PKZIP for iSeries

User's Guide

PKIU-V5R6000

PKWARE, Inc.

PKWARE, Inc.

9009 Springboro Pike

Miamisburg, Ohio 45342

Sales: 937-847-2374

Support: 937-847-2687

Fax: 937-847-2375

Web Site: http://www.pkzip.com

Sales - E-Mail: pksales@pkware.com

Support - http://www.pkzip.com/support

5.6 Edition (2003)

PKZIP for iSeries™**,** ***PKZIP for MVS***™**,** ***PKZIP for zSeries***™**,** ***PKZIP for OS/400***™**,** ***PKZIP for VSE***™**,** ***PKZIP for UNIX***™**, and** ***PKZIP for Windows***™ **are just a few of the many members in the PKZIP® family. PKWARE, Inc. would like to thank all the individuals and companies -- including our customers, resellers, distributors, and technology partners -- who have helped make PKZIP® the industry standard for Trusted ZIP solutions. PKZIP® enables our customers to efficiently and securely transmit and store information across systems of all sizes, ranging from desktops to mainframes.**

This edition applies to the following PKWARE of Ohio, Inc. licensed program:

PKZIP for iSeries™ **(Version 5, Release 6, 2003)**

Preface

The ***PKZIP®*** family of products consists of high performance data compression software. The archives resulting from compression by ***PKZIP for iSeries*** can be transported or transmitted to other operating system platforms where they will undergo decompression by PKUNZIP or an acceptable substitute.

Notices

- Licensing requirements have changed for this release. See Licensing Requirements in Chapter 3 for more details.

About this Manual

This manual provides the information needed to utilize ***PKZIP for iSeries*** in an operational environment. It is assumed that people using this manual have a good understanding of (Control Language) CL and dataset processing. Note that the contents of this manual applies to the following operating systems:

  - OS/400
  - iSeries

- Chapter 1. An Introduction to PKZIP for iSeries provides a general description of the PKZIP product, which is applicable to all supported platforms. It goes on to describe the OS/400 specific features of the PKZIP for iSeries product and provides a simple description of how the PKZIP for iSeries product is used to provide the compression and decompression of datasets.

- Chapter 2. Provides all of the general getting started information for invoking PKZIP and PKUNZIP. This chapter explains the details associated with compression, decompression, restrictions, migration, and a complete view of the general PKZIP for iSeries features

- Chapter 3. Provides detailed information and guidance on Installation from tape, CD-ROM, using FTP, general installation procedures and licensing procedures.

- Chapter 4. Provides all of the general file selection and naming processing to include: primary file selection, exclusions, and ZIP Archive files.

- Chapter 5. Provides a summary of the ZIP file data formats, such as, Text verses Binary, and file attributes.

- Chapter 6. Provides all of the general rules for extracting files to the QSYS Library file system, IFS, and spool files

- Chapter 7. Provides information regarding the OS/400 file processing support for library, integrated file system, SAVF (Save Files), and Spool Files.

- Chapter 8. Provides detailed information regarding old ZIP files, temporary archives, and new ZIP archives.

- Chapter 9. Provides a summary of all of the PKZIP commands that provides parameters with keyword formats and details.

- Chapter 10. Provides a summary of all of the PKUNZIP commands that provides parameters with keyword formats and details.

- Chapter 11. Provides a description of the GZIP format archive and the GZIP processing supported by PKZIP for iSeries.

- Chapter 12. Provides a description and a summary of all PKZIP generated messages.

- Glossary - Contains a glossary of OS/400 specific terms.

- Index - Contains a detailed list of terms and their locations within this document.

This manual is intended for persons responsible for implementing and using PKZIP Version 5.6 for iSeries. The manual assumes that the reader has a good understanding of CL and dataset processing.

Conventions Used in this Manual

Throughout this manual, the following conventions are used:

The use of the Courier font indicates text that may be found in control language (CL), parameter controls, or printed output.

The use of *italics* indicates a value that must be substituted by the user, for example, a dataset name. It may also be used to indicate the title of an associated manual or the title of a chapter within this manual.

Bullets (•) indicate items (or instructions) in a list.

The use of <angle brackets> in a command definition indicates a mandatory parameter.

The use of [square brackets] in a command definition indicates an optional parameter.

A vertical bar (|) in a command definition is used to separate mutually exclusive parameter options or modifiers.

Related Publications

The ***PKZIP®*** family of products includes the following manuals:

- ***PKZIP for zSeries™* User's Guide**
- ***PKZIP for zSeries™ & PKZIP for VSE™* Messages and Codes**
- ***PKZIP for VSE™* User's Guide**
- ***PKZIP for Command Line - UNIX™* User's Guide**
- ***PKZIP for Command Line - Windows™* User's Guide**

Related IBM Publications

IBM Manuals relating to the ***PKZIP for iSeries™*** product include:

- **System Messages:** This manual documents messages issued by the iSeries operating system. The descriptions explain why the component issued the message, provide the actions of the operating system, and suggest responses by the applications programmer, system programmer, and/or operator.
- **OS/400 CL Programming (SC41-5721):** This manual provides a wide-range discussion of iSeries e Advanced Series programming topics, including: Control language programming, iSeries e Advanced Series programming concepts, Objects and libraries, and Message handling.
- **OS/400 CL Reference (SC41-5722 thru SC41-5726):** This manual may be used in the iSeries Information Center to find information on the following CL Reference topics: OS/400 commands, OS/400 objects, Command description format, Command parts, Command syntax, About syntax diagrams, CL character sets and values, Object naming rules, Expressions in CL commands, and Command definition statements.
- **Integrated File System Introduction (SC41-5711):** This book provides an overview of the integrated file system, which includes: What is the integrated file system? Why you might want to use it; Integrated file system concepts and terminology; The interfaces you can use to interact with the integrated file system. The APIs and techniques you can use to create programs that interact with the integrated file system, and Characteristics of individual file systems.
- **File Management (SC41-5710):** This manual describes the data management portion of the Operating System/400 licensed program. Data management provides applications with access to input and output file data that is external to the application. There are several types of these input and output files, and each file type has its own characteristics. In addition, all of the file types share a common set of characteristics.

- **DDS Reference (RBAF-P000-00):** This manual contains detailed instructions for coding the data description specifications (DDS) for files that can be described externally. These files are the physical, logical, display, printer, and intersystem communications functions, hereafter referred to as ICF files.

Contents

Chapter 1. - An Introduction to *PKZIP for iSeries*

PKZIP for iSeries is a high performance data compression product, containing two main programs; PKZIP and PKUNZIP. The PKZIP program is used to compress files into a ZIP format archive, while PKUNZIP is used to decompress data either compressed by PKZIP at an earlier time, or compressed by other file compression programs. Both programs are controlled by options that allow a variety of functions to be performed.

Multiple levels of processing control are available through the use of customized option modules, shared command lists, and individual job inputs. In addition to file selection, features such as compression levels and performance selections can be specified. Also, a 32-bit Cyclic Redundancy Check (CRC) is a standard feature used to guarantee data integrity.

A ZIP archive is platform-independent; therefore, data compressed (*ZIPPED*) on one platform, for example, UNIX, can be decompressed (*UNZIPPED*) on another platform, for example, OS/400 and MVS/ESA, by using a compatible version of PKUNZIP.

Data Compression

Because data compression techniques reduce file size, a compressed data file will use less storage space and can be transferred in a faster, more efficient manner. A file can be compressed (a ZIP candidate) to a compact size (ZIPPED file), and then to use the file again, it must be uncompressed or extracted to its original size (UNZIPPED file).

One easy data compression method eliminates repeating or redundant data by replacing it with representative information that will be used when restoring the data. An example of a data compression technique is the Run-Length Encoding method, which applies to redundant data where a repeating character (the run) is represented as a count or value (the length). The compressed form is the repeated character with its count.

Example: B 2 2 2 2 E H H H H H H H H H

Compressed: B *4 2 E *9 H

Note: The efficiency of this method is dependent upon the amount of redundancy in the data.

To perform a thorough compression operation, more advanced algorithms and enhanced techniques are required. ***PKZIP for iSeries*** uses just such methods to achieve maximum results.

ZIP Archives

PKZIP for iSeries is capable of storing compressed data into ZIP archives. There is no limit to the number of archives you may create.

ZIP archive capability:

- ZIP archive refers to any valid ZIP-format file created by a ***PKZIP®*** 4.x-compatible product.
- ZIP64 refers to ZIP archives that includes the ZIP64 format that can handle more than 65,534 files and files that exceed 4 Gig. (See Large Files Considerations).
- Each standard archive can store up to 65,534 files.
- Files that are over 4 Gig have to be archived with GIZP or by using the Large File Support.
- Each standard archive may contain up to 4 Gig of data. ZIP64 is required for larger archives.

For each file in the archive, the following information is stored with the compressed data:

- Filename.
- File directory date and time.
- File's initial CRC value (see Cyclic Redundancy Check).
- Method of compression used.
- ***PKZIP for iSeries*** version required for file extraction.
- File size, uncompressed.
- File size, compressed.

Some files may contain the following additional information:

- The version of ***PKZIP for iSeries*** that created the file.
- File attributes.
- Any comment about the file.
- Any comment about the archive.
- Platform specific attributes (see Cross Platform Compatibility).
- If encrypted and what method of encryption.

Cyclic Redundancy Check

Cyclic Redundancy Check (CRC) is a method used to verify the integrity of a data file after it is restored from a ZIP archive.

Before a file is compressed, a ***PKZIP for iSeries*** algorithm computes a 32 bit hexadecimal value for its data. The CRC value is stored in a file that is within the ZIP archive. When the data in the file is extracted, ***PKZIP for iSeries*** processes it again by the same algorithm to produce a second CRC value. Once the file is processed, the original CRC value is compared to the new CRC value to ensure that they match.

Note: If the data is the same as its previous state, the same CRC value will be produced. When the two CRC values are compared, and should the extracted value not match the stored, initial value, the integrity of the file is in question and ***PKZIP for iSeries*** reports the results. In this case, it is possible the data was corrupted within the ZIP Archive.

Data Encryption

PKZIP for OS/400™ Version 5.5 encrypts data for security control and provide a password lockout for extracting data. Varying security levels are available with 96, 128, 192, and 256 bit encryption, with varying encryption algorithms. As of July 1, 2001 the US Government mandated through the Gramm-Leach-Bliley Act that customer information must be encrypted to keep it confidential. (See Section Advanced Encryption Standard (AES) for more information to help organizations meet this requirement for advanced encryption). ***PKZIP for OS/400™*** Version 5.5 first implemented the Advanced Encryption Standard.

Decryption of encrypted information requires a key. ***PKZIP for iSeries*** uses a multi-layer key generation process (based on a user-specified password of up to 200 characters) that creates a unique internal key for each file being processed. In addition, the same password will result in a different system generated key for each file.

PKZIP for iSeries implements the use of Cipher Block Chaining (CBC) to further enhance industry standard encryption algorithms. This feature ensures that each block of data is uniquely modified, further protecting the data from fraudulent access.

The following matrix provides information for compression compatibility of encrypted data and general encryption options available on the operating systems supported by PKZIP.

More specifically, the 96 bit password encryption is compatible and can be ported to and from any of the platforms list below. Advanced password encryption is also available in a cross platform environment that supports the AES (AES is Government requirement for password and data encryption as of May 26, 2002) requirement of 128, 192, and 256 bit password encryption.

