a gold.tut
        13 file(s)         36,880 bytes
         2 dir(s)      87,945,216 bytes free
```

Note the test.zip file you have created. The file you compressed into the .ZIP file still appears in the current directory. That is because when you compress files, they remain in their original location, as well as inside the .ZIP file, compressed.

You can choose to remove or "move" the files into the .ZIP file so that they exist only in the .ZIP file after compression. The *Moving Files Into Your .ZIP File* section on page 31 shows you how to do this.

Compressing Selected Files in the Current Directory

With PKZIP, you can compress specific selected files into your .ZIP file. To do this, you simply type the files in your command, including a space between each one.

Tutorial B

Compress the following files into the test.zip file:

- green.doc
- blue.fil

Follow these steps:

➢ From the command prompt, type the following and press ENTER:

pkzipc -add test.zip green.doc blue.fil

Your file starts to compress, and output similar to following appears:

```
Updating .ZIP: test.zip
  Adding File: green.doc    Deflating    (39.3%), done.
  Adding File: blue.fil     Deflating    (62.1%), done.
```

This particular example added the specified files to the existing test.zip file. As you can see in the screen output above, PKZIP reports that it is "Updating .ZIP: test.zip" the .ZIP archive now contains any file(s) we added in the previous tutorial as well as the file(s) we added in this tutorial.

Compressing Files that Match a Pattern

PKZIP allows you to compress "only" files of a specific file pattern or extension. For example, you might want to compress only .doc files or .bmp files. On the other hand, you might want to compress all files that begin with the letter "p" or any other character match.

Using PKZIP conventions for typing file patterns (commonly called “wildcards”), follow these guidelines:

- For files of a specific extension, type an asterisk, a period, and the extension. You would type ***.txt** for ASCII text files (.txt extension).
- For files that begin with a specific character or characters, type the character(s), then an asterisk (*). You would type **res***for all files that begin with res.
- For files that end with a specific character pattern, type an asterisk (*), then the character(s). For example, for all files that end in "all" you would type ***all**.

Note: For other “wildcard” conventions and general command-line conventions, refer to your Operating System documentation.

Tutorial C

Compress all files in your tut directory that end with “.doc”

Follow these steps:

➢ From your command prompt, type the following and press ENTER:

pkzipc -add test.zip *.doc

The following appears on your screen:

```
Updating .ZIP: test.zip
Updating File: green.doc    Deflating    (39.3%), done.
  Adding File: yellow.doc   Deflating    (59.4%), done.
  Adding File: white.doc    Deflating    (59.4%), done.
  Adding File: brown.doc    Deflating    (59.4%), done.
```

When PKZIP is finished, the command prompt reappears.

Compressing All Files in the Current Directory

PKZIP allows you to compress all files in a specified directory. In the previous tutorials you compressed files by specifying either a specific file name (e.g., red.txt) or file pattern (e.g., *.doc). When you compress "all" files, you only have to type the name of the .ZIP file with the ***add*** command. PKZIP will assume that you wish to compress all files in the current directory. If, however, you wish to specify "all" files in another directory, you must use the convention for specifying "all" files (e.g., "*").

Tutorial D

Compress all the files in your tut directory into the test.zip file.

Follow these steps:

- From your command prompt, type the following and press ENTER:

  pkzipc -add test.zip

  Your files start to compress. Because you are compressing all of the files in your tut directory, this will take a little longer.

  Output similar to the following appears:

```
Updating .ZIP: test.zip
Updating File: red.txt      Deflating    (62.1%), done.
Updating File: green.doc    Deflating    (39.3%), done.
Updating File: blue.fil     Deflating    (62.1%), done.
Updating File: yellow.doc   Deflating    (59.4%), done.
Updating File: white.doc    Deflating    (59.4%), done.
Updating File: brown.doc    Deflating    (59.4%), done.
  Adding File: tan.txt      Deflating    (62.1%), done.
  Adding File: orange.fil   Deflating    (39.3%), done.
  Adding File: black.tut    Deflating    (61.0%), done.
  Adding File: purple.txt   Deflating    (39.3%), done.
  Adding File: pink.tut     Deflating    (61.0%), done.
  Adding File: gold.tut     Deflating    (61.0%), done.
```

  The phrase *Updating File* appears in front of some files, while *Adding File* appears in front of others. This is because in this tutorial you compressed files that were already compressed in the previous tutorials. The phrase *Updating File* appears with those particular files.

Compressing Files from a Different Location

When you compress files, you do not have to be in the directory where those files reside. You can compress them from anywhere on your computer. You just need to specify a path to any files that are not in the current directory so that PKZIP can find them.

Tutorial E

In this tutorial, you will switch locations to the directory that is directly above (parent to) the PKZIP Server install directory and compress files that appear in the tutorial directory.

Follow these steps:

1. Go to the directory that is directly above the PKZIP install directory (for example, c:\program files\pkware or /usr/local/pkware).

2. To compress the file called purple.txt from the PKZIP tutorial directory into a file called test123.zip file and place the test123.zip file into the current directory, type one of the following command lines and press ENTER:

➢ From a UNIX based command prompt, type:

pkzipc -add test123.zip pkzipc/tut/purple.txt

➢ From a DOS based command prompt, type:

pkzipc -add test123.zip pkzipc\tut\purple.txt

Your files start to compress, and the following appears:

```
Creating .ZIP: test123.zip
  Adding File: purple.txt   Deflating    (39.3%), done.
```

The test123.zip file is created in the current directory as that is the directory in which you typed the command. You can also specify a different location in which to create or update an archive file. The next section shows you how to do that.

Specifying a Different Location for the .ZIP File

Until now, you have created an archive file in the current directory. With PKZIP, you can specify a "different" location for the .ZIP file right in the command line. Simply include the location, or directory path, in your command.

Tutorial F

While remaining in the directory that is parent to the PKZIP install directory, compress the file called gold.tut, which is in PKZIP Tutorial directory, and place it in the test.zip file, which exists in PKZIP Tutorial directory as well.

Follow these steps:

➢ To compress gold.tut into the test.zip file, and place the updated .ZIP file in the PKZIP Tutorial directory, type one of the following command lines and press ENTER:

From a UNIX based command prompt, type:

pkzipc -add pkzipc/tut/test.zip pkzipc/tut/gold.tut

From a DOS based command prompt, type:

pkzipc -add pkzipc\tut\test.zip pkzipc\tut\gold.tut

Your files start to compress, and a screen similar to the following appears:

```
Updating .ZIP: pkzipc/tut/test.zip
Updating File: gold.tut     Deflating    (61.0%), done.
```

Moving Files into Your .ZIP File

Normally, the original files that you compress remain on your hard drive after you compress them. That is the default action when you use the ***add*** command. PKZIP allows you to *remove* the files from your hard drive location *after* they are compressed into the .ZIP file.

CAUTION: Be sure to keep backups of your important files. If you move your only copy of a file into a ZIP file, and the ZIP file becomes lost or damaged, you will have lost your file.

To remove files after they are compressed, use the ***move*** option along with the ***add*** command. To include an option in your PKZIP command, you simply type that option - preceded by a dash - after the ***add*** command. The following tutorial shows you how to do this.

Note: For basic information on PKZIP options, refer to the *Understanding PKZIP Commands and Options* section on page 39*.*

Tutorial G

In this tutorial, you are going to compress the file called pink.tut into the test.zip file, and "remove" pink.tut from your tut directory.

Follow these steps:

- ➢ Change back to the PKZIP Tutorial directory.

Type the following and press ENTER:

pkzipc -add -move test.zip pink.tut

Your file starts to compress, and your screen looks similar to the following:

```
Updating .ZIP: test.zip
Updating File: pink.tut     Deflating    (61.0%), done.

Moving 1 files...
Moved File: pink.tut     , done.
```

Note that PKZIP performed two tasks: Updating and Moving. After compressing the file, PKZIP removed it from the tut directory.

- ➢ To verify that the file you compressed was removed from your directory, type one of the following commands and press ENTER:

From a UNIX based command prompt, type:

ls -l

From a DOS based command prompt, type:

dir

Verify from the listing that the pink.tut file no longer exists as an individual file in the PKZIP Tutorial directory. The pink.tut file does however exist, compressed, in the test.zip file.

Viewing Files Within a .ZIP File

In the previous section, you learned how to move a file from the hard drive into a .ZIP archive. The file you moved still exists in your test.zip file. You can use the ***view*** command to verify that this file was indeed archived in the .ZIP file. The ***view*** command allows you to list the files that exist within a .ZIP file. Because you are not compressing files - only viewing them - you do not have to include the ***add*** command.

Tutorial H

View the files within the test.zip file.

Follow these steps:

- ➢ Verify that you are in the PKZIP tutorial directory.

To view the files within the test.zip file, type the following and press ENTER:

pkzipc -view test.zip

PKZIP displays a file listing of all of the files contained in the test.zip file. The listing will look similar to the following:

```
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
    561B DeflatN     340B 39.4% 06/01/2001  4:50a b6a63b7f -a-w-  green.doc
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a 87b3c388 -a-w-  blue.fil
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a 87b3c388 -a-w-  brown.doc
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a 87b3c388 -a-w-  white.doc
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a 87b3c388 -a-w-  yellow.doc
    29KB DeflatN    8130B 73.1% 06/01/2001  4:50a faf7aa7a -a-w-  black.tut
    29KB DeflatN    8130B 73.1% 06/01/2001  4:50a faf7aa7a -a-w-  gold.tut
    561B DeflatN     340B 39.4% 06/01/2001  4:50a b6a63b7f -a-w-  orange.fil
    29KB DeflatN    8130B 73.1% 06/01/2001  4:50a faf7aa7a -a-w-  pink.tut
    561B DeflatN     340B 39.4% 06/01/2001  4:50a b6a63b7f -a-w-  purple.txt
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a 87b3c388 -a-w-  tan.txt
  ------           ------ -----                                   ----
   139KB             42KB 69.2%                                     12
```

For more information on the ***view*** command, see the ***Viewing the Contents of a*** .ZIP File section on page 108.

Note: The above ***view*** list was generated from a DOS command line. In a UNIX ***view*** listing, the "Attr" column would be replaced by a "Mode" column with permission numbers for each respective file.

More Information on Compressing Files

The tutorials in the previous sections presented some of the file compression options available to you. There are other options as well. For example, you can choose to compress only files that are newer than the files already stored in the .ZIP file, or you can choose to compress only files that were created after a certain date.

For more information on options for compressing files, see Chapter 3.

Extracting Files from an Archive

To restore compressed files in an archive to their original form so that you can use them again, you *extract* them from the archive. You can extract all the files in an archive, or just selected files.

PKZIP gives you a variety of options for picking files and for choosing how to extract them. For example, you can tell PKZIP to extract only files that do not already exist in the folder where you want to extract them. You can also have PKZIP recreate a structure of folders and subfolders to contain the files if this path information was saved when the files were compressed.

To extract files, include the following elements in your command line:

- The command ***pkzipc*** to run PKZIP

- The PKZIP command ***extract*** preceded by a hyphen (**-**)
- The name of the archive file that contains the files to extract

For example:

```
pkzipc –extract temp.zip
```

The command line above extracts all the files contained in the archive file `temp.zip` into the current directory. To extract just a selection of files from an archive, 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
```

Numerous options are available to use with ***extract***. The tutorials that follow cover several extraction scenarios. For more information, see Chapter 4.

Extracting a Single File From a .ZIP File

The simplest task for extracting files is to extract a single file into the current directory. To make it easy, we are going to create an "extract" directory under your tut directory. We will call this directory ext. You will also copy the test.zip file into this directory.

Before you extract files, create a directory called ext and copy the test.zip file into that directory.

Follow these steps:

1. Verify that you are in the PKZIP Tutorial directory.

2. To create the ext directory, type the following and press ENTER:

```
mkdir ext
```

3. To copy the test.zip file into the ext directory, type the following and press ENTER:

From a UNIX based command prompt, type:

```
cp test.zip ext
```

From a DOS based command prompt, type:

```
copy test.zip ext
```

4. Now change to the ext directory. To do so, type the following and press ENTER:

```
cd ext
```

Note: The **cd** command in the preceding line stands for “change directory.” It changes the current directory to a directory that you specify on the command line. Many of the following tutorials use this command. The command is an operating system command, not a PKZIP command. It works basically the same in DOS, Windows, and UNIX. Consult your operating system documentation to learn more about the **cd** command.

Tutorial I

Extract the file called red.txt from the test.zip file.

Follow these steps:

1. Verify that you are in the ext directory.

2. Type the following and press ENTER:

pkzipc -extract test.zip red.txt

PKZIP extracts the file and displays a message similar to the following:

```
Extracting files from .ZIP: test.zip
    Inflating: red.txt
```

The first three lines contain the PKZIP banner information (name of the product, date, version and so on).

The fourth line contains the task PKZIP performed (in this case, *Extracting*) followed by the name of the .ZIP file (in this case, test.zip).

The fifth line contains the word *Inflating*, then the name of the file being extracted (in this case, red.txt). *Inflating* is an internal method of extraction that is meaningful only to the program.

3. To confirm that red.txt has been extracted into the ext directory, type one of the following commands and press ENTER:

From a UNIX based command prompt, type:

ls -l red.txt

A listing similar to the following appears:

```
-rw-rw-rw-   1 user    pkware      8369 June 30  2:50 red.txt
```

From a DOS based command prompt, type:

dir red.txt

A listing similar to the following appears:

```
RED      TXT         8,369  06-30-97  2:50a red.txt
         1 file(s)          8,369 bytes
         0 dir(s)      88,793,088 bytes free
```

Note that red.txt has been extracted into the ext directory.