SECURITY TYPES	Generic ZIP Utilities	PKZIP for Windows	PKZIP for iSeries	PKZIP for OS/390	PKZIP for VSE	PKZIP for zSeries (See Note 1 below)	PKZIP for UNIX (See Note 2 below)
96 bit Encryption	**X**	**X**	**X**	**X**	**X**	**X**	**X**
This is advanced encryption; that is compatible with Windows.							
PKWARE Certificate Based Encryption		**X**					**X**
This is advanced encryption; that is compatible with multiple platforms.							
128, 192, & 256 bit AES Password Encryption		**X**	**X**	**X**	**X**	**X**	**X**

Note 1: *PKZIP for zSeries*™ Version 5.5 supports the following operating systems: MVS/ESA 4.2 and above, OS/390, and z/OS.

Note 2: PKZIP supports advanced password encryption for the following types of UNIX: Sun Solaris 2.6 and above, HP-UX 10.20 and above, IBM AIX 4.3 and above, Intel Linux based on 2.4 kernel.

PKZIP for iSeries uses AES which is the official US Government standard for encryption. The AES algorithm was approved as the Federal Information Processing Standard by the Commerce Department on May 26th, 2002.

The Rijndael (the name combination of the two researchers who developed Rijndael, Dr. Joan Daeman and Dr. Vincent Rijmen) algorithm uses a combination of advanced security, performance, efficiency, ease of implementation, and flexibility to make it an appropriate standard of advanced encryption for the AES.

Rijndael performs consistently in both hardware and software and in cross platform environments regardless of its use in feedback or non-feedback modes. Rijndael's key setup time is very good, and its key agility is excellent. Memory requirements are very low, making it the first choice for restricted-space environments, in which it also demonstrates high performance. Power and timing attacks are easily defended against due to Rijndael's operations.

Note that the AES was intentionally developed to replace DES.

AES Key Sizes

Currently, AES has three key sizes. They are: 128, 192, and 256 bits. Key sizes are depicted in the following decimal terms:

3.4×10^{38} possible 128-bit keys;
6.2×10^{57} possible 192-bit keys; and
1.1×10^{77} possible 256-bit keys.

In comparison, DES keys are only 56 bits, which means there are approximately 7.2×10^{16} possible DES keys. Therefore, there they are on the order of 10^{21} times more AES 128-bit keys than DES 56-bit keys.

Could "DES Cracker" like hardware break an AES key?

Specialized "DES Cracker" machines were built in the late 1990's that could recover a DES key after only a few hours. By trying possible key values, the hardware could determine which key was used to encrypt a message.

To illustrate the higher level of security that AES provides versus DES, if a machine that could recover a DES key in a second, such as, try 2^{55} keys per second, then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key. A further perspective is that the universe is believed to be less than 20 billion years old, thus making the case that cracking a 128-bit AES key is nearly impossible with today's technology.

Monitoring Algorithm Security

The National Institute of Standards and Technology (NIST) continues to follow developments in the cryptanalysis of Rijndael. The AES is formally reevaluated every five years. Plans for maintenance activities for the standard will be developed in the future, with full consideration of all circumstances. When an issue arises that requires immediate attention, NIST will act expeditiously and consider all available alternatives.

What is the Life Expectancy of AES?

No one can be certain of how long the AES will remain secure. However, NIST's DES was the U.S. Government standard for almost twenty years before it was "cracked" by a massive parallel network computer attacks and special-purpose "DES-cracking" hardware. The AES supports significantly larger key sizes than that of DES. Barring any attacks against AES that are faster than key exhaustion, and the advent of future advances in technology, AES could remain secure well beyond twenty years.

Large Files Considerations

Large File Support Summary

The Large File Support feature known as ZIP64 throughout this manual, was added to ***PKZIP for iSeries™*** in release 5.6. This separately licensed feature of ***PKZIP for iSeries™*** provides several enhancements relating to capacity, size and performance. Some of the key features include:

- Processing support (ZIP and UNZIP) for Archives enabled with the standard ZIP64 formats from other platforms.
- An increased ZIP Archive file capacity is raised from 65,534 to the theoretical limit of 4,294,967,295 files.

- An increased user file size handler, raised from 4 Gigabytes minus 1 byte (32 bit binary counter) to a theoretical limit of 9 Exabytes (64 bit binary counter).
- An increased support for ZIP Archive sizes exceeding 4 Gigabytes (same as user file size limit).

The theoretical numbers above are **only** given as theoretical points, where in practice there are reasonable limitations due to the availability of resources along with processing tolerances.

Note: 4 GB or Gigabyte is equal to 4,294,967,295 bytes.
9 EB or Exabyte is equal to 9,223,372,036,854,775,807 bytes.

Large File Support File Capacities

- The original .ZIP file format has faithfully met the needs of computer users since it was introduced by PKWARE in 1989. As computer technology has advanced over time, storage capacities have increased dramatically. These increases make the numbers and sizes of files that seemed unimaginable 10 years ago a reality today. To extend the utility of the .ZIP file format to meet these changing system needs, PKWARE extended the .ZIP file format to support more than 65,535 files per archive and archive sizes greater than 4 Gigabytes (GB). This is know as the ZIP64 format.
- The specification for the .ZIP file format has been publicly available and distributed by PKWARE in a file called APPNOTE.TXT. This file documents the internal data structures and layout that define a .ZIP archive. The extensions introduced by PKWARE fully supports all the features of your existing archives and newer versions of PKZIP that support these new extensions will continue to read all of your current archives. Prior to the ***PKZIP for iSeries™*** 5.6 release, versions of PKZIP on the OS/400 were limited to storing no more than 65,534 files in a .ZIP archive.
- Another limitation that existed prior to the 5.6 version of the ***PKZIP for iSeries™*** was that a single .ZIP archive or files in archive could not be larger than 4 GB (4,294,967,295 bytes). The extended ZIP64 file format specification available with PKZIP 5.6 supports creating .ZIP archives containing over 4 billion files and with sizes larger than 9 quintillion bytes. These are only theoretical limits and most iSeries systems and other computer systems in common use today do not have enough storage capacity, CPU or available memory to create and store ZIP64 archives approaching these limits.
- The practical limits imposed by a typical iSeries in use today and configured with various memory sizes will support compressing up to approximately 265,000 files. Compressing this number of files can take a long time, not only just for compression but to manage the directories and properties of each of these files.
- Your available system resources (processor speed, DADS, Memory, and other processing) limits the performance you can expect from PKZIP when processing large numbers of files or large archives. If you are compressing large numbers of files on a iSeries with insufficient memory or other resources you can expect slow processing.
- When compressing large files, it is a good idea to have your archives set up to be stored in the IFS rather than in a library/file. The overhead is much less when storing the archive in the IFS. It is even more important when updating or adding to an archive where the temporary archive will also be processed in the IFS
- Versions of ***PKZIP for iSeries™*** prior to 5.6 will not recognize these new features and will be unable to view or extract any files in your archives that are dependent on these ZIP64 features. Also, any ZIP compatible programs you may be using from other companies will not be able to access all of the contents of your large archives. They may report that an archive is too large, or they may incorrectly report that the archive has errors. To ensure access to data in your large archives, always use genuine PKZIP from PKWARE.

Cross Platform Compatibility

Cross platform compatibility provides ***PKZIP for iSeries*** its ability to allow data to move between different computer operating environments. PKZIP was intentionally designed for cross platform use. Regardless of platform, ***PKZIP for iSeries*** archives are 100% cross platform compatible with all other ZIP utilities like , ***PKZIP for zSeries™***, ***PKZIP for VSE™***, ***PKZIP for UNIX™***, ***PKZIP for LINUX™***, ***PKZIP for DOS™***, and ***PKZIP for Windows™*** to name a few. Because ***PKZIP for iSeries*** automatically converts the data between EBCDIC and ASCII, files prepared on the host are perfectly readable on any PC or UNIX system. The internal format of a ZIP archive is identical no matter which platform compressed the files that the archive contains. If you want to transfer data across platforms using any other version of PKZIP or other ZIP utility, you should always run a test to verify the cross platform compatibility.

PKZIP for iSeries uses the same ZIP file archive format used by other ***PKZIP®*** 2.x compatible products, independent of the platform on which it is running. ***PKZIP for iSeries*** archives are not platform dependent allowing greater flexibility in file usage. Data can be zipped on one platform, for example UNIX, and unzipped onto another platform, such as, OS/400. To do this, ***PKZIP for iSeries*** converts the data structure into the ***PKZIP®*** format and saves the appropriate file information in the ZIP Archival directory entries.

If you want to transfer data across platforms using any other ”Zip compatible” product, the user should check with the supplier first to confirm that the versions of PKZIP that are compatible.

For more information regarding Data Formats, see Chapter 5. - Text or Binary for a discussion regarding special considerations when transferring files between different platform types.

Release Summary

New Features 5.6

- Support for ZIP64 Archive formats to support Large File System Feature.
- Control over Extra Data to reduce total archive size. (New Options in PKZIP Parameter EXTRAFLD.)
- Warning when updating Digitally Signed archives.
- Ability to ignore case sensitivity when selecting files in the IFS. (New Option *IFS2 with parameter TYPFL2ZP).
- Ability to use CVTNAME when selecting spool files to change name in archive.
- New warning message AQZ0024 to monitoring when no files are selected.
- Ability to create and maintain self extracting archives. (New PKZIP Parameter SELFXTRACT.)
- Ability to insert a path to the file names in archive. (New PKZIP parameter ISRTPATH.)
- New translating tables for code pages included in file PKZTABLES.
- Ability to handle extremely large Spool File conversions to PDF or TEXT.
- Ability to convert a spool file to an ASCII Text document with ANSI control characters. The first character of every record contains an American National Standards Institute (ANSI) forms control character. (New option *TEXTFC in parameter SFTARGET.)
- More Information with detail view of Spool File in an archive.

- The ARCHIVE and FILE paths and file names has been expanded to 256 characters.
- The ARCHIVE will now accept imbedded spaces in the path and file name for archives in the IFS..
- A new PKZIP global setting to disallow wildcard selection is provided
- PKZIP will now post a standard description to archive document file descriptions for archives created in the /QDLS.

PKZIP for iSeries™ Restrictions

Due to various iSeries processing characteristics, the following restrictions should be carefully reviewed to determine the best way to proceed when using ***PKZIP for iSeries***:

- ***PKZIP for iSeries*** in the QSYS file system will only work with objects that have an object type of *FILE and an attribute of PF-DTA, PF-SRC, and SAVF. To process other objects such as *PGM, *CMD, etc., use the SAVF method (see Use of SAVF Method).
- ***PKZIP for iSeries*** in the Integrated File System (IFS) will only work with stream files (*STRM) and directories (*DIR).
- Special Database functionality, such as Triggers, File Constraints, Alternate Collating Sequence, and Logical files are not stored in an archive. To maintain this functionality, use the SAVF method (see Use of SAVF Method).
- Special Database fields for large objects (LOB) are not supported. These fields include : Character Large OBjects (CLOBs), Double-Byte Character Large OBjects (DBCLOBs), and Binary Large OBjects (BLOBs). In case where the database contains one of these types of fields, you will have to use the SAVF Method.

Chapter 2. - Getting Started with *PKZIP for iSeries*

PKZIP for iSeries is a broad, flexible product on the iSeries, and AS/400 platforms, allowing for compression and decompression of files. It is fully compliant with other ***PKZIP®*** compatible compression products running on other operating systems.

Because the ***PKZIP®*** standard for text data storage is ASCII, ***PKZIP for iSeries*** facilitates conversion between the ASCII and EBCDIC character sets. Therefore, compressed text files can be transferred between IBM mainframe environments and systems using the ASCII character sets, including UNIX, DOS, ***PKZIP for iSeries***, ***PKZIP for zSeries***, and ***PKZIP for VSE™***.

In addition to ***PKZIP®***-format archive support, ***PKZIP for iSeries*** can also produce and manipulate (GNU) GZIP-format archives. Additional information on this subject can be found in Chapter 11. - Processing with GZIP.

Basic Features of *PKZIP for iSeries*

PKZIP for iSeries is generally compatible with ***PKZIP®*** 2.x, and as such, has the following features:

- Compliance with compression programs on other platforms, including Windows, LINUX, UNIX, DOS, ***PKZIP for iSeries***, ***PKZIP for zSeries***, and ***PKZIP for VSE™***.
- User-selected compression ratios.
- Storage capability of 65,535 files within one ZIP Archive.
- Compression of files of up to 4 gigabytes.
- A maximum ZIP Archive size of 4 gigabytes.
- Data integrity assurance using 32-bit CRC error detection.
- Translation of data to a system-independent format, thus providing easy file transfers within a mixed or varied file environment.

Extended Features of *PKZIP for iSeries*

PKZIP for iSeries also offers a series of extended features such as creation of GZIP archive, Spool Files Support, Large File Support (files greater than 4 Gig and files in archive exceeding 65,535), Advanced Encryption and Self extracting archives. All licensable features can be found in Licensed Product Features in Chapter 3. - *PKZIP for iSeries* Installation.

Invoking *PKZIP for iSeries* Services

There are three commands used for ***PKZIP®*** within the OS/400 operating environments. The commands are:

- PKZIP - Compression Utility.
- PKUNZIP - Archive Extraction Utility.
- PKZSPOOL - Compress Utility for Spool Files.

All commands can be invoked interactively, submitted for a batch run, or used anywhere that an iSeries command can be issued.

Help panels for each command can be activated by using the F1 (help) key.

PKZIP for iSeries Differences with other Platforms

This section covers the differences between ***PKZIP for iSeries*** and other versions and operating systems (platforms). Most of the differences are due to the QSYS library file type system and the iSeries object-oriented base.

Attributes (non-extended)	Various MS/DOS options support the selection of files by file attributes such as hidden, read-only, and system. These attributes are not meaningful on the OS/400 file system.
ANSI comments	Because OS/400 does not support ANSI control codes, related options are not supported. When unzipping from an archive, the archive comment will be displayed, but ANSI control codes in this comment will not be masked out. This could cause attribute changes on the iSeries display.
Archive file date controls *PKZIP®*	DOS options control whether the ZIP file date is updated or retained when altering the archive. Because the last used date on OS/400 is not under program control or alterable by a command, these options are not supported.
Archive Comments *PKZIP®*	DOS options allow editing of comments for individual files in an archive. This version supports editing of a file's text description, but is not recommended for batch running, or for a large number of files due to the interactive message responses required.
File naming differences	The files used in the QSYS library file system have their own naming style. Each file associated with a library file and members would be depicted as library/file(member). Usually, all file names are stored as open system file names with directories, ending with a file name. For a detailed description and techniques see Chapter 6. - OS/400 File Processing.
HELP	***PKZIP for DOS™*** has options to display a list of commands. Because ***PKZIP for iSeries*** uses iSeries commands, the help system is built for each command and is activated by PF1 on each parameter.
Mixed Case Filenames	When using the IFS (Integrated File System), the file names are case sensitive and act like other file systems (UNIX, DOS, Windows, etc.). When using the QSYS Library file system, the file names are always in UPPER CASE. Occasionally, when trying to update and archive (or select from an archive), you may encounter a case sensitive search. Use PKUNZIP view to get the exact name stored. This would be appropriate when doing a PKZIP TYPE (*DELETE) where the selection file would need to match.

Use of SAVF Method

At this time, only physical files with attributes of PF-DTA, PF-SRC, and SAVF in the QSYS file system, Stream files in the IFS, and Spool Files can be processed by ***PKZIP for iSeries***. Also, some special Database functionality such as Triggers, File Constraints, Alternate Collating Sequence, and Large Object Fields, are not stored in the Archives.

To overcome some of the restrictions listed above, ***PKZIP for iSeries*** can compress and decompress SAVF. The objects to be compressed are saved to a SAVF using SAVOBJ or SAVLIB. The SAVF is then compressed to an archive using PKZIP. To restore the data, first use PKUNZIP to re-create the SAVF, then use RSTOBJ or RSTLIB to restore objects from within the decompressed SAVF. SAVF are binary and only pertain to OS/400.

Chapter 3. - *PKZIP for iSeries* Installation

This chapter describes the process of receiving ***PKZIP for iSeries***, either by tape, CD-ROM or by FTP, the process of setting the environment (optional) and the process of installing the license. These processes consists of building the ***PKZIP for iSeries*** version library, adding the library to your library list, optionally setting environmental pointers and running the Install License command with an Authorized License.

Even though the default library for ***PKZIP for iSeries*** Version 5.6 patch level 0 is PKZ560510 and is referenced in this manual, the library name for the product can be named otherwise to suit your environmental needs. The ***PKZIP for iSeries*** product is distributed with a predefined library of **PKZ560*vrm*** (where ***vrm*** is the minimum OS/400 operation system release supported). V5R1M0 would be 510. If you would like to use another library name or would like to run ***PKZIP for iSeries*** without it being in the library list, review Operating Environment discussion later in this chapter.

It is recommended that you use a generic name for the library that will be used for production (such as PKZIP or PKZIPPROD). This will allow new updates to be implemented without changing a lot of CL programs or processes. To use another library name, review the procedures described later in this chapter (Operating Environment) on how to use the CALL PKZSETLIB program to change the standard library name or run without the library in your library list.

Unloading *PKZIP for iSeries* from Tape

The ***PKZIP for iSeries*** installation tape contains a SAVF file with the product library for the OS/400. The file **PKZ560*vrm*** library contains ***PKZIP for iSeries*** Version 5.6 patch level 0 with the OS/400 version specified as *vrm* (OS/400 version, Target Release, and Modification for the product build, for example, 510 for V5R1M0).

To unload the required library for ***PKZIP for iSeries***, load the tape and issue the following command:

RSTLIB SAVLIB(**PKZ560*vrm)*** DEV***(tapnn***)

Where ***vrm*** is the version, release, and modification level and ***tapnn*** is the appropriate tape device name.

At this point the PKZIP library exist on you system and you can proceed to Restoring PKZIP for iSeries Product Library from a Save File.

Unloading *PKZIP for iSeries* from a CD-ROM

The ***PKZIP for iSeries*** installation from a CD-ROM can be performed using either of the following two methods. One method is to use the LODRUN command and the second method would be to restore the library manually. First load the ***PKZIP for iSeries*** CD into the optical unit and note the device name (usually OPT01). It is **METHOD A. LODRUN command:**

At a command line type ➔LODRUN DEV(*OPT) or ➔ LODRUN DEV(*optical device*).LODRUN will first display the following screen showing the default library with the version of ***PKZIP for iSeries*** that will be installed from the CD. The display will wait for a valid Library and the pressing of the Enter Key to install ***PKZIP for iSeries***. If the F3 or F12 key is pressed the install will end without installing the product.

LODRUN Screen Example:

```
  PKZIP for iSeries(tm)    Version  V5.6.0

  Install PKZIP for iSeries(tm) to Library  PKZ560510
      (Note:  Library Must Not Exist)




  Press Enter to Continue
  F3-Exit           F12-Cancel
Error message Area-----------------------------------------------------
```

Next the install process will restore the ***PKZIP for iSeries*** product to the library requested, and add the new library to the library list. Then using the program PKZIP program PKZSETLIB, all settings for commands, Data Area and Help files will be resolved to the new library.

At this point, ***PKZIP for iSeries*** is in your library list is ready to load the of the keys (DEMO or registered) using the INSTPKLIC command as describe later in this chapter. **METHOD B. RSTLIB command:**

The ***PKZIP for iSeries*** installation CD-ROM contains a SAVF file with the product library for the OS/400. The **PKZ560*vrm*** library contains ***PKZIP for iSeries*** Version 5.6 patch level 0 with the OS/400 version specified as ***vrm*** (OS/400 version, Target Release, and Modification for the product build, for example, 510for V5R1M0).

To unload the required library for ***PKZIP for iSeries***, load the CD-ROM and issue the following command:

RSTLIB SAVLIB(**PKZ560*vrm)*** DEV(***optnn***) OPTFILE(*pkzip*)

Where ***vrm*** is the version, release, and modification level and ***optnn*** is the correct optical device name to use.

Using FTP to iSeries to transfer a PKZIP Save File

1. After downloading the zip file to your PC extract the save file from the zip file. The save file is named PKZ556510.SAV. Be sure to note the path to this file, as it will be required in your FTP session (defaults to C:\PKZIP\PKAS4R\PKZ552430.SAV).
2. Start an iSeries workstation type session, and then sign on with sufficient authority to perform the commands used below.
3. Create a save file on the iSeries - CRTSAVF YourLIB/PKZIP560. The TCP/IP network server must be running (or start the TCP/IP network server with STRTCP).
4. Start a PC-based FTP session with your normal FTP procedures and send the PC file to the iSeries SAVF you created in Binary mode, or run the following:
   - Type "FTP" at the command prompt.
   - Open the iSeries IP Address and sign on (ftp> open 208.222.150.11 as an example).
   - Next, type "BIN" at the command line (this will transfer the file as a Binary object which is required).
   - Now type "PUT" and then the local file name (i.e. "PUT C:\PKZIP\PKAS4R\PKZ5605100.SAV")

- Finally, you type your remote file name (the destination): (YourLIB/PKZIP552) and the FTP of the file should start.

Restoring PKZIP for iSeries Product Library from a Save File

At this time you should have the save built on your system. Now you can restore the library from the save file.

➔RSTLIB SAVLIB(PKZ560510) DEV(*SAVF) SAVF(YourLIB/PKZIP560) RSTLIB(your_pkzip_lib).

Where "your_pkzip_lib" - is the name you want to call the restored PKZIP library.

At this point the ***PKZIP for iSeries*** product library should exist on iSeries and you proceed to the Installation procedures below.

If you want, you could now delete the save file created earlier with:

➔ DLTF YourLIB/PKZIP560

Installation Procedures

The ***PKZIP for iSeries*** product library should now exist on the iSeries. All objects including the library are owned by QPGMR. The objects' owner can be change to any valid owner with the CHGOBJOWN command. The ***PKZIP for iSeries*** library should now be added to the library list (ADDLIBLE your_pkzip_lib).

If you keep the PKZIP library name at PKZ560510, then you do not need to take this additional step. If you change the name from PKZ560510 to something else, then you will need to run a program to setup your PKZIP Environment. In this case you will type on a command line, "CALL PKZSETLIB your_pkzip_lib", where "your_pkzip_lib" is the name of the PKZIP library you restored. This will link the objects to your PKZIP library name. For more information see "How to Change the Standard PKZIP Library Name" later in this chapter.

Licensing Requirements

PKZIP for iSeries is a licensed product. Without proper licensing, the product can only be used to view archives. Product features and license types can be licensed separately as the user's needs dictate. The license key will contain all of the elements necessary to validate a customer's use of ***PKZIP for iSeries***.

The licensing process is comprised of several key elements that are described in the following sections.