Extracting Multiple Files From a .ZIP File

PKZIP allows you to extract multiple files from a .ZIP file at one time. This section shows you how to extract:

- Selected individual files.
- Files that match a specific file pattern. For example, files ending with a .txt extension.

- All files in the directory.

To prepare your ext directory to extract multiple files, first remove the red.txt file that you extracted in the previous tutorial.

Follow these steps:

1. Verify that you are in the ext directory.

2. Type one of the following commands and press ENTER:

From a UNIX based command prompt, type:

rm red.txt

From a DOS based command prompt, type:

del red.txt

The red.txt file is deleted. The only file that should be in the ext directory is test.zip.

Extracting Selected Files

With PKZIP, you can extract specific selected files from your .ZIP file. To do this, you simply type the files in your command, including a space between each one.

Tutorial J

Extract the following files from the test.zip file:

- green.doc
- blue.fil

Follow these steps:

1. Verify that you are in the ext directory.

2. Type the following and press ENTER:

pkzipc -extract test.zip green.doc blue.fil

A screen similar to the following appears:

```
Extracting files from .ZIP: test.zip
    Inflating: blue.fil
    Inflating: green.doc
```

3. To confirm that the files were extracted, type one of the following commands and press ENTER:

From a UNIX based command prompt, type:

ls -l

A listing similar to the following appears:

```
-rw-rw-rw-   1 user    pkware      1311 June 30 2:50 blue.fil
-rw-rw-rw-   1 user    pkware       131 June 30 2:50 green.doc
-rw-r--r--   1 user    pkware     24360 June 30 3:30 test.zip
```

From a DOS based command prompt, type:

dir

A listing similar to the following appears:

```
.              <DIR>        06-30-97  4:00a .
..             <DIR>        06-30-97  4:00a ..
TEST     ZIP        45,488  06-30-97  3:30a test.zip
GREEN    DOC           561  06-30-97  2:50a green.doc
BLUE     FIL         8,369  06-30-97  2:50a blue.fil
         3 file(s)         54,418 bytes
         2 dir(s)      88,788,992 bytes free
```

Extracting All Files From a .ZIP File

You can extract all of the files from your .ZIP file. When you extract specific files, as before, you must specify those files or file pattern. When you extract "all" files, you only have to type the name of the .ZIP file with the ***extract*** command. You can also, however, use the convention for specifying "all" files (*).

Tutorial K

Extract all the files from the test.zip file into the ext directory.

Follow these steps:

1. Verify that you are in the ext directory.

2. To extract all files from the test.zip file, type the following and press ENTER:

pkzipc -extract test.zip

Because you are extracting files that already exist in your ext directory, PKZIP asks you if you want to overwrite the file by displaying the following prompt:

Overwrite (<Y>es/<N>o/<A>ll/ne<V>er/<R>ename/<Esc>)?

3. Type **"y"** for yes if you wish to overwrite the file.

The same prompt will appear for the other files that already exist in the EXT directory. If you wish to overwrite those files as well type **"y"**.

PKZIP extracts the files, and the screen will look similar to the following:

```
Extracting files from .ZIP: test.zip
    Inflating: black.tut
PKZIP: (W7) Warning! file: blue.fil already exists.
Overwrite (<Y>es/<N>o/<A>ll/ne<V>er/<R>ename/<Esc>)? y
    Inflating: blue.fil
    Inflating: brown.doc
    Inflating: gold.tut
PKZIP: (W7) Warning! file: green.doc already exists.
Overwrite (<Y>es/<N>o/<A>ll/ne<V>er/<R>ename/<Esc>)? y
    Inflating: green.doc
    Inflating: orange.fil
    Inflating: pink.tut
    Inflating: purple.txt
    Inflating: red.txt
    Inflating: tan.txt
```

```
     Inflating: white.doc
     Inflating: yellow.doc
```

4. To confirm that the files were extracted into your ext directory, type one of the following commands and press ENTER:

➢ From a UNIX based command prompt, type:

ls -l

A listing similar to the following appears:

```
-rw-rw-rw-   1 user    pkware     10900 June 30 2:50 black.tut
-rw-rw-rw-   1 user    pkware      1311 June 30 2:50 blue.fil
-rw-rw-rw-   1 user    pkware      1588 June 30 2:50 brown.doc
-rw-rw-rw-   1 user    pkware       619 June 30 2:50 gold.tut
-rw-rw-rw-   1 user    pkware       131 June 30 2:50 green.doc
-rw-rw-rw-   1 user    pkware       337 June 30 2:50 orange.fil
-rw-rw-rw-   1 user    pkware     10900 June 30 2:50 pink.tut
-rw-rw-rw-   1 user    pkware      1998 June 30 2:50 purple.txt
-rw-rw-rw-   1 user    pkware       315 June 30 2:50 red.txt
-rw-rw-rw-   1 user    pkware      3075 June 30 2:50 tan.txt
-rw-r--r--   1 user    pkware     24360 June 30 3:30 test.zip
-rw-rw-rw-   1 user    pkware     17404 June 30 2:50 white.doc
-rw-rw-rw-   1 user    pkware      1637 June 30 2:50 yellow.doc
```

➢ From a DOS based command prompt, type:

dir

A listing similar to the following appears:

```
.               <DIR>        06-01-01 12:00a .
..              <DIR>        06-01-01 12:00a ..
ORANGE   FIL            561  06-01-01  4:50a orange.fil
YELLOW   DOC          8,369  06-01-01  4:50a yellow.doc
RED      TXT          8,369  06-01-01  4:50a red.txt
GREEN    DOC            591  06-01-01  4:50a green.doc
PURPLE   TXT            591  06-01-01  4:50a purple.txt
WHITE    DOC          8,369  06-01-01  4:50a white.doc
BROWN    DOC          8,369  06-01-01  4:50a brown.doc
PINK     TUT         30,155  06-01-01  4:50a pink.tut
TEST     ZIP         45,488  06-01-01  4:50a test.zip
TAN      TXT          8,369  06-01-01  4:50a tan.txt
BLACK    TUT         30,155  06-01-01  4:50a black.tut
BLUE     FIL          8,369  06-01-01  4:50a blue.fil
GOLD     TUT         30,155  06-01-01  4:50a gold.tut
        13 file(s)        187,850 bytes
         2 dir(s)      87,945,216 bytes free
```

You have now extracted all of the files contained in the test.zip file.

Overwriting Files that Already Exist in Your Directory

In the previous tutorial, you saw what appears when you extract files into a directory that contains file(s) with the same name. PKZIP offers several choices for handling the overwriting of files while extracting. Refer to Chapter 4 for complete information on extracting files from a .ZIP file.

Extracting Files to a Specified Directory

The extraction tutorials up to this point had you extract the contents of the test.zip file into the current directory. With PKZIP, you can specify a "different" extract location for the .ZIP file - right in your command. Simply include the location, or directory path.

Tutorial L

Extract the file called red.txt, archived in the test.zip file, into the directory above (parent to) the tutorial directory.

Follow these steps:

1. Verify that you are in the tut directory. If you are still in the ext directory, change to the tut directory.

2. To extract the red.txt file archived in the test.zip file into the tutorial directory's parent directory, type the following command line and press ENTER:

   pkzipc -extract test.zip red.txt ..

   The two periods (..) at the end of the command line tell PKZIP to extract the file into the directory up one level in the directory structure. PKZIP runs, and the following appears:

```
Extracting files from .ZIP: test.zip
    Inflating: ../red.txt
```

   As you can see from the "Inflating: ../" line above, PKZIP extracted the red.txt file to the parent directory of the current directory.

More Information on Extracting Files

The preceding tutorials show only a small sample of file extraction options. For example, you can extract only those files that are newer than the files in the extract directory. You can also specify how to handle the overwriting of files in the extract directory.

For information on these and other extract features, refer to Chapter 4.

Understanding Commands and Options

The previous sections demonstrated how to complete basic compression and extraction operations. In the tutorials, the ***add*** command was used to compress files while the ***extract*** command was used to extract files.

You can customize how an operation is done in PKZIP by using additional options on the command line, by explicitly specifying sub-options, and by configuring commands and options to use your specified sub-options by default.

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***, 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

In your PKZIP command, an option usually appears immediately following the main command (for example, ***add*** or ***extract***), separated by a space. The following is an example of using the ***maximum*** option with the ***add*** command, which instructs PKZIP to compress files at the "maximum level" but lowest speed:

pkzipc -add -maximum test.zip white.doc

An "option" is usually preceded by a command that represents the main task you are performing.

Tutorial M

Compress the white.doc file into the test.zip file using the highest level of compression (the ***maximum*** option).

Follow these steps:

1. Verify that you are in the tut directory. If you are in the ext directory, change to the tut directory.

2. To compress files using the ***maximum*** option, type the following and press ENTER:

pkzipc -add -maximum test.zip white.doc

PKZIP begins to compress the file using the ‘maximum’ or level 9 compression. Your screen will look similar to the following:

```
♦ Using compression level 9 - Maximum comp.

Updating .ZIP: test.zip
Updating File: white.doc    Deflating    (59.6%), done.
```

Abbreviating Commands and Options

When you include an option in your PKZIP command, you can abbreviate that option, as long as it remains unique. In the preceding example, you could have typed "max" instead of "maximum" as it is the only option that begins with the letters "max". The same example would look like the following:

pkzipc -add -max test.zip white.doc

Using Multiple Options

With PKZIP, you can use multiple options in the same command. For example, if you want both to use maximum compression and add a comment to a ZIP file, use both the ***maximum*** and ***comment*** options in your command:

pkzipc –add –max –comment test.zip brown.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.

Tutorial N

In this tutorial, you will compress the brown.doc file into the test.zip file using the ***fast*** option. Additionally, you will add a header comment to the test.zip file using the ***header*** option. The ***fast*** option instructs PKZIP to use the fastest compression speed.

Follow these steps:

1. Verify that you are in the tut directory.

2. To compress files using the ***fast*** and ***header*** options, type the following and press ENTER:

   pkzipc -add -fast -header test.zip brown.doc

   The ***fast*** option emphasizes faster compression speed over the compressed file size. The ***header*** option instructs PKZIP to prompt you to enter a header comment that will then appear in the header area of the .ZIP file.

   PKZIP will display the following prompt:

```
Zip Header ?
```

3. Specify a header comment (type ASCII text at the 'Zip Header' prompt) and press ENTER. The PKZIP output screen, including the " Zip Header ?" prompt, will look similar to the following:

```
♦ Using compression level 2 - Fast

Updating .ZIP: test.zip
Zip Header ? this is my header
  Adding File: brown.doc    Deflating    (58.8%), done.
```

As you can see from the screen output, PKZIP is using compression level 2. Also note that we supplied the comment "this is my header" at the "Zip Header ?" prompt.

Commands and Options That Have Values

Some commands and options have different possible values, called sub-options, that that let you customize how the command or option is done. For example, the ***level*** option enables you to specify how much compression you want to use (greater 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 include a value with a command or option, you 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, you use the command with the ***freshen*** sub-option:

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

With commands and options that have multiple possible predefined values or sub-options, one value is used as the *default* value. The default value is the value that PKZIP uses for the command or option unless you explicitly direct PKZIP to use some other value. For example, by default PKZIP uses the ***add*** command with the ***all*** sub-option to add all selected files to an archive. By default, PKZIP compresses the files using a compression level of 5.

Setting Default Values for Commands and Options

You can use the ***configuration*** command to replace original default sub-option settings for particular commands and options with your own. See Chapter 7.

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

Tutorial O

In this tutorial, you will compress files using a command and option, both of which contain sub-options or values. Specifically, you will compress only files that exist in the test.zip file, but that have changed. Additionally, you will specify that the file be minimally compressed while emphasizing maximum compression speed. This is accomplished by using the ***add=freshen*** command as well as the ***level=1*** option.

Follow these steps:

1. Verify that you are in the *tut* directory.

2. Open a text editor (e.g., vi; notepad; e). Type anything in the edit screen that you wish. Save the file into your tut directory as test.txt. Exit the text editor. A file called test.txt should now exist in your tut directory.

3. Add the test.txt file to the test.zip archive by typing the following:

   pkzipc -add test.zip test.txt

   The test.txt file is added to the test.zip archive.

4. Re-open test.txt file located in the tut directory with your text editor (e.g., vi; notepad; e). Edit or add characters in the edit screen. Re-save the file as test.txt into your tut directory. An updated test.txt file should now exist in your tut directory.

5. Type the following and press ENTER:

   pkzipc -add=freshen -level=1 test.zip

   PKZIP adds only those files that have been updated since test.zip was first created or last modified. Since we modified the test.txt file in Step 4, it will be updated in the test.zip archive. In addition, our command line tells PKZIP to compress the files using minimal compression while emphasizing maximum compression speed.

   The screen output will look similar to the following:

```
♦ Using compression level 1 - Maximum speed

Freshening .ZIP: test.zip
Updating File: test.txt     Storing      ( 0.0%), done.
```

3 Adding Files to an Archive

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

Conventions in This Chapter

Most commands and options discussed in this and subsequent 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.

Where a command has sub-options (also called a *value*), the heading introducing the section on a sub-option shows the command followed by an equal sign and then by the sub-option—for example, ***add=all***.

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 7 for information on how to change default settings.