Trial Period

You will need to contact your reseller or PKWARE, Inc Sales at 937.847.2374, or email sales at pksales@pkware.com to initialize your version of PKZIP for iSeries for the 15-day DEMO. If you do not work in the USA, please refer to the GLOBAL CONTACTS.TXT file to contact a dealer in your region.

Release Licensing

Each release of PKZIP for iSeries requires that a new license key be obtained from Customer Service and that a new license record be generated. The new release will fail with AQZ9077 "License Keys have invalid version setting" if the License file is used from a previous release.

Reporting Environment

To report on the status of a license at your location, you can run the Environment "WHATOSV" program by doing a program call:➔CALL WHATOSV. It will provide a report similar to:

```
PKWARE WHATOSV Current Operating Environment  Thu May 15 12:05:49 2003

PKZIP for iSeries(tm) Version  5.6.0  with build date 2003/05/14
       Current PKZIP Library is PKZ560510
IBM iSeries Model 9406, Type  270-23E7
Serial Number <010-7X8WT  >, PRC Group < P10>,  OS is at V5R2M0.

Press ENTER to end terminal session.
```

The output of this report is what you will need to send to your reseller or PKWARE Sales Representative to obtain a DEMO code.

Note: The PKZIP Library must be added to the library list prior to running this program.

Please have the output of this report handy when speaking with your reseller or account rep. You will be expected to supply the following additional information:

* Company Name
* Company Contact
* Phone number
* Contact Email

Updating License Files

Installing the PKZIP license activation keys, will be added the licensing information into a source file member (one is provide with distribution library call PKZLICIN) and then running the install license program to activate.

Trial activation is accomplished by first editing the member PKWARELIC and adding the company customer record and keys supplied by PKWARE, Inc. One way of editing the member would to use the following command with the correct library:

➔EDTF FILE(PKZ560510/PKZLICIN) MBR(PKWARELIC)

or

➔STRSEU SRCFILE(PKZ560510/PKZLICIN) SRCMBR(PKWARELIC)

Remember since this a source file member and you use the EDTF command that the data will start in column 13, because the source sequence number and date stamp is in the true columns 1 thru 12.

For Example use of command **EDTF FILE(PKZ560510/PKZLICIN) MBR(PKWARELIC)** :

```
 Edit File: PKZ560510/PKZLICIN(PKWARELIC)
 Record :       1   of       3 by   8           Column :   13     92 by  74
 Control :

CMD ..+....2....+....3....+....4....+....5....+....6....+....7....+....8....+.
     ************Beginning of data**************
    *LICENSED BY PKWARE of OHIO, Inc   06/03/03 Tait Hamiel
    55 A4CMD1NR 000014581 PKWARE Internal Demo Customer
    99 CMDOAXB1 20030703 0107X8WTP10
     ************End of Data********************



 F2=Save   F3=Save/Exit   F12=Exit   F15=Services   F16=Repeat find
```

Notice in this case the columns on the ruler shows column 13 for the first column of the license data.

An Example use of command STRSEU SRCFILE(PKZ560510/PKZLICIN) SRCMBR(PKWARELIC) :

```
 Columns . . . :    1  71           Edit                   PKZ560510/PKZLICIN
 SEU==>                                                          PKWARELIC
 FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
       ************** Beginning of data *************************************
0001.00 *LICENSED BY PKWARE of OHIO, Inc   06/03/03 Tait Hamiel
0002.00 55 A4CMD1NR 000014581 PKWARE Internal Demo Customer
0003.00 99 CMDOAXB1 20030703 0107X8WTP10
       ****************** End of data ***************************************



 F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
 F16=Repeat find       F17=Repeat change          F24=More keys
```

Once you have typed or copied the license information provided by PKWARE, you will need to save these changes and exit the edited member. Next, run the install program using the following command:

➔INSTPKLIC INFILE(*LIBL/PKZLICIN) INMBR(PKWARELIC) or prompt F4

```
                      Install PKZIP for iSeries License (INSTPKLIC)

 Type choices, press Enter.

 Type . . . . . .   *INSTALL      *INSTALL, *VIEW
 Input Control File . . . . . .   PKZLICIN      Name, PKZLICIN
   Library name . . . . . .     *LIBL       Name, *LIBL
 Control Member . . . . . .   pkwarelic         Name, *FIRST




                                                                      Bottom
 F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display
 F24=More keys
```

By executing the INSTPKLIC command, the LICENSE dataset will be updated and a report will be produced that will reflect the state of ***PKZIP for iSeries*** at your location.

```
PKZIP for iSeries(tm) Compression Utility Version   5.6.0,  2003/06/09
Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
Rec -  1 *LICENSED BY PKWARE of OHIO, Inc   06/03/03 Tait Hamiel
Rec -  2 55 A4CMD1NR 000014581 PKWARE Internal Demo Customer
Rec -  3 99 CMDOAXB1 20030703 0107X8WTP10
Compression           - Evaluation set to expire in 23 days on 20030703
Decompression         - Evaluation set to expire in 23 days on 20030703
Database File Handlers- Evaluation set to expire in 23 days on 20030703
IFS File Handlers     - Evaluation set to expire in 23 days on 20030703
GZIP                  - Evaluation set to expire in 23 days on 20030703
Advanced Encryption   - Evaluation set to expire in 23 days on 20030703
Spool Files           - Evaluation set to expire in 23 days on 20030703
Large Files           - Evaluation set to expire in 23 days on 20030703
Self Extracting       - Evaluation set to expire in 23 days on 20030703
License File PKZ560510/PKZLIC(PKZLIC) Updated successfully
Press ENTER to end terminal session.                                        .
```

Licensed Product Features

The license key will be comprised of codes to reflect the product features selected by the customer.

The following list contains the product features available:

Feature Code	Type	Description
01	**Compression**	This licensable module allows for the compression of data into a ZIP file. This license is required to read the input data and write out the archive or ZIP file.
02	**Decompression**	This licensable module allows for the decompression of data from a ZIP file or an archive. This license is required to read the ZIP archive and to write the OS/400 datasets during the extraction routine.
03	**Database File Handler**	This licensable module allows for PKZIP to read and write Physical Data Files, Physical Source Files, and SAVF in the QSYS Library Files system. When licensed with Compression and Decompression, it allows theses files and members to be compressed and decompressed. It also allows ZIP archives and "List Files" to be stored and processed in a Physical Data Files format in the QSYS Library Files system.
04	**IFS File Handler**	This licensable module allows PKZIP to read and write files to Stream Files in the Integrated File System (IFS). When licensed with Compression and Decompression, it allows Stream Files to be compressed and decompressed. It also allows ZIP archives and "List Files" to be stored and processed in a stream file format in the IFS.
09	**GZIP Support**	This licensable module allows a user to read and write GZIP compatible files. GZIP files created using PKZIP for iSeries can be extracted with any compatible GZIP utility on any other platform.
12	**Advanced Encryption**	This licensable module allows a user to compress and extract files with Advanced Encryption methods.
13	**Spool File Handler**	This licensable module allows a user to compress and extract Spool files. The module also allows the Spool File to be converted another format, such as a PDF or Text format.
14	**Large File Support**	This licensable feature allows the use of the ZIP64 archive format to support files over 4 Gigs and to have more than 65,535 files in an archive.
15	**Self Extract Support**	This licensable feature allows the creation and maintenance of self extracting archives.

Record Layouts

The record layouts for Control file records (portions of which your reseller of PKWARE, Inc. will provide) are described below for each type. Remember the columns are relative to column 1 in a source file and if you use EDTF then column 1 is now column 13.

Comment Control Card

Comment cards are free format and begin with an * in column 1.

Customer Control Card

Customer records always begins with a “55”.

	Control Code	Check Characters	Customer Number	Customer Name
Format	55	cccccccc	nnnnnnnnn	xxx . . . xxx
Columns	1-2	4-11	13-21	23-74
Length	2	8	9	50

55 Feature Control Code (always a 55).

cccccccc Check Character (supplied by PKWARE, Inc.).

nnnnnnnnn Customer Number (supplied by PKWARE, Inc.).

xxx . . . xxx Customer Name (supplied by customer - must be alphanumeric (AA-99).

Feature Control Card

	Feature Code	Check Characters	Expiration Date	Hardware Information
Format	ff	cccc	yyyymmdd	sssssssttt
Columns	1-2	4-11	13-20	22-33
Length	2	8	8	12

ff Feature Control Code (provided by PKWARE, Inc.).

cccccccc Check Character (provided by PKWARE, Inc.).

yyyy year (2001).

mm month (01-12).

dd day (01-31).

sssssss CPU serial # (12345ABC).

tttt CPU Processing Group (P10).

By executing this command, the LICENSE dataset will be updated and a report will be produced that will reflect the state of ***PKZIP for iSeries*** at your location.

Install Demo

```
PKZIP for iSeries(tm) Compression Utility Version   5.6.0,  2003/06/09
Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
Rec -  1 *LICENSED BY PKWARE of OHIO, Inc   06/03/03 Tait Hamiel
Rec -  2 55 A4CMD1NR 000014581 PKWARE Internal Demo Customer
Rec -  3 99 CMDOAXB1 20030703 0107X8WTP10
Compression           - Evaluation set to expire in 23 days on 20030703
Decompression         - Evaluation set to expire in 23 days on 20030703
Database File Handlers- Evaluation set to expire in 23 days on 20030703
IFS File Handlers     - Evaluation set to expire in 23 days on 20030703
GZIP                  - Evaluation set to expire in 23 days on 20030703
Advanced Encryption   - Evaluation set to expire in 23 days on 20030703
Spool Files           - Evaluation set to expire in 23 days on 20030703
Large Files           - Evaluation set to expire in 23 days on 20030703
Self Extracting       - Evaluation set to expire in 23 days on 20030703
License File PKZ560510/PKZLIC(PKZLIC) Updated successfully
Press ENTER to end terminal session.
```

Install Basic

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/07/12
Copyright.  2003  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
Rec -  1 *LICENSED BY PKWARE    07/13/01 WSS
Rec -  2 55 B7LJJ3A0 110531837                        My Company Name, Dayton OH
Rec -  3 *           yyyymmdd sssssssttt
Rec -  4 77 lLTJ1233 20020801 0107X8WTP10
License File PKZ560510/PKZLIC(PKZLIC) Updated successfully
```

Install Single

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/07/12
Copyright.  2003  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
Rec -  1 *LICENSED BY PKWARE    07/13/01 WSS
Rec -  2 55 X1LT83K1 110531837                        My Company Name, Dayton OH
Rec -  3 *           yyyymmdd sssssstttii
Rec -  4 *                    sssssssttt
Rec -  5 01 LLTB7233 20030801 0107X8WTP10
Rec -  6 02 ELTH823F 20030801 0107X8WTP10
Rec -  7 03 RLTB723U 20030801 0107X8WTP10
Rec -  8 04 MLTH823B 20030801 0107X8WTP10
Rec -  9 09 6LTB723P 20030801 0107X8WTP10
License File PKZ560510/PKZLIC(PKZLIC) Updated successfully
```

Reporting

To report on the status of a license at your location, you can run the license program with the following command: INSTPKLIC TYPE(*VIEW).