Compressing New and Existing Files

When you compress files into a .ZIP file, you can create a new .ZIP file or add/update files in an existing .ZIP archive. For example, you might only want to compress files that end in ".doc". On the other hand, you might want to update files in an existing .ZIP file, but only those that have changed since the last time you compressed them.

Note: The ***add*** command is used for creating new .ZIP files as well as adding files to an existing .ZIP archive.

Compressing All Files in a Directory

add

You have the option of compressing 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 this example, all files in the current directory will be compressed into the test.zip file.

Note: The above command only compresses files that are located in that particular directory, not in the subdirectories that might appear below that directory. To learn how to compress files that appear in subdirectories, refer to the *Compressing Files in Subdirectories* section on page 47.

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 resultant 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 **7**.

Compressing New and Modified Files

add=update

PKZIP allows you to specify that only new or modified files are added to an existing .ZIP archive. When the ***update*** sub-option is used in conjunction with the command, the files specified for archiving will be compared against the files already present in the .ZIP file. If the file(s) to be added into the .ZIP file is already present and is *not* newer, PKZIP will *not* re-compress the file.

By using this option, you may save yourself time when archiving files that are backed up repeatedly. This option differs from the behavior of the ***freshen*** option in that files which are not already present in the .ZIP file will be added.

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.

Compressing Only Files That Have Changed

add=freshen

The ***freshen*** value allows you to selectively update files archived in a .ZIP file. PKZIP will compress only files that exist in the .ZIP file and that have changed. To update files that have changed, use the ***freshen*** value with the ***add*** option, as shown below:

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

You can also abbreviate the value, so you could type the following instead:

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

When you use ***freshen*** with ***add***, only files that already exist in the .ZIP file "and" that have also changed will be compressed. No new files will be added to the .ZIP file.

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.

Clearing Archive Attributes (WIN32)

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 those files, use the ***add*** command 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 the ***add*** command 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.

Incremental Archiving (WIN32)

add=archive

By using this option, you can create a complete backup of your disk, while clearing the archive attributes to make the 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 a clearing of the archive attribute for all files.

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

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

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

The archive option should only be used if you are preparing your disk for incremental backup (by doing a full backup) or if you are doing a full backup of your disk.

Archive Attribute Explained

Any given file may have several properties associated with it. One such property or attribute is called the Archive Attribute. When a file is created, this attribute is set to be ON. In addition, if a file is altered, the attribute is set. After a file has been backed-up by a program which uses this attribute, the attribute is switched off. By making use of the archive attribute, you can make certain that you get all files that are new or changed. You save time by not backing up files you have previously archived. This process is called an Incremental Backup.

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.

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 of the files when you extract them.

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/<filename>
```

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 the ***path*** option with the ***add*** command. Or use the ***path*** option together with the ***recurse*** option with the ***add*** command.

Additional Methods for Storing Directory Path Information

In addition to storing relative path information, PKZIP allows you to further customize path information storage in your .ZIP files. Several sub-options allow you specify exactly what directory and path information is to be stored.

Each sub-option is a value that you include with the path option. The path option and/or sub-option will override the path value in the PKZIP Configuration File. If no sub-option is specified, only relative path information is stored. Examples of each sub-option appear in the table below:

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	Store the directory path information that is specified in your PKZIP command.	pkzipc -add -path=specify docs.zip temp/docs/***** In this example, temp/docs is the directory information that will be stored.
relative	Store the directory path relative to the current working directory of the drive specified. **(WIN32)**	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 the ***add*** and ***extract*** commands.

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

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

pkzipc –add –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.
```

On Windows, even if the command line includes the ***directories*** option, you can turn off the searching of subdirectories for files matching a particular file pattern by including the 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 –dir 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 the directory path information that is specified in your PKZIP command.	pkzipc -add -directories=specify docs.zip temp/docs/* In this example, temp/docs is the directory information that will be stored.
relative	Store the directory path relative to the current working directory of the drive specified. (WIN32)	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.

Refer to the previous section, Additional Methods for Storing Directory Path Information, for more information.

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 identify a list file as such on the command line by prefixing it with the list character—"@" by default. See the ***listchar*** option if you want to use a different character for the list character.

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

pkzipc -add test.zip @lst.txt

You can also use a list File to specify files to exclude from an archive, based on some criteria, using the ***exclude*** option. The ***exclude*** option is discussed in more detail on page 55. For more information on the ***listchar*** option, see "**Changing the List Character for** List Files" on page 127.

Note: The file format for a list file when extracting files may differ from the format referenced above. See the section "**Extracting Files with a List** File" on page 99 for more information.

Getting a List of Files from Standard Input

You can tell PKZIP to treat as a list a set of files output by another program. PKZIP can then compress the files in the dynamically constructed list.

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 Based on Some Criteria

after, newer, before, older, larger, smaller, include, exclude

With PKZIP, you can compress files based on certain criteria. You can compress:

- Files that are newer than a specified date.
- Files that are older than a specified date or number of days.
- Files that are newer than a specified number of days.
- Files that are larger or smaller than a specified size.
- All files of a specific pattern.
- All files "except" those specified.

Refer to the sections that follow for more information.

Compressing Files Newer Than a Specified Date or Number of Days

after

With PKZIP, you can choose to compress only those files that are newer or equal to a specified date. To do this, use the ***after*** option, followed by an equal sign and the date, as shown below:

```
pkzipc -add -after=062495 test.zip
```

In this example, only files that contain a date newer or equal to June 24, 1995 will be compressed.

With PKZIP, you can enter dates in the following formats:

- mmddyy
- mmddyyyy

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

newer

With PKZIP, you can choose to compress only files that are newer than a specified number of days. To do this, use the ***newer*** option, followed by an equal sign and the number of days, as shown below:

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

In this example, only files that have been modified in the past 5 days will be compressed.

Compressing Files Older Than a Specified Date or Number of Days

before

With PKZIP, you can choose to compress only files that are older than a specified date. To do this, use the ***before*** option, followed by an equal sign and the date, as shown below:

```
pkzipc -add -before=062495 test.zip
```

In this example, only files that contain a date older than June 24, 1995 will be compressed.

With PKZIP, you can enter dates in the following formats:

- mmddyy
- mmddyyyy

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

older

With PKZIP, you can choose to compress only files that are older than a specified number of days. To do this, use the ***older*** option, followed by an equal sign and the number of days, as shown below:

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

In this example, only files that have been modified more than 5 days prior to the current date will be compressed.

Compressing Files Larger or Smaller than a Specified Size

larger

With PKZIP, you can choose to compress only files that are larger (in bytes) than a specified size. To do this, use the ***larger*** option, followed by an equal sign and the size, as shown below:

```
pkzipc -add -larger=5000 test.zip *
```

In this example, only files that are larger than 5000 bytes will be compressed.

smaller

With PKZIP, you can choose to compress only files that are smaller (in bytes) than a specified size. To do this, use the ***smaller*** option, followed by an equal sign and the size, as shown below:

```
pkzipc -add -smaller=5000 test.zip *
```

In this example, only files that are smaller than 5000 bytes will be compressed

Including Files That Match a Pattern

include

PKZIP allows you to compress files that match a specific pattern, for example, all files that end in the .doc extension.

To compress files that match a pattern, simply include the pattern after the ***add*** command, as in the following example:

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

You can use a configuration setting to include one or more file patterns by default. If, for example, you want to automatically include all files with the extension of .doc in PKZIP compress and extract operations, enter the following:

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

When you use the ***include*** option with the ***configuration*** command, PKZIP prompts you to configure the ***include*** default for add *and/or* extract operations. The .doc file pattern will henceforth be included by default in compress and/or extract operations. If, for example, you type the following command line:

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

Those files with .txt *and* .doc extensions are added to the test.zip archive.

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

For example, if you set up the configuration file to exclude **.txt* files by default but want to include text files in one particular case, use the ***include*** option followed by an equal sign and the file pattern to include. For example:

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

If you do not need to override a default configuration setting, it is not necessary to specify the ***include*** option in your command, only the file pattern.

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

Excluding Files from Being Compressed

exclude

When you compress files, you might want to exclude specific files from being compressed or extracted, for example, all files that end in .bmp. To exclude files, use the ***exclude*** option with the ***add*** command, as shown below:

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

In the example above, all files except those that end in the .bmp extension will be compressed.

Furthermore, you might want to exclude a list of files from being compressed or extracted, for example, all files that exist in the list file lst.txt. To exclude files listed in a list file, use the ***exclude*** option with the ***add*** command, as shown below:

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

In the example above, all files except those listed in the lst.txt file are compressed.

You have the option of specifying a default file pattern setting in your PKZIP Configuration Settings file. If, for example, you want to automatically exclude all files with the extension of .doc in PKZIP compress and extract operations, enter the following:

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

When you use the ***exclude*** option with the ***configuration*** command, PKZIP prompts you to configure the ***exclude*** default for add *and/or* extract operations. The .doc file pattern will henceforth be excluded by default in compress and/or extract operations.

Compressing Files in a Specific Format

shortname

The ***shortname*** option allows you to convert a file name in *long* file name format to a DOS equivalent *short* (8+3) file name before compressing the file(s). You may specify how you wish files to be compressed by using the ***shortname*** option with the ***dos*** sub-option:

```
pkzipc –add –short=dos save.zip
```

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 That Contain Certain Attributes (WIN32)

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 default attribute for purposes of compression is "read-only". That is, if you do not use the ***attributes*** option on your command line, "system", "archive" and "hidden" files are not compressed into your .ZIP files. If you do use the ***attributes*** option on your command line but do *not* specify a
sub-option, all files, including those that include "hidden", "archive" and "system" attributes, are compressed into your .ZIP files.

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 values in the configuration file.
- 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 dash (-) 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

PKZIP allows you to process files based on a specific file type. Use the ***filetype*** option with the following sub-options to process specific file types:

Sub-Option	To	For example
block	Include/exclude block special files. This would include files with a mode that begins with a "b". (brw-------).	pkzipc -add -filetype=block test.zip /dev/fd*
char	Include/exclude character special files. This would include 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. This would include filenames that have a dot (.) in the first position of the filename (.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

Sub-Option	To	For example
pipe	Include/exclude pipe files. This would include files with a mode that begins with a "p". (e.g., prwxr-xr-x)	pkzipc -add -filetype=pipe test.zip
regular	Include/exclude regular files. This is the default setting. If no filetype is specified, PKZIP will automatically include/exclude regular files.	pkzipc -add -filetype=regular test.zip
slink	Include/exclude symbolically linked files. This would include files with a mode that begins with a "l". (for example, lrwxr-xr-x)	pkzipc -add -filetype=slink test.zip
none	Exclude all file types except for those specified on the command line. The none option is typically followed by one or more file types. Only the specified types are included in the .ZIP file. For example, filetype=none,pipe results in only PIPE files being included.	pkzipc -config -filetype=none,slink
all	Include/exclude all file types.	pkzipc -add -filetype=all test.zip

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.

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 2 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, it automatically stores extended attributes with those files. PKZIP defines extended attributes as any file attributes other than the standard FAT file system attributes (Read-Only, Archive, System, Hidden, Directory). Extended attributes usually represent a characteristic of a file, such as the date and time the file was last modified.

If you do *not* wish to store any 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. However, be aware that 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
group ID	last modification time.
last modification time	last access time.
last access time	.
link information	

Note: Typically, PKZIP will automatically extract extended attributes with archived files and/or directories. Consequently, PKZIP will overwrite existing files, directories and extended attributes (EAs) with those files, directories, and extended attributes (EAs) stored in the .ZIP file. It should be noted, however, that extended attribute preservation is dependent on such things as the user's file system privileges as well as the option=suboption (e.g., ***id***, ***permission***, ***times***) specified on the command line or in the configuration file.

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 *
```

Removing File Attributes (WIN32)

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 *
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
```

Note: By default, PKZIP does not mask any file attributes when compressing files. All file attributes are retained in the archive. When extracting files, on the other hand, PKZIP masks *all* file attributes by default: file attributes stored in an archive are not restored by default on file extraction.

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
```

Including Additional Information in a ZIP File

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

You can include a:

- Text comment.
- Password to protect your .ZIP file.
- 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 only the new 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 *
```

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 is the same as 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 that the file was created.	pkzipc -add -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 *

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 password that you assign can decrypt and extract the files.

Depending on whether you have PKZIP or SecureZIP and the features you have licensed, 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 ***password*** and ***recipient*** options control encryption when you add files to an archive.

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

- 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.

The ***recipient*** option is used only to do strong encryption and is available only in SecureZIP. Depending on your platform, it may also require an add-on module. Both PKZIP and SecureZIP can decrypt files encrypted with either kind of strong encryption (password 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 Password

password

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

To specify a password, do one of the following:

- Specify the ***password*** option and enter a password of at least eight characters (preceded by an equal sign). For example (where the password is *mypassword*):

pkzipc –add –password=mypassword test.zip

- If you include just the ***password*** option and do not list a password, PKZIP prompts for the password. For example, the following command line produces a prompt:

pkzipc -add -password test.zip

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

```
Password?
```

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

```
Re-enter password for verification.
Password?
```

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

```
Passwords don't match!  Please try again.
Password?
```

Specify an Encryption Method

When you use strong encryption (available only with SecureZIP), you have a choice of encryption algorithms to use. The set of algorithms that are available depends on your version of SecureZIP and your system configuration. (On Windows, it may also vary with your version of Internet Explorer). 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 to use.

pkzipc –add –password –cryptalgorithm=aes,128 test.zip

By default, ***cryptalgorithm*** specifies AES,256. If you do not use ***cryptalgorithm***, SecureZIP applies traditional PKWARE encryption.

Note: Many other ZIP utilities can decrypt archives encrypted with traditional ZIP encryption, but most cannot decrypt strongly encrypted archives. To decrypt strongly encrypted archives requires PKZIP version 6.0 or later or a copy of ZIP Reader.

Extracting Password-Protected Files

To extract files from a password-protected archive, use the ***extract*** command with the ***password*** option.

- Type the password (preceded by an equal sign) as part of your command. For example:

pkzipc -extract -password=secret test.zip

If the correct password, the files are extracted (to the current directory, by default). If the password is incorrect, PKZIP displays a warning message:

```
PKZIP: (W20) Warning! Incorrect password for file: filename.ext
```

Re-type your command line with the correct password.

- If you specify the ***password*** option without a password, PKZIP prompts for a password. For example:

pkzipc -extract -password test.zip

When you press ENTER, a prompt appears:

```
Password?
```

Type the password. The characters appear on the screen as asterisks, for security. Press ENTER. If you specified the correct password, the files will be extracted to the current directory. If the password you entered is incorrect, a warning message displays:

```
PKZIP: (W20) Warning! Incorrect password for file: filename.ext
```

Retype your command line and when prompted enter the correct password.

- If you do not specify the ***password*** option when extracting an archive that contains password-protected files, PKZIP warns that the encrypted files are being skipped, and the files are not extracted.

Note: Passwords are case sensitive.

Note: For greater security, enter passwords at the prompt so that asterisks hide the characters you are entering. For information on using passwords in scripts, see Appendix E.

Encrypting Files with a Recipient List

recipient

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

Note: The ***recipient*** option is available only with SecureZIP and may also require an add-on module.

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 password was also specified).

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

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 70) to cause SecureZIP to search a specified LDAP directory.

Specifying Recipients

You can specify recipients 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 recipients by common name unless another sub-option is used.	-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
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 to specify an ordinary 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

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

  To use the ***recipient*** option to specify one of these types of file to define a recipient list comprising the owners of the certificates in the file, prefix the file name with a hash (#) character:

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

Specifying an Encryption Method with a Recipient List

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

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

Encrypting File Names

cd

The ***cd*** option uses strong encryption and is available only with SecureZIP. It may also require an add-on module.

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 password, if the archive was encrypted using a password.

Use the ***cd*** option 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 is 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 password encryption or a recipient list (or both), but you must use one of the strong encryption methods. You cannot encrypt file names using traditional, password 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 password. When you use the ***cd*** option with a password, SecureZIP uses the default strong encryption algorithm (ordinarily AES 256) if you do not explicitly specify an algorithm.

```
pkzipc –add –password=secret –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 (password/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 password or recipient list originally used to encrypt file names in the archive.

Accessing Recipients in an LDAP Directory

ldap

The ***ldap*** option enables you to access digital certificates in an LDAP directory.

To access certificate stores in directories requires the optional Directory Integration Module. This premium add-on is available only with SecureZIP and is licensed separately. SecureZIP accesses certificates in directory stores by making Lightweight Directory Access Protocol (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 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:password@]server[:port]/]ldap_base
```

where:

- userid (optional) is the user account with which to log in if the LDAP server requires a login
- password (optional) is the password 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.
- 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:secret@192.172.0.1:389/cn=users,dc=xyz,dc=com -recipient=”Mary Samplename” save.zip *.doc

pkzipc -add -ldap=jon_p:secret@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 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.

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 because it uses a larger dictionary than the Deflate algorithm that PKZIP uses by default.

Files compressed with the Deflate64 method can be extracted with most versions of PKZIP 2.5x and later but generally not by other .ZIP-compatible programs.

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 with the ***level*** option set for maximum compression:

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

Compressing Files Compatible with the Data Compression Library

dclimplode

The ***dclimplode*** option enables you to create .ZIP archives that are compatible with the PKWARE Data Compression Library (DCL). 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 to a Specified Type of Archive

archivetype

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
```

With the ***archivetype*** option, you can explicitly tell PKZIP what type of archive to create. The following example creates an archive *myfile.foo.bz2* of the BZIP2 archive type. Note that the file name extension *bz2* associated with the specified 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
```

However, when the ***archivetype*** option is used to specify an archive type, you can use the ***noarchiveextension*** option with it to tell PKZIP not to add an extension to the specified file name. For example, the following command creates an archive *myfile.foo* of the BZIP2 archive type:

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

Setting the Compression Level

Native ZIP compression (which uses the Deflate compression algorithm) and the ***bzip2*** and ***delate64*** 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 *.doc
```

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

```
pkzipc –config –level=9
```

For information on changing default settings, see Chapter 7.

Specifying a Compression Level by Name

store, speed, fast, normal, maximum

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

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

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

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

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 **7** for information on setting defaults.

* Types of files that the ***speed*** and ***fast*** options store uncompressed are:

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

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 the ***add***, ***extract***, ***test***, and ***view*** commands. 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
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.

Sub-Option	To sort by	For example
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.

Wiping Deleted Files

wipe

A deleted file still remains on your disk and can often be fully or partly recovered. If you want to prevent recovery of certain files that PKZIP deletes, you can use the ***wipe*** option with the ***add*** command to have PKZIP *wipe* them. Wiping a file overwrites the file's data so that it cannot be read.

If wiping is turned on, PKZIP wipes deleted files:

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

The wipe option has three sub-options:

Sub-Option	*Description*
None	Turns wiping off if it is configured on
Random	Overwrites files once with random data (the default)
NSA	Overwrites files seven times, to the NSA standard. (Takes much longer.)

For example:

```
pkzipc -add -wipe test *.doc
```

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 SFX files, for short.)

Note: You must have SecureZIP to create a PKSFX archive, or, with PKZIP, you must purchase the separate Enhanced Data Processing add-on module from PKWARE.

You can create two types of self-extractors:

- 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 (Windows versions 9x, NT [Intel], and XP). 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-extractor, 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 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
```

produces a display like this:

```
The SFX sub-option choices are:

    WIN32_X86_C610 - V6.10 Command Line SFX for Windows on X86
    WIN32_X86_G610 - V6.10 Windows SFX for Windows
```

In the list above, ***win32_x86_c610*** is the sub-option for the native Windows command line self-extractor, and ***win32_x86_gc610*** is the sub-option for the graphical Windows self-extractor.

To create a graphical Windows self-extractor, use the ***sfx*** option with the ***win32_x86_g610*** sub-option. For example:

```
pkzipc –add –sfx=win32_x86_g610 mysfx *.doc
```

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

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

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

If the ***SFXDestination*** option is used without a path, the self-extractor extracts to the temporary folder on the user’s system.

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.

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.

Sub-option	*Description*
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. (See “Run Programs with the Self-Extractor,” below.)
RegularSFX	Presents a dialog that enables the user to change the destination folder and other options before the archive is extracted

For example:

```
pkzipc –add –sfx=win32_x86_g –sfxuitype=regularsfx mysfx *.doc
```

Run Programs with the Self-Extractor

Use the ***runafter*** option with the ***sfx*** option to create a self extractor 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 is available only for the following types of self-extractors (the ellipses […] indicate omitted version numbers):

- 32-bit self-extractors

  win32_x86_g…

- Command line self-extractors under Win32 and UNIX

  win32_x86_c…
  aix4x_ppc_c…
  hpux_par_c…
  lnx2x_x86_c…
  sol2x_spc_c…

- X Windows System self-extractors

  aix4x_ppc_g…
  hpux_par_g…
  lnx2x_x86_g…
  sol2x_spc_g…

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
```

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	license	permission (UNIX only)
before	links (UNIX only)	print
console	locale	silent
directories	lowercase	smaller
exclude	mask	sort
extract	more	test
filetype (UNIX only)	newer	times
help	noextended	translate
Id (UNIX only)	older	version
include	overwrite	warning
larger	password	

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

```
test.exe –exclude="*.txt"
```

Compressing Files to Diskette

span

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

Creating a Spanned Archive (Windows)

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

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

To create a spanned archive:

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

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

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

Creating a Split Archive

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

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

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
```

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

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

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. Depending on your platform, you may also need to purchase an add-on module.

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

SecureZIP signing functionality is based on the X.509 certificate standard to be compatible with standard authenticity functionality in other applications such as Microsoft's Internet Explorer.

SecureZIP currently supports Level (or Class) One certificates (otherwise known as “email” or “personal” certificates). These certificates must be in 1024-bit (minimum) RSA format and must contain a private key.

Before you can use SecureZIP to sign files, you must have a digital certificate. Digital certificates are available from a variety of certificate authorities. Visit our web site for information on obtaining a certificate:

http://www.pkware.com

Options for Working with Signatures

certificate

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

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

Sub-Option	*To use*	*For example*
<Common name>	Specify, in quotes, the common name of the subject of the certificate (that is, the *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
#<filename>	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 password" 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

One way to specify a certificate to use for signing is to reference the file that contains it (see 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
```

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 password" save.zip
*.doc
```

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

hash

You can use the ***hash*** option with the ***certificate*** option to specify the hash method/algorithm to use for signing. You can choose between the SHA-1 and MD5 methods. Follow the option with an equal sign and either `SHA1` or `MD5`, as shown below:

```
pkzipc -add -certificate="My Cert" -hash=md5 test.zip *.*
```

In this example, the files compressed are signed using the certificate "My Cert". By default, both the local file and central directory are signed, using the MD5 hash method .

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 *.*
```

listcertificates

Use the listcertificates option to display a list of certificates available for signing files on your system. Specify the option with no sub-options, as shown below:

```
pkzipc -listcertificates
```

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 the extracting files.

Conventions in This Chapter

In some of the headings in this chapter, a word appears immediately below the heading. That word is the command or option you would type in your PKZIP command line. In some cases, the command or option appears with an *equals* sign (=) used to specify a value or sub-option—for example, ***extract=all***.

Default Values for Commands and Options

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

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

Extracting New and Existing Files

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

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

- All files in an archive (the ***all*** sub-option)

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

Extracting All Files from an Archive

extract=all

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

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

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

The ***all*** sub-option is the original default for the ***extract*** command. 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 just 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

The ***update*** sub-option 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

The ***freshen*** sub-option 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
```

Extracting Files Based on Some Criteria

after, newer, before, older, larger, smaller, include, exclude

With PKZIP, you can extract files based on certain criteria. You can extract:

- Files that are newer than a specified date or number of days.
- Files that are older than a specified date or number of days.
- All files that are larger than a specified size.
- All files that are smaller than a specified size.
- All files of a specific pattern.
- All files "except" those specified.

Refer to the sections that follow for more information.

Extracting Files Newer Than a Specified Date

after

With PKZIP, you can choose to add or extract only those files that are newer or equal to a specified date. To do this, use the ***after*** option, followed by an equal sign and the date, as shown below:

```
pkzipc -extract -after=062495 test.zip
```

In this example, only files that contain a date newer than or equal to June 24, 1995 will be extracted.

With PKZIP, you can enter dates in the following formats:

- mmddyy
- mmddyyyy

This order in which you type the month, day, and year depends on your ***locale*** setting. For more information on the ***locale*** setting, see Chapter 8.

newer

With PKZIP, you can choose to extract only files that are newer than a specified number of days. To do this, use the ***newer*** option, followed by an equal sign and the number of days, as shown below:

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

In this example, only files that have been modified in the past 5 days will be extracted.

Extracting Files Older Than a Specified Date

before

With PKZIP, you can choose to add or extract only those files that are older than a specified date. To do this, use the ***before*** option, followed by an equal sign and the date, as shown below:

```
pkzipc -extract -before=062495 test.zip
```

In this example, only files that contain a date older than June 24, 1995 will be extracted.

With PKZIP, you can enter dates in the following formats:

- mmddyy
- mmddyyyy

This order in which you type the month, day, and year depends on your ***locale*** setting. For more information on the ***locale*** setting, see Chapter 8.

older

With PKZIP, you can choose to extract only files that are older than a specified number of days. To do this, use the ***older*** option, followed by an equal sign and the number of days, as shown below:

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

In this example, only files that have been modified more than 5 days prior to the current date will be extracted.

Extracting Files Larger or Smaller than a Specified Size

larger

With PKZIP, you can choose to extract only files that are larger (in bytes) than a specified size. To do this, use the ***larger*** option, followed by an equal sign and the size, as shown below:

```
pkzipc -extract -larger=5000 test.zip *
```

In this example, only files that are larger than 5000 bytes will be extracted.

smaller

With PKZIP, you can choose to extract only files that are smaller (in bytes) than a specified size. To do this, use the ***smaller*** option, followed by an equal sign and the size, as shown below:

```
pkzipc -extract -smaller=5000 test.zip *
```

In this example, only files that are smaller than 5000 bytes will be extracted.

Including Files That Match a Pattern

include

PKZIP allows you to add or extract files that match a specific pattern, for example, all files that end in the .doc extension.

To extract files that match a pattern, simply include the pattern after the ***extract*** command, as in the following example:

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

You have the option of specifying a default file pattern setting in your PKZIP configuration file. If, for example, you want to automatically include all files with the extension of .doc in PKZIP compress and extract operations, enter the following:

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

When you use the ***include*** option with the ***configuration*** command, PKZIP prompts you to configure the ***include*** default for add *and/or* extract operations. The .doc file pattern will henceforth be included by default in compress and/or extract operations.