EXAMPLES from above:

View Demo

```
PKZIP for iSeries(tm) Compression Utility Version   5.6.0,  2003/06/09
Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
*****************************************
A License Report requested on 0107X8WT  from CPU Serial#
5.6 Product Licensed to Customer # 000014581 -PKWARE Internal Demo Customer
*****************************************
Compression           -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Decompression         -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
GZIP                  -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
IFS File Handlers     -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Database File Handlers-DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Advanced Encryption   -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Spool Files           -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Large Files           -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Self Extracting       -DEMO with 23 Days remaining (07/03/2003)
      Contact PKWARE of Ohio, Inc. for Licensing
*****************************************
Press ENTER to end terminal session.
```

View Single

```
PKZIP for iSeries(tm) Compression Utility Version   5.6.0,  2003/06/09
Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
*****************************************
A License Report requested on 0107X8WT  from CPU Serial#
5.6 Product Licensed to Customer # 000003079 -Key Testers Inc.
*****************************************
Compression           :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Decompression         :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
GZIP                  :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
IFS File Handlers     :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Database File Handlers:Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Advanced Encryption   :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Spool Files           :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Large Files           :Licensed -Expires 00/00/0000 for processors:
*****************************************
```

```
Self Extracting       :Licensed -Expires 00/00/0000 for processors:
*****************************************
Press ENTER to end terminal session.
```

View Single with Exception

```
PKZIP for iSeries(tm) Compression Utility Version   5.6.0,  2003/06/09
Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Machine ID = 0107X8WT, Processor Group =  P10
*****************************************
A License Report requested on 0107X8WT  from CPU Serial#
5.6 Product Licensed to Customer # 000003079 -Key Testers Inc.
*****************************************
Compression           :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Decompression         :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
GZIP                  :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
IFS File Handlers     :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Database File Handlers:Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Advanced Encryption   :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Spool Files           :Licensed -Expires 02/28/2400 for processors:
  Serial# 0107X8WT    Processor Type P10
*****************************************
Large Files           :Licensed -Expires 00/00/0000 for processors:
*****************************************
Self Extracting       :Licensed -Expires 00/00/0000 for processors:
Serial# 0107X8WT is Not Licensed.  Use Of The Product will CEASE in 7 days (6/17/2003)
*****************************************
Press ENTER to end terminal session
```

Conditional Use

PKWARE, Inc recognizes that there may be periods where the licensing environment established by the customer is no longer valid. Circumstances such as disaster recovery processing or the installation or upgrade of new processors will affect the environment. To accommodate the customer, ***PKZIP for iSeries*** has a process that will allow a customer to continue using the product for a period of 5 days. During this time, error messages will be displayed on the console (as well as the printout) for each execution of ***PKZIP for iSeries***. At the end of the grace period, if the license keys are not updated, the product will no longer function in any environment other than to VIEW an archive. This 5 day grace period is designed in such a way that it will not cease to function on a weekend or the Monday following the 5 day grace period.

During this process you must contact your reseller or PKWARE, Inc. at 1-937-847-2687 to obtain licensing to allow use beyond the conditional period.

Operating Environment

PKZIP for iSeries is distributed in a standard library, such as PKZ560510 where the 560 is the version release of ***PKZIP for iSeries*** and 510 is the minimum OS/400 operation system release that it supports (such as V5R1M0). The ***PKZIP for iSeries*** Library contains Programs, Commands, Help Panels, Message File, License Files, Translation Source File, Command Source File, CL Source File (CVTNAME CL program and examples), and the ***PKZIP for iSeries*** Control Data Area necessary to run the product. As released, the distributed library PKZ560510 must be in the library list and the library name must be the same as distributed.

The Library name used by ***PKZIP for iSeries*** is determined by the Data Area PKZDTA5.

PKZDTA5 Data Area

The "PKZDTA5" data area is required to be in the library list in order run the programs and commands of ***PKZIP for iSeries***. The data area is 50 bytes in length with the following layout:

Bytes 1 thru 10 Library name for the product (required to run the product).

Bytes 11 thru 20 Release Levels for ***PKZIP for iSeries***.

Bytes 21 thru 40 Distribution and Generation Information only.

Bytes 41 thru 50 Global settings.

1. Byte 41 - Disallow wildcard Selection
    a. N = Wildcard selection is permitted (default).
    b. Y = Wilcard selection with PKZIP is not permitted.

An example of the "DSPDTAARA PKZDTA5" output might look like:

```
                              Display Data Area
Data area . . . . . . . :   PKZDTA5
  Library . . . . . . . :     PKZ560510
Type  . . . . . . . . . :   *CHAR
Length  . . . . . . . . :   50
Text  . . . . . . . . . :   PKZIP for iSeries Library-V5.6

           Value
Offset      *...+....1....+....2....+....3....+....4....+....5
    0      'PKZ560510 V5.6              NNOV5R1M0   N       '
```

The running of ***PKZIP for iSeries*** requires the data area to determine the library name to use for running the commands, help panels, and message file.

How to Change the Standard PKZIP Library Name

To assist the customer in changing the environment for running the ***PKZIP for iSeries*** product, the program "PKZSETLIB" has been included in the distribution library. PKZSETLIB will set the PKZDAT5 data area's Library name and will change the commands and help panels to recognize the library links.

An example of how use PKZSETLIB to change the standard library name to a user environment Library name is shown in the steps below:

1. Rename the ***PKZIP for iSeries*** standard library to the new library name:

===>RNMOBJ OBJ('PKZ560510) OBJTYPE(*LIB) NEWOBJ(newlib)

2. Verify that newlib is in library list with ADDLIBLE or EDTLIBL:

   ===>ADDLIBE newlib

3. Execute PKZSETLIB program to change the data area and change links:

   ===>CALL PKZSETLIB ('newlib')

4. Display the PKZDTA5 data area and verify that the Library has been changed.

   ===> DSPDTAARA PKZDTA5 (Positions 1 thru 10 should contain newlib)

5. Test the commands PKZIP or PKUNZIP to make sure they execute correctly.

At this point the "newlib" is the new ***PKZIP for iSeries*** library for this environment and will need to be placed in the library list to run.

Alternate Install from SAVF with Non-Standard Library Name

After the SAVF is ready to restore the library, a slight modification to the commands as shown in above will be required. In the RSTLIB command you will specify "newlib" in the parameter RSTLIB to restore the library with the new library name.

For example:

==>RSTLIB SAVLIB('PKZ560510) DEV(*SAVF) SAVF(mylib/PKZ560) RSTLIB(newlib)

To complete the installation, do the following:

Add the "newlib" to your library list with ADDLIBLE:

===>ADDLIBLE newlib

Execute the PKZSETLIB program to change data area and change links:

===>CALL PKZSETLIB ('newlib')

Display the PKZDTA5 data area and verify that the Library has been changed.

===> DSPDTAARA PKZDTA5 (Positions 1 thru 10 should contain newlib)

Test the commands PKZIP or PKUNZIP to make sure they execute correctly.

At this point the "newlib" is the new ***PKZIP for iSeries*** library for this environment and will need to be placed in the library list to run.

Running Without the PKZIP Library in the Library List

PKZIP for iSeries can run without the library in the library list as long as a data area as described above is in a library that is in the library list. One method is qualifying the command with the library name, such as:

===> my*library*/PKZIP ARCHIVE('myarchive/V5(test1)' FILES("testlib/*all')

The other method copies the commands to a library that exist in their Library List.

Note: The running of ***PKZIP for iSeries*** requires the data area to determine the library name to use for running the commands, help panels, and message files. The following are 5 steps to run ***PKZIP for iSeries*** without its distribution library in the library list.

Assume **'PKZ56x510'** is the ***PKZIP for iSeries*** library and the customers library MYlibrary is in the customer's standard library list. We will now add 3 new object to the library MYlibrary.

1. Create a new data area in the chosen library MYlibrary:

===>CRTDTAARA DTAARA(MYlibrary/PKZDTA5) TYPE(*CHAR) LEN(50) TEXT('PKZIP control data area')

2. Next set the data area value to the same values found in supplied data area in the PKZIP Library.

===>"DSPDTAARA 'PKZ56x510/PKZDTA5" output might look like:

```
                          Display Data Area
Data area . . . . . . . :   PKZDTA5
  Library . . . . . . . :     PKZ56x510
Type  . . . . . . . . . :   *CHAR
Length  . . . . . . . . :   50
Text  . . . . . . . . . :   PKZIP for iSeries Library-V5.6

           Value
Offset      *...+....1....+....2....+....3....+....4....+....5
    0      'PKZ56x510 V5.6                NNOV5R1M0 N          '
```

3. Now change the data area:

===>CHGDTAARA DTAARA(MYlibrary/PKZDTA5)
VALUE(''PKZ56x510 V5.6xxxV5R1M0')

Next we need to copy the two commands to the MYlibrary Library.

4. **===>CRTDUPOBJ OBJ(PKZIP) FROMLIB(PKZ56x510) OBJTYPE(*CMD) TOLIB(MYlibrary)**

5. **===>CRTDUPOBJ OBJ(PKUNZIP) FROMLIB(PKZ56x510) OBJTYPE(*CMD) TOLIB(MYlibrary)**

At this point, PKZIP will work for all users that have the library MYlibrary in their library list.

Note: When installing another ***PKZIP for iSeries*** Version the same process should be followed in order to pick up the
latest settings of the data area and commands.

Next ensure the ***PKZIP for iSeries*** library is NOT in your library list and run a few sample test for PKZIP and PKUNZIP to make sure the product works.

Chapter 4. - File Selection and Name Processing

ZIP Processing File Selection

This section discusses how file selection is performed for ZIP processing with ***PKZIP for iSeries***. The primary commands used for ZIP processing are discussed here, along with some overview notes and known restrictions.

This section also discusses how files are selected within an iSeries environment. Remember, ZIP directory entries within a ZIP archive will be defined in a system-independent format, which is not iSeries compatible.

Note: Directory entries within a ZIP archive are actually in a format compatible with UNIX systems and have been translated into the ASCII character set. In addition, the dataset level separators are typically set as the forward slash ("/"), not the period (".") as in iSeries, although this can be controlled through command actions in ***PKZIP for iSeries***.

See Chapter 7. - iSeries File Processing Support for further information on how ***PKZIP for iSeries*** handles file name interchanges between iSeries and common ZIP format.

Primary File Selection Inputs

PKZIP for iSeries will only process:

1. iSeries objects of type FILE (only with attributes PF-SRC, PF-DTA, and SAVF).
2. IFS Stream files (*STMR) and IFS Directories(*DIR).
3. Spool Files.

Other objects must first be unloaded into an iSeries save file (SAVF) before they can be processed by ***PKZIP for iSeries*** (see: Use of SAVF Method).

The FILES parameter in both PKZIP and PKUNZIP specifies which files are to be processed for all files except Spool Files (SPLF have their own selection parameters). One or more names can be specified, and each name is in either OS/400 QSYS format, or IFS format, depending on F2ZTYPE settings. An asterisk may be used at the end of the library name, file name, or member name to select names beginning with the prefix used. To select all members of a file, *ALL may be used. To select all files in a library *ALL may be used (as long as it is qualified by at least a library name), for example, FILES('mylib/*ALL'). If *ALL is specified without at least a qualifying library name, the specification is ignored and no files will be selected.

The ***PKZIP for iSeries*** QSYS file system expands a partial file specification in several ways to make file specification more convenient. Each file specification may consist of a filename; a library name; a file name and member name; a library name and file name; or a library name, file name, and member name. iSeries SAVF may also be selected, but because a *SAVF file does not contain members, a SAVF will not be selected if a member name was included in the file specification.

In the Integrated File System, each file specification may consist of a directory, a path of directories, a directory and file, or a path of directories and file.