If you set up a default pattern in your configuration file, but want to specify a different pattern for this instance only (for example, extract all files that end in .txt), use the ***include*** option, followed by an equal sign, then the file pattern, as in the example below:

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

If you do not set a default for ***include*** in the configuration file, you do not have to specify the ***include*** option in your command, only the file pattern. However, if you specify both an ***include*** value as well as a file pattern, both are used in the file extraction process.

Excluding Files from Being Extracted

exclude

When you extract files, you might want to exclude specific files from being extracted, for example, all files that end in .bmp. To exclude files, use the ***exclude*** option with the ***extract*** command, as shown below:

```
pkzipc -extract -exclude="*.bmp" test.zip
```

In the example above, all files except those that end in the .bmp extension will be extracted.

You have the option of specifying a default file pattern setting in your PKZIP configuration file. If, for example, you want to automatically exclude all files with the extension of .doc in PKZIP compress and extract operations, enter the following:

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

When you use the ***exclude*** option with the ***configuration*** command, PKZIP prompts you to configure the exclude default for add *and/or* extract operations. The .doc file pattern will henceforth be excluded by default in compress and/or extract operations.

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 the inner archive file 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 embedded archive file of the specified type. You must specify the type.

For example:

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

In the example, if *outerarchive.zip* contains *innerarchive.zip* and no other archive files, PKZIP extracts the files from *innerarchive.zip* instead of extracting *innerarchive.zip* itself and does not prompt.

You can also use ***embedded*** to tell PKZIP *never* to extract the contents of an embedded archive of a specified type. This usage of embedded, like the first, causes PKZIP not to prompt for instructions. PKZIP simply extracts the embedded archive file, not its contents. To tell PKZIP *never* to extract the contents of a specified type of embedded archive, prefix the sub-option with a hyphen:

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

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.

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 Files in Lower Case

lowercase

The ***lowercase*** option allows you to extract files in lower case regardless of how the file name was originally archived. To force the file names to be extracted in lowercase, use the following example:

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

Preserving File Times

times

The *times* option allows you to preserve the access, creation and modification times of the extracted files. Specify the sub option *all* to preserve all times, use *access* to preserve the access times only, use *modify* to restore the time of last modification times or *create* to restore the creation times.

To preserve all the file times, use the following example:

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

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

Translating End of Line Sequence

translate

The *translate* option allows you to translate the carriage return/newline sequence end-of-line characters to another platform sequence. Specify the sub-option `dos` to translate each text line to the DOS standard (return/newline), specify `mac` to translate it to the MacOS standard (single carriage return), or *UNIX* to translate it to the UNIX standard (single newline)

To translate the text lines of an archive to UNIX, use the following example:

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

Retaining Directory Structure while Extracting

directories

If you stored directory path information within a .ZIP file, you can re-create those directory paths when you extract the files. For example, if you compressed a file called apples.doc in the temp/fruit directory, and you stored temp/fruit you can re-create temp/fruit in the location in which you extract the files.

To re-create directories, use the ***directories*** option with the ***extract*** command, as in the following example:

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

When you use this command, all directories that were stored in the .ZIP file will be retained during extraction. The directory path stored is appended to the directory in which you extract the files. For example, if your extract directory is /doc, and a directory path stored with the files is temp/fruit, the files would now be extracted to /doc/temp/fruit.

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 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 NT based systems.

Extracting Files with a List File

The file format of a list file used to extract or exclude certain files is somewhat different than the format used to include files. When compressing files, the list file needs to be in a format that the operating system can understand. For example, in DOS based command lines, it may be necessary to specify a drive letter when compressing files. By contrast, when extracting or excluding files, a drive letter is not necessary and therefore cannot be used in list files.

A list file may contain wild card specifications (*,?) as well as exact file names and paths. An example of a list file used in extract operations follows:

```
*.exe
*.doc
temp/readme.doc
temp/?????.*
text/news.txt
```

Using the @ character in your command line, you can point to the list file. Assume that the list file is called lst.txt and is located in your current directory. An example of such a command line follows:

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

In the example above, PKZIP extracts files from test.zip using file information it retrieves from a file called lst.txt, located in the current directory. It is important that the file format contained in the list file matches the format of the files in the test.zip file. For example, if test.zip contains no path information, specifying **text/news.txt** in your list file will *not* extract the file **news.txt** from the test.zip file. See the section on page 51 for more information on list files. For information on viewing the files as they appear in the .ZIP file, refer to the section “Viewing the Contents of a .ZIP File” on page 108.

Digital Signatures

PKZIP allows you to authenticate .ZIP files that have been signed with a standard X.509 Digital Certificate. Digitally signing a file allows you to test a file to determine if the archive is from a reliable source. This will ensure that the file is from a reliable source and can help prevent virus programs from damaging your computer. It is advised that if a file contains digital signatures, you should test the file before extracting files. The testing determines whether the file has been modified since the original person created it. If the archive was modified, its authenticity has been compromised and you may not want to extract the files to your computer. The authenticity of an archive can also be determined when files are extracted.

Signatures and certificates can be determined to be valid or invalid for each file and/or the central directory. The following table illustrates the warning messages that can be displayed when testing or extracting files from an archive.

Message	*Explanation*	*What to do?*
Signature is invalid	Indicates the archive has been changed after it was signed (usually by someone who is not using a digital signature). The archive may be corrupt.	You may want to try to obtain the file again (i.e., 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 (not by the creator of the file).
Certificate is not trusted	Indicates the certificate is currently not to be trusted.	This message indicates that the certificate is not to be trusted but the archive may not necessarily be compromised. Contact the issuer of the certificate to validate the certificate/signature.
Certificate is expired	Indicates the certificate has expired (i.e., 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 the file/archive may not necessarily be compromised.
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 the file/archive may not necessarily be compromised.
Certificate not found: XXX	Indicates that the certificate (with the name listed) could not be found on your system.	Check to see if the certificate name was misspelled. Be certain that the certificate exists on the system (for example check output of pkzipc or use PKZIP Explorer).

5 Sending an Archive

This chapter describes the command line options to transfer a new or existing archive to other people by FTP or email. This functionality is included with SecureZIP and may be added to PKZIP by purchasing the add-on Enhanced Data Processing Module.

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, password, 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`*`[:`*`password`*`[:`*`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`*`[:`*`password`*`[:`*`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`.
- password (optional) is the password associated with the user account. If no password is given, PKZIP tries an empty password. A colon (:) is not allowed in the password as this character is used to separate username, password, 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.

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.

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. (You must include ***mailTo*** on the command line even if it 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 password. This tells PKZIP to try plain-text or login authentication to connect to the server. Prefix the password with a colon (:), and use an *at* sign (@) to separate user/password information from the server address like this: `user:pass@server`. For example:

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

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

Note the colon prefixing the password.

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
```

If you have configured the ***recipients*** option to specify a file that lists the names of certificate holders, you can use the ***mailTo recipients*** sub-option to point PKZIP to this file for email addresses. PKZIP searches each certificate for an embedded email address to use and displays a warning for any certificate where no address is found.

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

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

The ***recipients*** option is available only in SecureZIP and may also require an add-on module.

Sending Encrypted Attachments

In general, you use the ***password*** or ***recipient*** options to encrypt files in an archive (see “Encrypting Files That You Add to an Archive” in Ch. 3). The ***mailTo*** option simplifies applying certificate-based encryption to files in an archive to be attached to an email message. To use certificate-based encryption on files that you add to an archive to be attached to an email message, you can simply use the ***cryptalgorithm*** option on the command line with ***mailTo*** to specify the encryption algorithm. You do not need to explicitly specify the ***recipient*** option as well.

For example, the following command line uses the certificate for tom.jefferson@wash.com and the one for sam.adams@wash.com to encrypt the files added to `files.zip`. The example assumes that ***cryptalgorithm*** has been configured to specify an encryption algorithm. Otherwise, the algorithm needs to be specified on the command line.

```
pkzipc –add –cryptalg –mailto=tom.jefferson@wash.com –mailserver=mail01
–mailfrom=sam.adams@wash.com files.zip *.doc
```

The example above has the same effect as the command line below in which the ***recipient*** option is used (assuming that the same algorithm is configured for ***cryptalgorithm***):

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

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

You need SecureZIP to do certificate-based encryption. You may also require an add-on module.

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.

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 ***undisclosed*** sub-option requires PKZIP to do less processing than the ***each*** sub-option 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 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
never	PKZIP does not overwrite any same-named files	pkzipc –extract –overwrite=never test.zip *.bmp

If you use the ***add*** or ***extract*** command 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 two additional methods for displaying information from a .ZIP file. Specify the desired method as a value in addition to the ***view*** option. These methods include:

- brief - a compact, less informative view of the .ZIP file.
- detail - more information than the default view.

Displaying a Brief View of a .ZIP File

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

pkzipc -view=brief test.zip

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

```
Viewing .ZIP: test.zip

  Length Method     Size  Ratio    Date     Time  Name
  ------ ------     ----  -----    ----     ----  ----
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a red.txt
   8369B DeflatN    3084B 63.2% 06/01/2001  4:50a tan.txt
  ------           ------ -----                   ----
    16KB            6168B 63.2%                      2
```

Displaying a Detailed View of the .ZIP File

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

pkzipc -view=details test.zip

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

```
Viewing .ZIP: test.zip


          FileName: red.txt
          FileType: text
        Attributes: -a-w--------
     Date and Time: Jun 01,2001   4:50:00a
Compression Method: DeflatN
   Compressed Size: 3084
 Uncompressed Size: 8369
       Compression:  63.2% - 2.948 bits/byte
  32 bit CRC value: 87b3c388
Version created by: PKZIP: 4.5
 Needed to extract: PKZIP: 2.0 or later

          FileName: tan.txt
          FileType: text
        Attributes: -a-w--------
     Date and Time: Jun 01,2001   4:50:00a
Compression Method: DeflatN
   Compressed Size: 3084
 Uncompressed Size: 8369
       Compression:  63.2% - 2.948 bits/byte
  32 bit CRC value: 87b3c388
Version created by: PKZIP: 4.5
 Needed to extract: PKZIP: 2.0 or later

-------------------

       Total Files: 2
   Compressed Size: 6168
 Uncompressed Size: 16738
       Compression:  63.2% - 2.948 bits/byte
```

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.

Printing the Contents of a .ZIP File (WIN32)

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 a .ZIP File

test

PKZIP allows you to test .ZIP files to verify that they are not damaged. Before storing an important .ZIP file or sending it to another person, it is a good idea to test it first. For example, if you wish to test the contents of test.zip, type the following:

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, for some reason, the archive has been damaged, use the ***fix*** option to repair the .ZIP file.

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 -recip=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 -recip=xxx foo.zip *.doc

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

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

```
-warning=42,-43
```

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

Treating Warnings as Errors

error

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

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

Multiple warning numbers can be specified, separated by commas:

```
-error=42,43
```

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

```
pkzipc -add -error=42,43 -recip=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 the ***add***, ***extract***, ***test***, and ***view*** commands. 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 the ***add***, ***delete, header***, ***sfx,*** and ***comment*** commands.

Fixing a Corrupt .ZIP File

fix

In the event that a .ZIP file becomes damaged, you may find that it is not possible to extract or perform other PKZIP operations on the contents of the file. The ***fix*** option in PKZIP will attempt to repair damaged .ZIP archives.

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.
```

Please note that when you enter the ***fix*** option, PKZIP will prompt you to enter a new .ZIP file name. In the above example, test1.zip was entered. If you do not enter a file name, the name "pkfixed.zip" will be used. PKZIP scans the original file, attempts to repair the archive, and saves the updated file with the file name provided. The original, damaged file is not updated. The ***fix*** option will not repair all damaged .ZIP files. Depending on the degree of damage to the data within a .ZIP file, you may not be able to recover your files from that archive.

Create a Temporary .ZIP File on a Alternate Drive

temp

Every time you update a .ZIP file, PKZIP creates a temporary work file. Before modifying the original file, PKZIP performs all of its compression and extraction operations on the temporary work file. When the modifications to the .ZIP file are successfully completed, the original .ZIP file is replaced with the updated file (temporary work file). This means you must have as much additional disk space available as was used by the original .ZIP file. For example, if you have an existing .ZIP file of 500K and you are adding another file to it that is 10K compressed, you need additional workspace of at least 510K during the update process.

The ***temp*** option allows you to create the temporary .ZIP file on a drive other than the one on which the original .ZIP file resides. This allows you to update large .ZIP files when space is limited, such as a large .ZIP file on a floppy disk.

Immediately following the ***temp*** option, place the drive and/or path you wish to use for the temporary work file as in one of the following examples:

- UNIX based command line:

```
pkzipc -add -temp=/usr/tmp test.zip readme.doc
```

- DOS based command line:

```
pkzipc -add -temp=z:/public test.zip readme.doc
```

Note: It is necessary to specify a path in addition to the drive letter only if you are in a situation where disk space or access is being limited by subdirectory, such as on a local area network.

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 error and warning messages, requests for input, and so on are displayed.

You may wish to use the ***silent*** option with the ***nofix*** option, which suppresses the "attempt to fix" prompt if PKZIP encounters errors in a .ZIP file. For example:

```
pkzipc -add -silent -nofix test.zip *.doc
```

In this example, PKZIP adds all files in the current directory with a *.doc* extension to a ZIP archive called *test.zip*. All screen output is suppressed. The "attempt to fix" prompt is suppressed as well if PKZIP encounters any errors in the *test.zip* file. For more information on the ***silent*** and ***nofix*** options, refer to Appendix A.

Setting Internal Attributes

ASCII/BINARY

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

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

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

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

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

To prompt the type for each file:

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

Encoding an Archive to Another Type

encode

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

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

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 were run without the ***listfile*** option. With the option, no files are actually added or extracted.

For example, the following command 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:

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

Similarly, when the option is used with ***extract***, a list file is created containing the names of all the files that would have been extracted if the ***listfile*** option were not used.

The ***listfile*** option takes account of any other options used. For example, if files are to be extracted using stored path information, the path information will appear in the list file as well, but if files are to be extracted *without* using stored path information, no path information will appear in the list file either.

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.

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.

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 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.

Using Other Devices (UNIX)

device

This option allows you to specify a device when compressing or extracting spanned files. When creating spanned archives, the device is treated like a 1.44 MB floppy, adding a FAT-12 file system and volume label. This command only works with 1.44 floppy disks and will fail with other disk sizes. Since this command only does a partial format, you must use a formatted floppy disk when using this command to compress files.

Note: The floppy diskette device must not be mounted. If your system automatically mounts diskettes, disable the auto-mount feature before using.

To compress files using floppy diskette /dev/fd0, use the following command:

```
pkzipc -add -device=/dev/fd0 -span save.zip *.doc
```

To extract files using floppy diskette /dev/fd0, use the following command:

```
pkzipc -ext -device=/dev/fd0 save.zip
```

7 Changing Defaults for Commands and Options

Default values for PKZIP commands and options are stored in a configuration file. You can view current settings by using the ***configuration*** command. You also use this command to change default values for any command or option that has a default. Another command—***default***—lets you restore default settings for all commands and options to their original values.

With the ***altconfig*** option, you can create and use alternate configuration files for special purposes.

Viewing the Configuration File

You use the ***configuration*** command to view the current default values in the configuration file. To view default settings, enter the command by itself, with no options. For example:

pkzipc –config

A screen similar to the following appears:

```
204 = Disabled                          Add = Add All Files
ArchiveDate = None                      CD = Normal
Comment = None                          Encode = Disabled, UUE
Comp Method = Deflate                   Level = 5
Extract = Extract All Files             Hash = SHA-1
ListChar = @                            Locale = Disabled
Lowercase = Disabled                    More = Disabled
NoArchiveExtension = Disabled           NoExtended = Disabled
NoFix = Disabled                        NoSmartCard = Disabled
OptionChar = -                          Password = Disabled
Recurse = Disabled                      Shortname = None
Sort = None                             Span = None, Auto-Detect
Test = Extract All Files                Times = All
Translate = None - No Conversion        View = Normal
Wipe = None

ASCII = Disabled
Binary = Disabled
Certificate = John Public
CryptAlgorithm = Traditional
Embedded = Disabled
Error = Disabled
Header = Disabled
LDAP = Disabled
```

```
Recipient = Disabled
Sign = Disabled, None
Silent = None
Temp = Disabled
Warning = Disabled

Compression Options
        After = Disabled
        Attributes = Read-Only, Archive
        Before = Disabled
        Exclude = Disabled
        Include = Disabled
        Larger = Disabled, 0
        Mask = None
        Newer = Disabled
        Older = Disabled
        Overwrite = Always Overwrite
        Path = No Path Information
        Smaller = Disabled, 18,446,744,073,709,551,615

Extraction Options
        After = Disabled
        Attributes = Read-Only, Hidden, System, Archive
        Before = Disabled
        Exclude = Disabled
        Include = Disabled
        Larger = Disabled, 0
        Mask = None
        Newer = Disabled
        Older = Disabled
        Overwrite = Prompt
        Path = Full Path
        Smaller = Disabled, 18,446,744,073,709,551,615
```

In this display, the commands/options are to the left of the equal sign, and the default settings are to the right.

With options that take sub-options, the default setting is generally a sub-option. For example, the ***overwrite*** option shows a default of ***prompt***, which is one of its possible sub-options. Some other options are set to ***none***. An option that does not have a ***none*** sub-option shows as *Disabled* when it is not configured. For example, ***After*** shows a date if the option is configured and *Disabled* if it is not.

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.

For information on changing defaults, refer to the section below, “Changing a Default Value.”

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 no sub-options:

- 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
  ```

After you change a default value, the updated configuration file is displayed.

See Appendix A for a list of PKZIP commands, options, and sub-options and information about which commands and options have configurable defaults.

Changing Defaults for Filter Options

Options listed as filter options in the display of default settings take separate defaults for compression and extraction. To specify a default for a filter option for one of these operations, include the related command (***add*** or ***extract***) on the command line. For example:

```
pkzipc –config –add –newer=1d
```

If you specify a default for a filter option without including the related command, as in the following example, PKZIP asks whether you want to specify the default for compression, extraction, or both:

```
pkzipc –config –newer=1d
```

Changing Defaults for Compression Method

The *Comp Method* item in the screen of configuration settings shows the current default setting for compression method. To set a default compression method, specify the compression method that you want to make the default. For example, the following command makes BZIP2 the default compression method:

```
pkzipc –config –bzip2
```

The options in the table below set compression method:

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
store	Sets the compression method to Store (that is, no compression)

The options in the next table set both compression method and level:

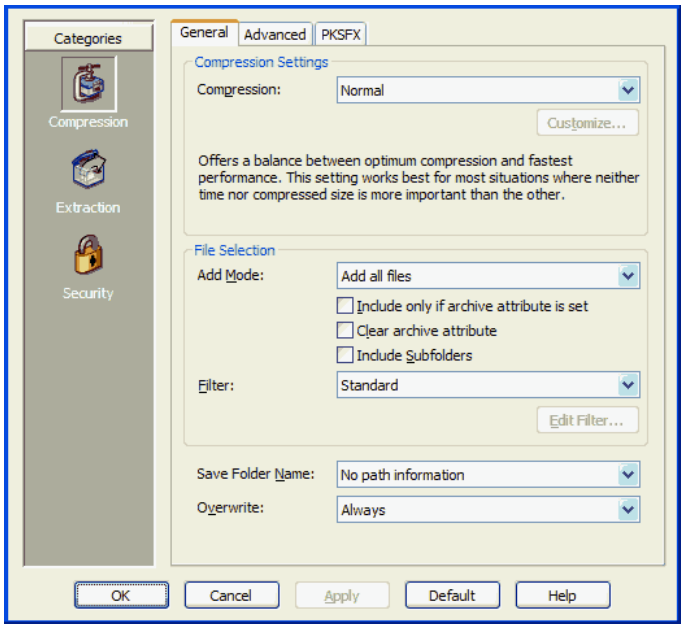
Option	*Description*
speed	Sets the compression method to Deflate—the initial PKZIP default method—and the level of compression to 1 (the lowest)
fast	Sets the compression method to Deflate and the level of compression to 2
normal	Sets the compression method to Deflate and the level of compression to 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 installed, you can use the graphical Options dialogs instead of the command line to change defaults:



Compression Settings
Compression: Normal

Offers a balance between optimum compression and fastest performance. This setting works best for most situations where neither time nor compressed size is more important than the other.

File Selection
Add Mode: Add all files
- [ ] Include only if archive attribute is set
- [ ] Clear archive attribute
- [ ] Include Subfolders

Filter: Standard

Save Folder Name: No path information
Overwrite: Always

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 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 default without resetting any other Configuration 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***.

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 and use alternate configuration files to use for special purposes. The ***altconfig*** option creates such files and loads them. With an alternate configuration file, you can temporarily change multiple default command or option settings in a single pass just by loading the configuration file that defines them.

Creating an Alternate Configuration File

An alternate configuration file must be in valid format for a PKZIP Server XML configuration. The easiest way to create one is to 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 not overwritten. Instead, default settings in that file are updated with the new defaults from the command line.

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 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.

An alternate configuration file must already exist for you to use it in a command line with the ***add*** command or any other command besides ***configuration***. The only time you can use the ***altconfig*** option to specify an alternate configuration file that does not already exist is when you use the option with the ***configuration*** command to create an alternate configuration file.

8 Command Characteristics

This chapter describes changes you can make to the PKZIP infrastructure. For example, you can specify different characters to use for the list character and the option character, and you can cause PKZIP to display dates and times using a different format from the one used by default on your system.

Ordinarily, the original values for the settings described in this chapter should be satisfactory. You should not change them without a good reason.

Changing Date and Time Environment Variables

locale

The ***locale*** option causes PKZIP to use your system's format for displaying dates and times. The option has two sub-options, ***enable*** and ***disable***, to set it on or off. The option is configurable and is set on by default.

Formerly PKZIP used a date format of MMDDYY and a 12-hour time format of HH:MM. If you prefer PKZIP to use this format, you can revert to it by setting ***locale*** to ***disable***.

If you have disabled the ***locale*** option by default, you can enable it for a particular command line by setting the option to ***enable*** in the command line. For example:

```
pkzipc –add –locale=enable test.zip *.doc
```

This command line causes PKZIP to use the system-defined settings regardless of the default settings.

Changing the List Character for List Files

listchar

PKZIP allows you to specify an ASCII file as a source list of the files to be archived. By default, you specify this ASCII file by pointing to it with the "@" character in your command line. However, if you have files that begin with an "@", you may experience problems when trying to add these files to a .ZIP archive. Fortunately, PKZIP allows you to change the default list character to avoid such problems. This is accomplished using the ***listchar*** option. For example, if you wish to define the "+" character in place of the "@" as your default list character, type the following and press ENTER:

```
pkzipc -config -listchar=+
```

If you wish to specify an alternate list character on the command line itself, could type a command line similar to the following and press ENTER:

```
pkzipc -add -listchar=+ test.zip +file1.txt
```

When used as a command line option, the ***listchar*** option only applies to the options that follow it on that particular command line. In our example the ***listchar*** option allows you to add files that begin with an "+" character (e.g., +file1.txt). For more information on using list files with PKZIP see the section on page 51 and the “Extracting Files with a List File” section on page 99.

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 -opt=+ +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:

Category	Represents
Name/Description	The full name of the command/option and a brief description of what that command/option does. If the command/option can be configured (defaulted) in the configuration file, the word "Configurable" appears after the description.
Value(s)	The value(s) associated with this command/option, including the "default" value for each. If a command/option does not have an associated value, the phrase "no sub-options" appears.
Example usage	An example of how to include this command/option in your PKZIP command line, including possible abbreviations. For most options, you can abbreviate the command/option. These abbreviations are illustrated in the examples used in this appendix.
Used with	This command can be used for compression, extraction, viewing, testing, a combination, or as a standalone (none of the above).

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 file Configurable	**all** – Compress and add files that are new to the archive as well as files that the archive already contains a copy of **archive** – Turn off archive attribute of all added files (prepares backup file set for incremental archiving) (WIN32). **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 (WIN32) **–incremental** – Add only files that have the archive attribute on, and do not turn off the archive attribute afterward (WIN32) --------------------- Default = **all**	pkzipc –add save.zip *.doc pkzipc –add=freshen save.zip *.doc pkzipc –add=increm save.zip *.doc pkzipc –add=–increm save.zip *.doc	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 –aft=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 –cryptalg=aes,256 –sign=all –cert="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 –pass foo.zip *.doc	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
archivedate Set the file modification date of the archive file. Configurable **Note:** The archivedate option is the same as the older **zipdate** option, which is now deprecated.	**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 = the current date. Default if used on command line without a sub-option = retain.	pkzipc –add=update –archivedate=retain save.zip *.txt	add, delete, fix, header, comment, sfx

Name/Description	*Value(s)*	*Example usage*	*Used with*
archivetype Specifies the type of archive to create. Normally, the archive type can be inferred from the specified name of the file to create. It is necessary to use the archivetype option only if the archive type cannot be inferred (for example, because the file name has no extension). If this option is not specified and the file name does not indicate the archive type, a ZIP archive is created. Configurable	**bzip2** – Specifies the Bzip2 archive type.* **zip** – Specifies the .ZIP archive type. (default) **gzip** – Specifies the GZIP 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.	pkzipc –add –archivetype=tar myfile.foo Creates a TAR archive named myfile.foo.tar	add
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. (WIN32)	**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 –attr=system,hidden save.zip *	add
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 –bef=09152003 save.zip *.doc For extraction: pkzipc –ext –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 *password* 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 –password=secret –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	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, header

Name/Description	Value(s)	Example usage	Used with
	more than one certificate in the MY store has the specified name, the first certificate is used. **#<filename> –** 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 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 within .ZIP file. **unchanged** – only existing files that have not changed. **add** – only new files. **freshen** – only existing files. **update** – existing and new files. **none** – turn off comment. --------------------- Default = **add**	pkzipc –add –com=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. --------------------- 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	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
cryptalgorithm Encrypts files using the specified strong encryption algorithm. Configurable	The encryption algorithm to use, as listed in the output from the listcryptalgorithms command. The listcryptalgorithms command lists the algorithms available to you. Default = AES, 256	pkzipc –add –cryptalgorithm=aes,128 –password save.zip *.doc encrypt all files added with 128–bit AES using the specified password. pkzipc –add –cryptalgorithm=3DES,168 –recipient="My friend" save.zip *.doc encrypt all files added with 3DES using the certificate named "My friend".	add

Name/Description	Value(s)	Example usage	Used with
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
deflate64 Compress files using the Deflate64 method. Configurable **Note:** Files compressed with this method can be extracted with most versions 2.5x and later of PKZIP. Other .ZIP programs generally cannot extract files compressed with this method.	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 –del save.zip doc1.txt For a specific file pattern: Pkzipc –del save.zip *.doc	standalone

Name/Description	*Value(s)*	*Example usage*	*Used with*
device Allows spanning to a specified device. (UNIX) N**ote**: You must use a formatted floppy when compressing files. The floppy device must NOT be mounted. If you system automatically mounts diskettes, disable the auto–mount feature before using PKZIP.	Device such *as /dev/fd0* No default value.	pkzipc –add –device=/dev/fd0 –span save.zip *.doc Compresses *.doc into save.zip on the floppy diskette in /dev/fd0, prompting for further disks as necessary pkzipc –ext –device=/dev/fd0 save.zip Extracts all the files in save.zip, which is on a DOS formatted floppy diskette.	add, extract, test, view, delete, print, console
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** – Store the amount of path information passed on the command line with the file name **relative** – Same as **current** (WIN32) **none** – No path information stored Default = **none** Default if used on command line without a sub-option = **current**	Compression example (assumes you are in "/wp"): pkzipc –add –dir=root save.zip wp/docs/* The path stored would be "wp/docs/". pkzipc –add –dir=current save.zip wp/docs/* The path stored would be: "docs/". Extraction example: 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 1.	add, extract
embedded Suppresses prompting whether to extract the contents of a lone embedded archive file of the type specified with the sub-option.	**arj** – Extract the contents of an embedded ARJ archive without prompting **–arj** – Do not extract the contents of an embedded ARJ archive and do	To extract an embedded ZIP file without prompting: pkzipc –extract –embedded=zip outerarchive.zip To suppress the prompt and not extract any embedded ZIP file:	extract, console, print

Name/Description	Value(s)	Example usage	Used with
The option can be used to tell PKZIP either to extract the contents of an embedded archive automatically, without prompting, or to never extract the contents of an embedded archive automatically. Configurable	not prompt **BinHex** – Extract the contents of an embedded BinHex archive without prompting **–BinHex** – Do not extract the contents of an embedded BinHex archive and do not prompt **bzip2** – Extract the contents of an embedded BZIP2 archive without prompting **–bzip2** – Do not extract the contents of an embedded BZIP2 archive and do not prompt **cab** – Extract the contents of an embedded CAB archive without prompting (WIN32 ONLY) **–cab** – Do not extract the contents of an embedded CAB archive and do not prompt (WIN32 ONLY) **gzip** – Extract the contents of an embedded GZIP archive without prompting **–gzip** – Do not extract the contents of an embedded GZIP archive and do not prompt **lzh** – Extract the contents of an embedded LZH archive without prompting **–lzh** – Do not extract the contents of an embedded LZH archive and do not prompt **rar** – Extract the	pkzipc –extract –embedded= –zip outerarchive.zip	

Name/Description	Value(s)	Example usage	Used with
	contents of an embedded RAR archive without prompting (WIN32 ONLY) **–rar** – Do not extract the contents of an embedded RAR archive and do not prompt (WIN32 ONLY) **uue** – Extract the contents of an embedded UUENCODED archive without prompting **–uue** – Do not extract the contents of an embedded UUENCODED archive and do not prompt **xxe** – Extract the contents of an embedded XXENCODED archive without prompting **–XXE** – Do not extract the contents of an embedded XXENCODED archive and do not prompt **ZIP** – Extract the contents of an embedded ZIP archive without prompting **–ZIP** – Do not extract the contents of an embedded ZIP archive and do not prompt --------------------- No default value. A sub-option must be set or you are prompted.		

Name/Description	Value(s)	Example usage	Used with
encode Converts an archive to the archive type specified by the sub-option 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 (default) **xxe** – Creates an XXENCODED file No default value.	Add files to save.zip and encode to the configured default *encode* format (UUE, initially): 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 *	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.	Compression example: pkzipc –add –excl="*.doc" save.zip Extraction example: pkzipc –ext –excl="*.txt" save.zip Setting exclude default: pkzipc –config –excl="*.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 –ext=up 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

Name/Description	Value(s)	Example usage	Used with
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 filenames that have a dot "." in the first position of the filename (e.g., .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" (e.g., prwxrw–rw–). **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–). **all** – include all of the above file types. **none** – exclude all of the above file types; generally, this option should be followed by one, or more, file types. This results in just the type(s) specified being included in the ZIP file. For example,	pkzipc –add –filetype=all save.zip	add

Name/Description	***Value(s)***	***Example usage***	***Used with***
	–filetype=none, pipe results in only PIPE files being included. Default = **regular**		
fix Attempts to repair a corrupt ZIP archive file	**<filename>** – The name of the ZIP archive to fix No default value.	pkzipc –fix save.zip	standalone

Name/Description	***Value(s)***	***Example usage***	***Used with***
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 **password** (optional) is the password associated with the user account. Colons are not allowed in the password. **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 –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*
hash Sets the hashing algorithm to use when signing an archive Configurable **Note:** This option requires SecureZIP.	**sha1** – sign files or central directory using the SHA-1 hash algorithm. **md5** – sign files or central directory using the MD5 hash algorithm. default = **sha1**	pkzipc –add –certificate=”John Smith” –hash= –sha1 save.zip *.doc	add, certificate
header Creates a comment for a ZIP archive file in the header area of the file Configurable	**<filename>** – 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 –hea save.zip *.doc Note: when you type this command, PKZIP will prompt you for the header text To include an existing file: pkzipc –add –hea=@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
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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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.	pkzipc –add –include="*.doc" save.zip In this example, only .doc files will be compressed. pkzipc –config –include="*.txt" In this example, you are setting up .txt files as the files that you always want to compress or extract, until you change the default or override from the command line with the exclude option. Note: When you use the *include* option with the configuration command, PKZIP prompts you to configure the *include* default 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
keyfile Specifies the file containing the private key for PKZIP to use, for example, with the **certificate** option Configurable	<**filename**> – The name and location of the file	pkzipc –add –certificate=#mycert.pem –keyfile=mykey.key save.zip *.doc	add
keypassphrase Specifies a keypassphrase to decrypt an encrypted private key for PKZIP to use, for example, with the **certificate** option	<**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

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 (in bytes) that indicates a minimum desired file size. No default value.	pkzipc –add –larger=5000 save.zip * In this example, PKZIP adds only files that are at least 5000 bytes in size.	add, extract, test, view, delete, print console

Name/Description	***Value(s)***	***Example usage***	***Used with***
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 :password@] server[:port]/] ldap_base where: **userid** (optional) is the user account with which to log in if the LDAP server requires a login **password** (optional) is the password 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. **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 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 70.	pkzipc –add -ldap=john_p:secret@192.17 2.0.1:389 /cn=users,dc=xyz,dc=com -recipient=”Mary Samplename” save.zip *.doc pkzipc –add -ldap=cn=users,dc=xyz, dc=com -recipient=e=mary.samplena me@ xyz.com save.zip *.doc	add

Name/Description	*Value(s)*	*Example usage*	*Used with*
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 –lev=9 save.zip *.doc	add
license Displays the product license information for PKZIP	No sub-options. No default value.	pkzipc –lic	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**	pkzipc –add –links=hlink save.zip This example compresses regular and hard linked files and duplicates link and file data for each hard linked file added to the .ZIP file.	add
listcertificates Displays a list of available digital certificates	No sub-options. No default value.	pkzipc –listcertificates	standalone

Name/Description	*Value(s)*	*Example usage*	*Used with*
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 under your PKZIP license **Note:** This option is only available in versions that have strong encryption.	None	pkzipc –listcryptalgorithms	standalone
listfile Generates a 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.	pkzipc –add=update –listfile=mylist.txt myarchive.zip * In this example, PKZIP generates an ASCII text file (list.txt) that lists all files in the current directory: pkzipc –add –listfile=list.txt * In this example, PKZIP generates a list file that lists all files in the save.zip archive: pkzipc –extract –listfile=list.txt save.zip	add, extract
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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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**	Configures the option to be off by default: pkzipc –config –locale=disable Turns 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
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 <**filename**> – 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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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 <**filename**> – 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
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	**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.	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 names in lower case regardless of how they appear in the archive Configurable	No sub-options. --------------------- No default value.	pkzipc –extract –lowercase save.zip *	extract

Name/Description	Value(s)	Example usage	Used with
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**> – A file name that contains the text to use for the message body. The file 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
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 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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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

Name/Description	Value(s)	Example usage	Used with
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**, 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**	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
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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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 password. The user name and password are both optional. For example, either of these works: mailserver=user@ mail.abc.com Note the colon prefixing the password: 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
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

Name/Description	*Value(s)*	*Example usage*	*Used with*
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: They must be configured or explicitly appear 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 **mailTo recipient** sub-option). 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 –cryptalg –mailto=tom.jefferson @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 (WIN32) **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** Default if used on command line without a sub-option (add and extract) = **all**	pkzipc –add –attr=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 –max save.zip *.doc pkzipc –config –max	add
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
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, such as "five days" Configurable separately for add and extract operations. **Note:** Specifying a newer value is functionally equivalent to specifying an after value.	**<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 no older than 24 hours: pkzipc –add –newer=24h save.zip * Adds 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 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, such as "five days" Configurable separately for add and extract operations. **Note:** Specifying an older value is functionally equivalent to specifying a before value.	**<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
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 –opt=+ +add save.zip *.doc pkzipc +config –option=+	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 **never** – Never overwrite a file that already exists in the target directory or archive --------------------- Default if used on command line without a sub-option = all.	pkzipc –ext –over=all save.zip pkzipc –add –overwrite=prompt save.zip	add, extract

Name/Description	Value(s)	Example usage	Used with
password Protects an archive with password-based encryption PKZIP prompts for a password if none is specified with the option. Configurable	**<password>** – The password that must be supplied to extract and decrypt the files --------------------- No default value.	To include a password in the command: pkzipc –add –pass=beowulf save.zip To have PKZIP prompt for a password after you type the command: pkzipc –add –pass save.zip To extract password-protected files from an archive: pkzipc –extract –password=beowulf 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 1.	**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** – Store the amount of path information passed on the command line with the file name **relative** – Same as **current** (WIN32) **none** – No path information stored --------------------- Default = **none** Default if used on command line without a sub-option = **current**	Assuming in 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 would be "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.	pkzipc –extract –permission save.zip In this example PKZIP will preserve all permissions as well as other attributes on extraction. pkzipc –extract –permission=4111 save.zip In this example PKZIP will preserve and/or attempt to modify all permissions as well as other attributes on extraction.	extract
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. (WIN32)	**<print device>** – The print device use, for example, "lpt1". --------------------- Default = the default printer on your system.	pkzipc –print=lpt1 save.zip readme.txt If you do not specify a print device, your default printer is used.	standalone

Name/Description	*Value(s)*	*Example usage*	*Used with*
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
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. **@<filename>** – Specifies a text file	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*
	which contains the names of recipients, one on each line. **#<filename>** – Specifies a PKCS#7 or PKCS#12 file that contains certificates of the recipients you want to list. --------------------- Default = **cn=**		
recurse Search subdirectories for files to compress Use with path to store path information for files in subdirectories. Or use directories, which combines 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

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.	pkzipc –add –sfx –runafter="notepad.exe readme.txt" test.exe * Launches the file (for example, readme.txt) via the specified applications (for example, notepad.exe). pkzipc –add –sfx –runafter="${} readme.txt" test.exe * Launches the file (for example, readme.txt) via the associated application (WIN32 only). pkzipc –add –sfx –runafter="${install.inf}" test.exe * Runs the install script (for example. install.inf) (WIN32 only) pkzipc –add –sfx –runafter= "${install}%0install.inf" test.exe Runs the install script (for example, install.inf) with the full short path pre–appended (for example, c:\program~1\temp) (Win32 only).	(add) sfx
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

Name/Description	*Value(s)*	*Example usage*	*Used with*
sfxdestination Specifies a default target folder for files extracted from a self-extractor Configurable	No sub-options --------------------- 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
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

Name/Description	Value(s)	Example usage	Used with
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 --------------------- No default value.	pkzipc –add –short=dos save.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. **none** – do not sign files (Used for turning signing off if it has been configured) --------------------- Default = **all**.	pkzipc –add –certificate="John Smith" –sign=cd save.zip *.doc	add

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 **Note:** If the silent option is enabled by default in the configuration file, PKZIP prompts for confirmation every time you attempt to modify the configuration file.	**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 **input** – Suppresses all requests for input. If any operation requests input, an error is given **normal** – Suppresses all normal outputs **output** – Suppresses all normal, error, and warning outputs **progress** - Suppresses "percent complete" messages **all** – Same as specifying both Input and Output. (Default) --------------------- No default value.	pkzipc –add –silent save.zip *.doc pkzipc –config –silent	All commands except list-certificates, listcryptalgo-rithms, listsfxtypes, license, and version
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 (in bytes) --------------------- 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

Name/Description	***Value(s)***	***Example usage***	***Used with***
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 were compressed. **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** Default 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	**Format** – fully format media before attempting to write to it. **Quick** – quick format media before attempting to write to it. **Wipe** – erase contents of media before attempting to write to it. **None** – no media format or erase of media contents before attempting to write to it. **<segment size>** – split files into predefined size (see choices below) or a specified size (in bytes) greater than 65536. 360K, 720K, 1.2MB, 1.44MB, 2.88MB, 95.7MB, 650MB, 700MB --------------------- No default value.	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
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
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.	pkzipc –add –temp=z:\public test.zip *.txt This example updates the .ZIP file test.zip and uses the z:\public directory location for temporary files. pkzipc –add –temp=/temp test.zip *.txt This example updates the .ZIP file test.zip and uses the /temp directory location for temporary files.	add, delete, sfx, header, comment