The various combinations that may be used are shown below:

File Type	File specification	Expanded As	Notes
QSYS	library*/	library*/*all(*all)	Finds all files in libraries beginning with *library*.
	fileinlib	*LIBL/fileinlib(*ALL)	Searches library list for all files called *fileinlib*. If a matching file is found, all of its members will be selected. If a SAVF is found, it will be selected.
	fileinlib*(mem*)	*LIBL/fileinlib*(mem*)	Searches library list for all files beginning with *fileinlib*. If a matching file is found, members beginning with *mem** will be selected. If a SAVF is found, it will NOT be selected because the file specification includes a member name.
	library*/file*	library*/file*(*ALL)	Searches libraries that begin with *library prefix* and for files that begin with *file prefix*. If a matching file is found, all of its members will be selected. If a SAVF is found, it will be selected.
	library*/file*(memo*)	library*/file*(mem*)	Searches libraries that begin with *library* prefix and files that begin with *file* prefix. If a matching file is found, members beginning with *mem* prefix will be selected. If a SAVF is found, it will not be selected because the file specification includes a member name.
IFS	Dir/*	Dir/*all	Searches all files in path DIR.
Spool Files	N/A		Uses parameters: SPLFILE, SFUSER,

			SPLFILE, SFUSER, SFQUEUE, SFFORM, SFUSRDTA, SFSTATUS, SFJOBNAM, and/or SPLNBR.

Note: If parameter TYPE(*DELETE) is used, then the file name format for these names must be in MS/DOS format (that is if CVTFLAG has not been used). See the FILES keyword. Files may also be excluded. See the EXCLUDE keyword.

The valid parameter values for the FILES keyword are as follows:

'file specification 1' 'file specification 2'...'file_specification nn'

This is the list of one or more file specifications, separated by spaces.

For example, mylib/myfile(prf*)

mylib/*all(*all)

By default, ***PKZIP for iSeries*** does a match on files in the QSYS library system with **no** case sensitivity and in the IFS with case sensitivity. Some IFS file systems contain case sensitive file names. To force ***PKZIP for iSeries*** to perform NON-case sensitive file name matching use TYPFL2ZP(*IFS2).

File Exclusion Inputs

Using similar file specification techniques as described above in Primary File Selection Inputs, ***PKZIP for iSeries*** can specify from one to many file patterns that will be used to exclude files that were selected with the FILE parameter. The files can be inputted into the command parameter EXCLUDE or into a text file that can be processed by parameter EXCLFILE.

Care should be taken when using wildcards excluding inputs to ensure that FILES and EXCLUDE parameters select the desired files.

Input ZIP Archive files

During a FRESHEN or UPDATE request, files contained within the old ZIP Archive are added to a candidate list. Names stored previously are used to search the system files for viability (any file names not found in the system remain in the ZIP Archive).

SPOOL File Selecting

The FILES parameter is not used to select spool files for compression, but instead uses its own selection parameters.

There are eight positional parameters that can be specified to select the spool files: the Spool File Name (SPLFILE), the Spool File Number (SPLNBR), the user that created the files (SFUSER), the OUTQ that the file is residing (SFQUEUE), the form type specified (SFFORM), the user data tag associated with the spool file (SFUSRDTA), the status of the Spool File (SFSTATUS), or the specific Job Name/User Name/Job Number (SFJOBNAM). Only files that meet all of the selection values will be selected.

If the parameter SFJOBNAM is coded, the job must exist and the parameter SFUSER will be ignored, since it is already part of the SFJOBNAM parameter.

Chapter 5. - ZIP Files

Data Format - Text Records vs. Binary Records

Binary data is stored in a ZIPPED Archive in its original format. Binary data may be graphics or numbers that are already in "computer format"; therefore, no translation is done, such as, EBCDIC will stay EBCDIC. The length of binary records in UNZIP processing is determined by the archive's fixed-length records. ***PKZIP for iSeries*** will fill the available block automatically according to allocation specifications.

In the context of ZIPPED Archives, a "text file" is one that is stored in the ASCII format. A text file contains records of data, each separated by a delimiter to signify the end of the record.

Note: An EBCDIC file containing text information (such as source code) can be stored in its original format by using BINARY, but it is not considered to be a "text" file within the ZIP architecture.

PKZIP for iSeries uses the default line delimiter CR-LF (x'0D0A') at the end of each text record. Text File members in the QSYS Library file system use new line characters (NL=X'15') internally. ***PKZIP for iSeries*** will handle the CR-LF and NL in both extraction and compressions automatically.

At the time of PKUNZIP file extraction, ***PKZIP for iSeries*** will convert text data from ASCII to EBCDIC by using a translation table. During installation, several translation tables are available, and the customization process will select one of the translation tables as a default. Additional translation tables may be created through the customizing procedure.

Situations may arise in unique platform interchanges, or when working with text files from other countries where the default text translation table is not adequate. Users may select any available translation table by using TRAN and FTRAN parameters.

PKZIP for iSeries extracts text records stored in the ZIP Archive by examining data for record delimiter and file terminator indicators. Using these indicators, ***PKZIP for iSeries*** aligns records in accordance with target file attributes.

Text files (such as program source code) are held within an archive using the ASCII character set for compatibility with other versions of ***PKZIP®***. For these to be usable on OS/400, they must be converted to the IBM EBCDIC character set. Additionally, the carriage return and line feed characters must be removed before writing lines to a file because OS/400 files are record-based and do not use control characters to separate records or lines. Text files usually have spaces at the end of a line. When using the text file handlers, ***PKZIP for iSeries*** has less data to read because the input/output routines remove trailing spaces and replace them with a new line character. This improves ***PKZIP for iSeries*** performance.

When extracting files from an archive, ***PKZIP for iSeries*** must know whether to perform text conversions. ***PKZIP for iSeries*** stores an indicator in the archive file's local header defining if a file is binary or text-based. Because this indicator may be wrong in some circumstances, use the FILETYPE keyword to specify whether text conversions are required. When adding files to an archive, ***PKZIP for iSeries*** will flag the file according to the FILETYPE used.

PKZIP for iSeries uses translation tables that should be suitable for most customers, but some users may wish to alter the tables. The procedure for changing the translation tables is discussed. If text files are only used on iSeries, then the FILETYPE(*EBCDIC) may be used. This uses iSeries files "as is" for the file (which are faster for text files), but does not translate the data to ASCII. This will provide a small improvement in performance.

Additionally, ***PKZIP for iSeries*** will translate each character in a text file from EBCDIC character format to ASCII character format by default. This is done using one of the two internal translation tables, which are

named UKASCII and USASCII. It is recognized that these translation tables may not suffice for all countries or all situations, especially on those sites where text files are received from several different countries for processing into a single format. The source of the translation tables used by the PKZIP and PKUNZIP programs has been supplied, together with instructions for modifying the tables to create additional files (see the Translation Tables section of this chapter for more details). This enables sites to modify the translation table as required.

In a case where FILETYPE is neither *TEXT nor *BINARY, *DETECT is the default mode. ***PKZIP*** will read up to 64K of data from the input file and scan it for non-translatable text characters using the active text translation table. If any characters will not translate successfully using this method, the entire file will be treated as if *BINARY has been used.

Note: One exception to this is X'00' or the NULL terminator character, which is commonly used in C language. The NULL character will be allowed within the files. If file type is of a file in the archive is un-known whether it is text or binary, the user may use the TYPE(*VIEW) and VIEWOPT(*DETAIL) parameters to examine the file attributes.

File Attributes

Within each ZIP Archive there are two different directories providing information about the files held in that archive. A Local Directory is included at the front of each file, with information pertaining to each file (for example: file size and date ZIPPED), and a Central Directory is located at the end of the ZIP Archive. The Central directory lists the complete contents of the ZIP Archive and is the primary source of information for UNZIP processing.

PKZIP for iSeries will store extended attributes about the file that can be useful in recreating the file during UNZIP processing. See Appendix L - Extended Attributes.

PC Shared Drives Format

One common mistake made when extracting a text file to a shared drive folder in the IFS where the use of the file will be by a Windows application is to extract the file in text mode. Extracting a file as a TEXT file on the iSeries will cause PKUNZIP to translate the file to the EBCDIC format, since this is the native iSeries format. The Windows application expects the file to be in ASCII, so therefore this file should be extracted using Binary, since the files are store in ASCII in the archive.

Chapter 6. - File Extracting Process

Extracting Files to the QSYS Library File System

When extracting files to the QSYS Library File system using TYPFL2ZP(*DB), some items to consider are 1) does the file exist or will a new file be create?, 2) did the file come from PKZIP for iSeries or did it come from another platform?, 3) are the files text type files from another platform where I need to know the record length, and etc. These are just a few questions that might impact how you want to extract the files.

If the file does not exist and if the file did not come from PKZIP for iSeries, you should provide a record length for the file with the parameter DFTDBRECLN. If the file is coming from PKZIP for iSeries or PKZIP for zSeries the record length will be in the extra data. If the file is from PKZIP for iSeries and the parameter was DBSERVICE(*YES), the complete database definition from the extract data for database will be used to create the file.

If the file is to be created as text and the record length is too short then you will receive messages indicating the records are being truncated.

Two common parameters that are used to alter or guide the extraction process are the EXDIR and DROPPATH parameters. EXDIR provides the path library or library/file that the file will be extracted to when no library or path exist for the files in the archive. Of course this is where the DROPPATH comes in to drop the first path or library with *LIB or to remove all paths in a name with *ALL.

Some Examples might be the files in the archive might look like:

```
Archive/#1:
    My Document/myfiles/test/myheader.txt
    My Document/myfiles/test/mydata.txt
    My Document/myfiles/test/mytrailer.txt

Archive/#2:
    QGPL/QCLSRC/MYCL01
    QGPL/QCLSRC/MYCL02
    QGPL/QCLSRC/MYCL03
    QGPL/QCLSRC/MYCL04
```

In archive #1 lets assume the all three text files are of different records lengths. If we want to extract each with their own length, we would have to make three runs to create the files with different parameters. Or we could use CRTPF and create each of the files so the files would exist with the proper record length.

➔ PKUNZIP ARCHIVE('Archive/#1') FILES('My Document/myfiles/test/myheader.txt') TYPE(*EXTRACT) EXDIR('MYLIB/MYHEADER') DROPPATH(*ALL) DFTDBRECLN(50) CVTTYPE(*DROP)

➔PKUNZIP ARCHIVE('Archive/#1') FILES('My Document/myfiles/test/mydata.txt') TYPE(*EXTRACT) EXDIR('MYLIB/MYDATA') DROPPATH(*ALL) DFTDBRECLN(150) CVTTYPE(*DROP)

➔PKUNZIP ARCHIVE('Archive/#1') FILES('My Document/myfiles/test/myheader.txt') TYPE(*EXTRACT) EXDIR('MYLIB/MYTRAILER') DROPPATH(*ALL) DFTDBRECLN(20) CVTTYPE(*DROP)

The above would create three files in library MYLIB and with all files having different record lengths.

Now suppose the files already exist with the names and record lengths. In this case, we could do all three files at once with:

➔ PKUNZIP ARCHIVE('Archive/#1') TYPE(*EXTRACT) EXDIR('MYLIB/?MBR') DROPPATH(*ALL) CVTTYPE(*DROP) OVERWRITE(*YES)

The ?MBR will force each member name to also become the file name.

In archive #2 lets assume, we want to extract the CL source member and place them in a different library call MYNEWLIB. If the QCLSRC file does not exist and the archive was not built with DBSERVICE(*YES), then you would need to do a ➔ CRTSRCPF FILE(MYNEWLIB/QCLSRC) to have the file setup correctly for source files. If the QCLSRC file already exist in MYNEWLIB or the archive was built with DBSERVICE(*YES), then no special handling is required.