Name/Description	Value(s)	Example usage	Used with
test Tests the integrity of files within a .ZIP file 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
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 (WIN32). **all** – all file times are restored. **none** – file times are not restored. --------------------- Default = **none**	pkzipc –extract –times=access save.zip	extract

Name/Description	Value(s)	Example usage	Used with
translate Translates EOL (“end of line”) characters when 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. Configurable	**none** – no translation is performed. **dos** – translates text files so that lines end with a return/newline pair (Win32 default) **mac** – translates text files so lines end with a single carriage return **unix** – translates text files so lines end with a single newline --------------------- Default = **none** Default if used on command line without a sub-option = native operating system compatibility translation.	pkzipc –extract –translate=unix save.zip	extract, console, print
version Identifies the version of PKZIP	**major** – returns the major version number of the release. **minor** – returns the minor number of the release. For example, if the version number is 2.5.1, the value returned is 5. **step** – returns the step, or patch value. For example, if the version is 2.04.01, the step value returned is 1. --------------------- Default = **major**	pkzipc –version	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. --------------------- 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 –ext –warn=43 save.zip * pkzipc –ext –warning save.zip * pkzipc –ext –warn=–43 save.zip *	add, extract, test, view
wipe Overwrites files deleted by PKZIP to prevent recovery of their data Configurable	**none** – turns off wiping: files are not overwritten **random** – Overwrites files once with random data **nsa** – Overwrites files seven times, to the NSA standard --------------------- Default = **none** Default if used on command line without a sub-option = **random**	pkzipc –add –move –wipe=nsa myfiles.zip *	add

Name/Description	*Value(s)*	*Example usage*	*Used with*
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 = the current date. Default if used on command line without a sub-option = retain.	pkzipc –add=update –zipdate=retain save.zip *.txt	add, delete, fix, header, comment, sfx

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.

Error	*Potential Cause(s)*
(E5) Unknown or illegal sub-option - XXX.	The sub-option you specified on the command line is invalid. It does not match any known sub-options. Verify that you typed the sub-option on your command line correctly. Verify that you are not using an illegal sub-option (-add -sort=crc). Check the spelling.
(E6) No .ZIP file specified.	There was not a .ZIP file specified on the command line. PKZIP does not accept wildcards for .ZIP file name when adding files to a .ZIP archive.
(E7) Can't create: XXX.	PKZIP could not create a .ZIP file when fixing. PKZIP could not create a volume label on a spanned archive. PKZIP could not create a temporary file for a spanned archive. Verify that you have write access to the drive or diskette on which you are creating these files.
(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)

Error	*Potential Cause(s)*
(E13) Can’t open .ZIP file: XXX.	The named .ZIP file is read-only or locked by another application and can not be modified. This may also occur on a Network drive if you do not have sufficient access rights to the file to allow you to modify it.
(E14) Can't create archive: XXX.	PKZIP is not able to create the archive file. Verify that the destination directory is not full, the archive file does not already exist. If you are creating the file on a network drive, confirm that you have the appropriate rights to the network file system.
(E15) Renaming temporary .ZIP file, saved as: XXX.	PKZIP could not rename the temporary file to the specified .ZIP file name. Verify that the destination drive is not full. If you are updating a non-spanned .ZIP file on removable media (floppy diskette) and the updated archive exceeds the size available on the removable media, you will receive the (E15) error. You will need to recreate the archive for spanning. Keep in mind that you cannot update a spanned archive. If you are creating the file on a network drive, confirm that you have the appropriate rights to the network file system.
(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.
(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. See if you can abbreviate the name of the option or its sub-option to make it shorter.

Error	*Potential Cause(s)*
(E24) Insufficient disk space for ZIP comment.	There is not enough space on the system or media to write the ZIP comment.
(E25) Insufficient disk space for updated file.	Insufficient disk space for the new archive. If you are adding files to an archive on a removable media, the media may not be large enough to write the modified file (too large).
(E26) Device not ready: XXX.	The removable media device is not ready. The disk may not be in the drive properly.
(E27) 2.04g compatibility cannot be used with the option - XXX	Option **204**, which creates an archive compatible with PKZIP for DOS v. 2.04g, was used with another option that is not supported for that version of PKZIP
(E34) Invalid archive format: <archive name>	The file is not in a format currently supported by PKZIP
(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 files 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 password or certificate not found, unable to open ZIP archive: <archive name>	The archive contains encrypted file names that PKZIP cannot decrypt. If the archive is password-protected, you must include the *password* 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.
(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.

Error	***Potential Cause(s)***
(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.
(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 option 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 page 51 for more information on LIST files.
(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 option 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.
(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.
(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 password protected. You need the password 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.

PKZIP Warning	*Potential Cause(s)*
(W20) Incorrect password for file: XXX.	Verify that you entered the correct password for the file. When a file is password protected, you can only access its contents with the correct password. **Note:** Passwords 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 password, files will not be password protected.	When trying to password protect your file, you entered a password containing no letters or numbers.
(W37) Can't sign file.	This warning appears when PKZIP fails to sign a file using the specified digital certificate. Common reasons are incorrect password for the certificate (not all certificates have passwords), 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 password was supplied to access the certificate (not all certificates have passwords) 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.

PKZIP Warning	*Potential Cause(s)*
(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, PKZIP will not strongly encrypt files for recipients. If you did not tell it to use passwords; that is, you did not use the –password option, it will not encrypt files at all. In addition, if you specify –password 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.
(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.

PKZIP Warning	*Potential Cause(s)*
(W57) Incorrect password or recipient not found for file: XXX	Verify that you entered the correct password for the file. When an archived file is password protected, you can only access the file if you have the correct password. Passwords 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 (*password*, *recipient* options), encryption algorithm (*cryptalgorithm* option).
(W63) You must specify –password or –recipient to encrypt files!	You specified *–cryptalgorithm* or *–cd=encrypt* but did not specify either the *recipient* or *password* 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.
(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 password is supplied to use for password 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 password is supplied to use for password encryption.

PKZIP Warning	***Potential Cause(s)***
(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 password needed to access the certificate that contains the key may have been reset or changed by an administrator. Get help from your system administrator.

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.

The ***freshen*** option 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.

The ***update*** sub-option 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 password; what do I do?

- Try to remember the password.
- Try passwords that are "close" to what you think it was.
- Try mixed upper and lower case versions of your password.

Do not forget or lose your passwords! 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 password(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 MS-DOS, Windows(98, NT, Me, 2000, XP), OpenVMS, HP-UX, IBM AIX, Linux, Sun Solaris, MVS/ESA, OS/390, z/OS, VSE, and OS/400 platforms. PKWARE intends to support additional platforms and will announce this support as it becomes available.

Because PKWARE has dedicated the .ZIP file format to the public domain, it is possible for other people to write programs which can read .ZIP files. We are aware of PKZIP compatible programs for a number of different platforms. A .ZIP file can be transferred to any platform for which you can find a compatible extraction program. Extraction and Compression programs not developed by PKWARE may not be completely compatible with the .ZIP file standard. Contact PKWARE for a list of platforms for which PKZIP and PKZIP compatible software is available.

D How PKZIP Works

This Appendix provides a description of how PKZIP actually does its job. It is not necessary for you to know or understand the information presented here, any more than you need to know how your carburetor works to drive a car. It is presented to help you feel more knowledgeable about the software.

Two Processes

PKZIP performs two functions: compression and archiving. Although the two ideas may seem related, they are actually completely separate.

- **Compression** is the process of representing a given piece of information with a smaller piece of information.
- **Archiving** is the process of combining many files into a single unit, usually along with vital information about each file.

Compression

The actual process used by PKZIP for its compression is too complex to explain in detail. Instead, some of the general principles behind information theory and compression are explained.

To understand data compression, you need to understand two ideas: Information Content and Binary Coding.

Information Content

Everything in your computer, everything you ever read, is "information". The more complex a message is, the higher the information content. The less complex, the less "random" a message is, the lower the information content.

If a message contains a low amount of information, it should be possible to represent it in a smaller amount of space. Look at this page, for example. How much of the page is white space with no letters (information) on it? If you took away all of the white space this page would be significantly smaller. How many times are the words "the", "information" and "compression" on this page? If you could replace each of these words with something smaller, you would save a significant amount of space.

The more frequently the same group of symbols (in this case, letters) appear, the lower the information content of the message.

The "Field of Information Theory" uses the term *entropy* to describe the "true" information content of a message. Formulas can be used to determine the *entropy* of a message. The idea behind data compression is to derive a new smaller message from a larger original message, while maintaining the *entropy* of the original message.

As a simple example, consider this sentence:

> she sells sea shells by the sea shore

This sentence is 37 characters long, including spaces. The spaces cannot be simply thrown away as the meaning of the original message would be lost.

There are obvious patterns to the sentence. The combination 'se' appears three times, 'sh' three times, and 'lls' twice. In fact, the 'se' pairs all have a space in front of them, so these can be ' se'.

> *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, 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.

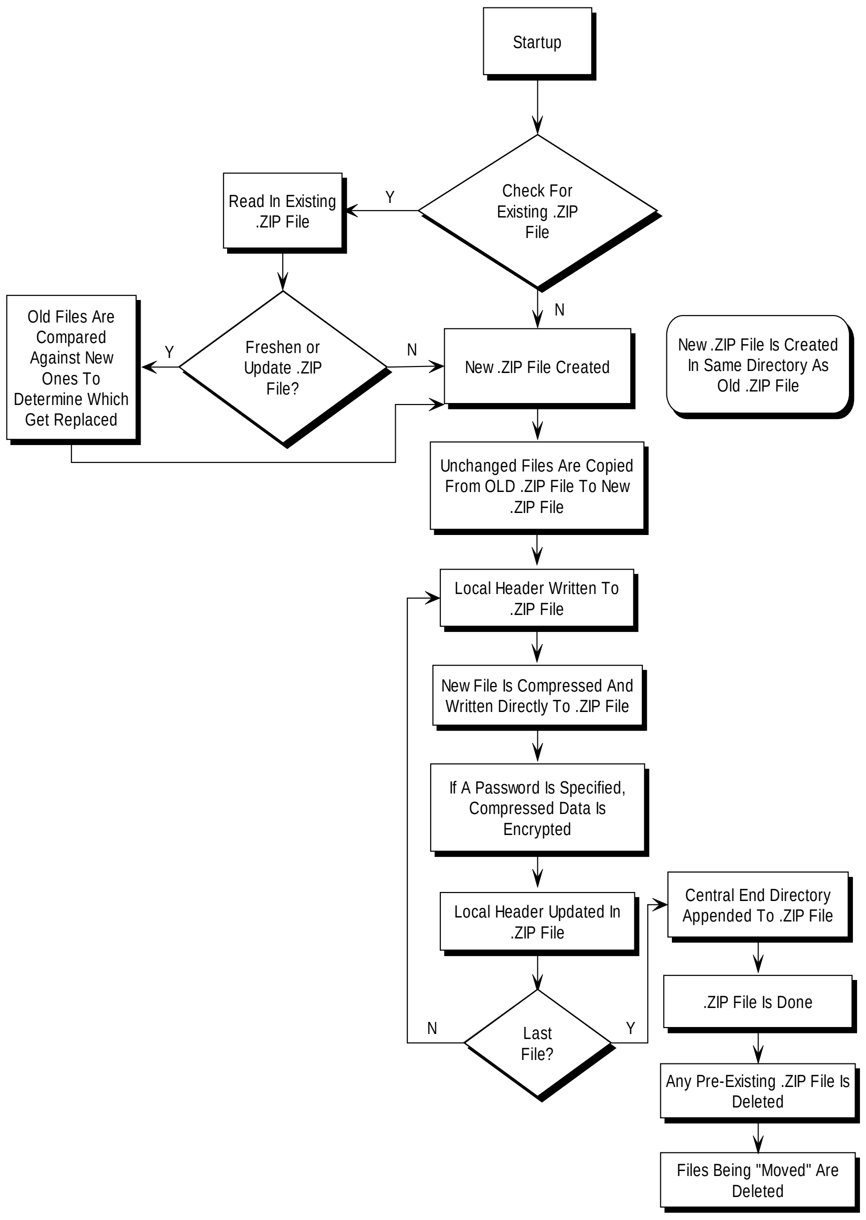
2. Reserves the memory it will need to perform the compression, archiving and buffering.

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


Startup
Check For Existing .ZIP File
Y
Read In Existing .ZIP File
N
Freshen or Update .ZIP File?
Y
Old Files Are Compared Against New Ones To Determine Which Get Replaced
N
New .ZIP File Created
New .ZIP File Is Created In Same Directory As Old .ZIP File
Unchanged Files Are Copied From OLD .ZIP File To New .ZIP File
Local Header Written To .ZIP File
New File Is Compressed And Written Directly To .ZIP File
If A Password Is Specified, Compressed Data Is Encrypted
Local Header Updated In .ZIP File
Last File?
N
Y
Central End Directory Appended To .ZIP File
.ZIP File Is Done
Any Pre-Existing .ZIP File Is Deleted
Files Being "Moved" Are Deleted

- 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 message "Nothing to do!".

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 passwords.
- 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
```

Index