Extracting Files to the IFS

When extracting files to the Integrated File system with TYPFL2ZP(*IFS), record lengths are not a concern as they were in the QSYS Library file system. The main considerations when extracting to the IFS is "what paths do you want for the file, or should the file be stored in EBCDIC or ASCII".

Path Considerations:

If the Name of Files in the archive, starts with '/', then with no other changes this will be extracted to a the root system with the first name in the path. This is call fully qualified path.

If the name does not start with a '/', (call relative path) the item will be extracted to the paths based on the Current directory (DSPCURDIR).

In both cases if the path(s) does not exist, the path(s) will be created with the attributes of the parent folder.

Changing the path(s):

In cases where the path that is stores with a name of the file in archive is not desired, then using the EXDIR and DROPPATH parameters should help guide the file to where it should be placed.

Using EXDIR, you can define the path of the file(s) that will be extracted. If you need to remove the path of the file in the archive, you can use DROPPATH(*ALL) to remove all the paths before extracting or you can use DROPPATH(*LIB) to remove only the first path name.

Again the coding of EXDIR follows the same rule with regards to Full qualified path or relative path.

File Type Considerations:

When extracting a file, the decision to whether the contents of file should be stored in ASCII or EBCDIC needs to be made.

If the file is not a text file, it does not matter and should be stored as Binary. If the file is text, and will be used by a PC program, chances are the data is expected to be in ASCII. Since the files are stored in the archive as ASCII, these files should be extricated as TYPEFILE(*BINARY). If the file is to be used by an AS/400 application or will be translated later, then chances are the file should be stored in EBCDIC. In this case use TYPEFILE(*TEXT) to extract the file in EBCDIC.

Extracting Spool Files

When Extracting spool files with PKUNZIP, the attributes at the time compression will be preserved except for new spool file numbers that will be generated. Parameter SPLUSRID is for the user ID on the new extracted spool file. If it is *DFT the original user ID will stay with the new spool file. Parameter SFQUEUE is for the OUTYQ and OUTQ Library that the new extracted spool file will be placed. If *DFT is specified then the original OUTQ will be used to place the spool file.

Note on extracting Spool Files: To create or extract spool file with PKUNZIP, the user must have *USE authority to the API QSPCRTSP. The normal setting for the API QSPCRTSP is Authority PUBLIC(*EXCLUDE). The API authority is set this way so that system administrators can control the use of this API. This API has security implications because you can create spooled file from the data of another spooled file. To allow user to extract spool files change the API authority on a need basis.

When extracting a spool file with PKUNZIP, the new spooled file will be created with attributes based on values taken from the spooled file attributes when PKZIP archived the spool file. The spool file's File Number, Job, Job User, Job Number, date, and time are controlled by IBM OS/400 operating system during the creation of a spool file.

The new spool file that is created by the PKUNZIP is spooled under one of two jobs and is dictated by IBM's create spool file API. The job is determined by the user-name field from the attributes. If the user name is the current user, it is a part of the user's job and is owned by the user profile that the job was started with. First the user profile for the user name must already exist. When using the user id override (parameter SPLUSRID), the spool file will be now belong to the override user.

If the ownership of the new spooled file is assigned to a different user by a different user profile name in the user-name field from the attributes, then the current user must have *SPLCTL authority to assign the spooled file to another user. When this is done, the new spooled file is by the user specified in the user name field or override parameter. The new spooled file is then part of a **special system job (QPRTJOB)** that is created for each user.

The new spooled file is placed on the output queue specified in the output queue name field from the original spool file attributes. If the parameter SFQUEUE is used it will override the attribute for the output queue.

In both cases, the spooled file name is the one contained in the spooled file attributes parameter. The spooled file number will be the next sequential one available for the job that the spooled file becomes a part of.

OS/400 Authority requirements when extracting spool files:

1. *Special Authority* - *SPLCTL. This authority is needed if you are creating a spooled file for another user.

2. *Output Queue Authority* -*USE

3. *Output Queue Library Authority* - *EXECUTE

4. *Object QSPCRTSP API Authority* -*USE

The following are several examples of results of extracting spool files:

Start with archiving the following spool files that were created with job MYJOB1 and User EVWSS:

```
File        File Nbr  Job         User        Number  Date      Time
QSYSPRT          397  MYJOB1      EVWSS       010893  12/11/02  13:35:29
QSYSPRT          398  MYJOB1      EVWSS       010893  12/11/02  13:36:09
QSYSPRT          399  MYJOB1      EVWSS       010893  12/11/02  13:36:09
```

Now extract with job MYJOB1 and User EVWSS but I am now on a different day and job number:

```
File        File Nbr  Job         User        Number  Date      Time
QSYSPRT            2  MYJOB1      EVWSS       010927  12/12/02  09:57:42
QSYSPRT            3  MYJOB1      EVWSS       010927  12/12/02  09:57:42
QSYSPRT            4  MYJOB1      EVWSS       010927  12/12/02  09:57:42
```

Next extract with job MYJOB2 and User EVWSS but I am now on a different day and job number:

File	File Nbr	Job	User	Number	Date	Time
QSYSPRT	2	MYJOB2	EVWSS	010928	12/12/02	09:59:06
QSYSPRT	3	MYJOB2	EVWSS	010928	12/12/02	09:59:06
QSYSPRT	4	MYJOB2	EVWSS	010928	12/12/02	09:59:06

Next using the User override with the SPLUSRID(WSS) and submit MYJOB1 with User EVWSS.

File	File Nbr	Job	User	Number	Date	Time
QSYSPRT	26	QPRTJOB	WSS	010118	12/12/02	10:02:50
QSYSPRT	27	QPRTJOB	WSS	010118	12/12/02	10:02:50
QSYSPRT	28	QPRTJOB	WSS	010118	12/12/02	10:02:50

Notice that the JOB was changed to QPRTJOB since the user being extracted was different than the user running the Job.

Next being signed on as user WSS, submit a job MYJOB11 with the job parameter USER profile specified for user EVWSS.

```
===> SBMJOB CMD(PKUNZIP ARCHIVE('atest/splftst/tst02') TYPE(*EXTRACT))
                JOB(MYJOB11) USER(EVWSS)
```

File	File Nbr	Job	User	Number	Date	Time
QSYSPRT	2	MYJOB11	EVWSS	010936	12/12/02	10:25:25
QSYSPRT	3	MYJOB11	EVWSS	010936	12/12/02	10:25:25
QSYSPRT	4	MYJOB11	EVWSS	010936	12/12/02	10:25:25

Chapter 7. - iSeries File Processing Support

PKZIP for iSeries can support files maintained in both the traditional QSYS Library file system and in IFS (Integrated File System) along with supporting Spool files.

QSYS (Library File System)

The QSYS file system supports the iSeries library structure. This file system provides access to database files and all other iSeries object types that library manages. On the IBM iSeries system, each QSYS type file (also called a file object) has a description that details the file characteristics and how the data associated with the file is organized into records, and, in many cases, the fields associated for each record. Whenever a file is processed, the iSeries uses this description.

Of the objects in the library system, ***PKZIP for iSeries*** will only process physical files that have an attribute type of PF-DTA (physical data files), PF-SRC (physical source file), or SAVF (save files).

QSYS files always exist in a library, and the PF-DTA and PF-SRC files (if data exist) will always have one to many members in the file. Therefore, PF-DTA and PF-SRC files have a name format of "library/file(member)." A SAVF (a special type of iSeries file for saving and restoring iSeries objects) does not have any members giving a file format of "library/file." Because SAVF types are handled in a special way, they are given additional consideration (see SAVF and Use of SAVF Method).

QSYS Summary

If the archive file is to be in the QSYS file system, set parameter TYPARCHFL(*DB).

If the file being compressed or extracted is in the QSYS file system, set parameter TYPFL2ZP(*DB).

If the list files (Appendix E - List Files and their Usage) are to be in the QSYS file system, set parameter TYPLISTFL(*DB).

Format Summary:

PF-DTA	LIBRARY/FILE(MEMBER)
PF-SRC	LIBRARY/FILE(MEMBER)
SAVF	LIBRARY/FILE

IFS (Integrated File System)

The Integrated File System is a part of iSeries which supports stream input/output and storage management similar to personal computer and UNIX operating systems, while providing an integrating structure over all information stored in the iSeries.

The key features of the integrated file system are:

- Support for storing information in stream files that can contain long continuous strings of data. These strings of data might be, for example, the text of a document or the picture elements in a picture. The stream file support is designed for efficient use in client/server applications.
- A hierarchical directory structure that allows objects to be organized by specifying the path through the directories to an object for access to an object.

- A common view of stream files that are stored locally on iSeries, Integrated Netfinity Server for iSeries, or a remote Windows NT server. Stream files can also be stored remotely on a Local Area Network (LAN) server.

Directories and Current Directory

A **directory** is a special object that is used to locate objects by names specified by users. Each directory contains a list of objects that are attached to it, and that list may include other directories.

A **current directory** is the first directory in which the operating system locates files, and it also stores temporary files and output files. When you request an operation for an object, such as a file, the system searches for the object in the current directory, unless a different directory path is specified. The current directory is similar in nature to the current library. If the file selection does not start with '/' (Root Directory), the files should be in the path of the current directory.

Path and Path names

A **path name** (also called a **pathname** on some systems) informs the system how to locate an object. The path name is expressed as a sequence of directory names followed by the name of the object. Individual directories and the object name are separated by a slash (/) character. An example might be: directory1/directory2/file.

For convenience, the back slash (\) can be used instead of the slash in integrated file system commands.

There are two ways of indicating a path name:

- An **absolute path name** begins at the highest level, or **root directory** (which is identified by the / character). For example, consider the following path from the / directory to the file named Testit. */mydept/myfiles/testit*.
- If the path name does not begin with the / character, the system assumes that the path begins at your current directory. This type of path name is called a **relative path name**. For example, if your current directory is *mydept* and it has a sub-directory named *myfiles* containing the file *testit*, the relative path name to the file is: *Myfiles/testit.* Notice that the path name does not include the name of the current directory. The first item in the name is the directory or object at the next level below the current directory.

Stream Files

A **stream file** is a randomly accessible sequence of bytes with no further structure imposed by the system. The integrated file system provides support for storing and operating on information in the form of stream files. Documents that are stored in iSeries folders are stream files. Other examples of stream files are PC files and the files in UNIX systems. An integrated file system stream file is a system object that has an object type of *STMF.

Other IFS Objects

There are other object types (such as link objects, etc.) in IFS which at this time are not supported by ***PKZIP for iSeries***.

File Systems in the IFS

There are currently ten (10) file systems that are part of the Integrated Files system. Each file system is a major sub-tree in the IFS directory structure. A **file system** provides the support to access specific segments of storage that are organized as logical units. These logical units on the iSeries are files, directories, libraries, and objects.

Each of these file systems has a set of logical structures and rules for interacting with information in storage. These structures and rules may be (and often are) different from one file system to another. The IFS treats the library support and folders support as separate file systems.

The ten file systems are:

1. **"root"** - / **file system.** This file system takes full advantage of stream file support and hierarchical directory structure of the integrated file system. The root file system has the characteristics of the Disk Operating System (DOS) and OS/2 file systems. Most of references throughout this guide refer to the "root" system.
2. **QDLS** - **Document Library Services file system.** This file system provides access to documents and folders. See IBM's *Office Services Concepts and Programmer's Guide (SH21-0703)* for additional information.
3. **QOPT** - **Optical file system.** This file system provides access to stream data that is stored on optical media (such as CDs). See IBM's *Optical Support (SC41-5310)* for additional information.
4. **QSYS.LIB** - **Library file system.** This file system supports the iSeries library structure and provides access to database files and all of the other iSeries object types that the library support manages.
5. **NFS** - **Network File System.** This file system provides the user with access to data and objects that are stored on a remote NFS server. An NFS server can export a network file system that NFS clients will then mount dynamically. See IBM's *OS/400 Network File System Support (SC41-5714)* for additional information.
6. **QFileSvr.400.** This file system provides access to other file systems that reside on remote iSeries systems. See IBM's *Integrated File System Introduction (SC41-5711)* for additional information.
7. **QNetWare** - **QNetWare file system.** This file system provides access to local or remote data and objects that are stored on a server that runs Novell NetWare 4.10 or 4.11 or to standalone PC Servers running Novell Netware 3.12, 4.10, 4.11, or 5.0. A user can mount NetWare file systems over existing local file systems dynamically. See *File Management (SC41-5710)* for additional information.
8. **QNTC Windows NT Server file system.** This file system provides access to data and objects that are stored on a server running Windows NT 4.0 or higher. It allows iSeries applications to use the same data as Windows NT clients. This includes access to the data on a Windows NT Server that is running on an integrated PC Server. See IBM's *OS/400-iSeries Integration with Windows NT Server (SC41-5439)* for details.
9. **QOpenSys** - **Open Systems file system.** This file system is compatible with UNIX-based open system standards, such as POSIX and XPG. Like the root file system, this file system takes advantage of the stream file and directory support that is provided by the integrated file system. In addition, it supports case-sensitive object names. See IBM's *Integrated File System Introduction (SC41-5711)* for additional information.
10. **UDFS** - **User-Defined File System.** This file system resides on the Auxiliary Storage Pool (ASP) of the user's choice. The user creates and manages this file system. See IBM's *Integrated File System Introduction (SC41-5711)* for additional information.

PKZIP for iSeries works with all file systems, but the rules of each file system must be adhered to or a File I/O error will most likely occur. In most cases, the files can be compressed and extracted in one run when all the file names and paths meet the file system's rules. When creating an archive file in one file system, one restriction is that when using the TMPPATH option, the temp path must also be in the same file system as the archive files.

On the following pages are rules for some of the most used file systems.

Document Library Services File System (QDLS)

The QDLS file system supports the folders structure. It provides access to documents and folders. Additionally, it supports iSeries folders and document library objects (DLOs) and supports data stored in stream files.

Considerations and Limitations:

1. You must be enrolled in the system distribution directory when working with objects in QDLS.
2. QDLS converts the lowercase English alphabetic characters *a* through *z* to uppercase when used in object names. Therefore, a search for object names using only those characters is not case sensitive. All other characters are case sensitive in QDLS.
3. Each component of the path name can consist of just a name, such as: /QDLS/MYFLR1/MYDOC1 - or - a name plus an extension (similar to a DOS file extension), such as: /QDLS/MYFLR1/MYDOC1.TXT.
4. The name in each component can be up to 8 characters long, and the extension (if any) can be up to 3 characters long. The maximum length of the path name is 82 characters, assuming an absolute path name that begins with /QDLS.
5. The directory hierarchy within QDLS can be 32 levels deep.
6. Must have proper authority within the path.
7. The folders in the path must already exist.
8. PKZIP will not create folders at this time.

For more details, see the “Rules for Specifying Folder and Document Names” discussion in the publication *CL Reference*.

Optical File System (QOPT)

The QOPT file system provides access to stream data that is stored on optical media (such as CDs). Additionally, it provides a hierarchical directory structure (similar to PC operating systems such as DOS and OS/2), is optimized for stream file input/output, and supports data stored in stream files (known as DSTMF or Distributed Stream Files).

Considerations and Limitations:

1. QOPT converts the lowercase English alphabetic characters a to z to uppercase when used in object names. Therefore, a search for object names using only those characters is not case-sensitive. For more details, see the publication *Optical Support*.
2. The path name must begin with a slash (/) and contain no more than 294 characters. The path is made up of the file system name, the volume name, the directory and sub-directory names, and the file name. For example: /QOPT/VOLUMENAME/DIRECTORYNAME/SUBDIRECTORYNAME/FILENAME
3. The file system name (/QOPT) is required.
4. The volume name is required and can be up to 32 characters long.
5. You can include one or more directories or sub-directories in the path name, but QOPT requires none. The total number of characters in all directory names and sub-directory names (including the leading slash) cannot exceed 256 characters. Directory and file names allow any character except X'00' through X'3F', X'FF', lowercase alphabetic characters, and the following characters:
   - Asterisk (*)
   - Hyphen (-)
   - Question mark (?)
   - Quotation mark (")

- Greater than (>)
- Less than (<)

6. The file name is the last element in the path name. The file name length is limited by the directory name length in the path. The directory names and file name combined cannot exceed 256 characters, including the leading slash.

For more details on path name rules in the QOPT file system, see the "Path Name Rules" discussion in the publication *Optical Support*.

Using QSYS.LIB Through the Integrated File System Interface

Even though ***PKZIP for iSeries*** accesses the QSYS Library file system directly, there is an ability to access the QSYS.LIB file system through the integrated file system interface. In using the integrated file system interface, you should be aware of the following considerations and limitations:

1. File handling restrictions in the QSYS.LIB file system are:
    - Logical files are not supported.
    - Physical files supported for text mode access are program-described physical files containing a single field and source physical files containing a single text field. Physical files supported for binary mode access include externally-described physical files in addition to files supported for text mode access.
    - If any job has a database file member open, only one job is given write access to that file member at any given time. Other requests are allowed read-only access.
2. In general, the QSYS.LIB file system does not distinguish between uppercase and lowercase in the names of objects. A search for object names achieves the same result, regardless of whether characters in the names are uppercase or lowercase. If a name is enclosed in quotation marks, the case of each character in the name is preserved. A search involving quoted names, therefore, is sensitive to the case of the characters in the quoted name.
3. Each component of the path name must contain the object name followed by the object type of the object. For example: /QSYS.LIB/TESTLIB.LIB/MYFILE.FILE/MYFILE.MBR. The object name and object type are separated by a period (.). Objects in a library can have the same name if they are different object types, so the object type must be specified uniquely to identify the object.
4. The object name in each component can be up to 10 characters long, and the object type can be up to 6 characters long.
5. The directory hierarchy within QSYS.LIB can either be two or three levels deep (two or three components in the path name), depending on the type of object being accessed. If the object is a database file, the hierarchy can contain three levels (library, file, or member), otherwise, there can be only two levels (library or object). The combined length of each component name plus the number of directory levels determine the maximum length of the path name. If / and QSYS.LIB are included as the first two levels, the directory hierarchy for QSYS.LIB can be up to five levels deep.
6. The characters in names are converted to code when the names are stored. Quoted names, however, are stored using the code page of the job.

For information about code pages, see the publication *National Language Support*.

IFS Summary

Only directories and Stream Files are supported by ***PKZIP for iSeries***.

If the archive file is to be in IFS, set parameter TYPARCHFL(*IFS).

If the file being compressed or extracted is in IFS, set parameter TYPFL2ZP(*IFS)

If the files to selected for compression are to be NON-case sensitive set parameter TYPARCHFL(*IFS2).

If the list files are to be in IFS (See Appendix E - List Files and their Usage), set parameter TYPLISTFL(*IFS).

Format Summary:

Directory	Directory1/directory2	will be current directory
Stream File	filename or directory/filename	will be current directory
Full Path	/Directory1/Directory2/filenam e	

For more information, see reference IBM publication **Integrated File System Introduction** (SC41-5711) or visit the IBM web site.

SAVF

SAVF, denoted by the OS/400 system TYPE(*FILE) and ATTR(SAVF), is a special form of file designed specifically to handle save/restore data in the iSeries system.

Some SAVF special characteristics are:

- The SAVF is always processed as binary with all records being 528 characters in length.
- Only a save and restore iSeries function can update or change data.
- A SAVF will not be selected if a member name is included in the file specification.

SAVF are a means to compress other iSeries object types (programs, modules, commands, logical files, triggers, etc.) that are in the iSeries system by first doing a SAVLIB or SAVOBJ for those objects to a SAVF. Then you can compress and extract the SAVF.

Compressing a SAVF file

The only difference when compressing a SAVF is not to specify a member (only library/file). If a member is specified, then no SAVF types will be compressed.

Extracting Records into a SAVF file

It is helpful before extracting records from a ZIP Archive to be aware of what file names and file attributes are being stored for the compressed file. VIEWOPT(*DETAIL) may be used on the Archive to verify the information. An attribute is stored in the archive header that identifies if the file is a SAVF. The PKUNZIP program will also retain the original attribute from the extended attributes, such as SAVF Description and Library Description.

A common problem in some iSeries environments is that some users may not have the authority to the SAVF commands which can result in failures.

Overwriting Current SAVF File

When extracting a compressed file, it may be desirable to overwrite the existing file. By using the OVERWRITE(*YES) parameter, PKUNZIP will first issue a CLRSAVF command to clear the save. This demonstrates why care should be taken when extracting a SAVF.

Compressing Spool Files

PKZIP for iSeries has the ability to select, compress and extract spool files. Not only can a spool file be compressed, they can be converted to other document formats that will allow the document file to be distributed and read by other media and software.

All spool files are eligible for compression but only Spool File Types *SCS, *IPDS are supported for Text Document conversion.

By using the PKZIP command and setting parameter TYPFL2ZP(*SPL), other parameters will be shown to help select the spool files. To assist, a new command PKZSPOOL is provided to sequenced the selections and to eliminate parameters that are not valid for the selection of spool files.

Spool File parameters specifies the group of spool files that are to be selected. Eight positional values can be specified to select the spool files: the Spool File Name (SPLFILE), the Spool File Number (SPLNBR), the user that created the files (SFUSER), the OUTQ that the file is residing (SFQUEUE), the form type specified (SFFORM), the user data tag associated with the spool file (SFUSRDTA), the status of the Spool File (SFSTATUS), or the specific Job Name/User Name/Job Number (SFJOBNAM). Only files that meet all of the selection values will be selected. A sample of the default selection parameters is shown in the window below:

Selection Sample using PKZSPOOL Command.

```
                  SPLF File Compression  5.6 (PKZSPOOL)

Type choices, press Enter.

Archive Zip File name  . . . . . > myar



Spool File . . . . . .   *ALL          Spool File Name,  *All
Spool File User  . . . . . .   *CURRENT      User ID, *CURRENT, *ALL
             + for more values
Output Queue Name  . . . . . .   *ALL          OutQ name, *ALL
  Library  . . . . . .   _________          Library, *LIBL, *CURLIB
Print Form Type  . . . . . .   *ALL          Form Type, *STD, *ALL
Print File User Specified Data     *ALL          User Data,  *all
Spool Files Status . . . . . .   *ALL          *ALL, *READY, *HELD...
             + for more values
Spool File Job Name  . . . . . .   _________      Job name, blank for all
  User . . . . . .   _________      User Id
  Job Number . . . . . .   ______       Job Number
Spooled file number  . . . . . .   *ALL          1-9999, *ALL, *LAST
Target File Format . . . . . .   *SPLF         *SPLF, *TEXT, *PDF, *TEXT1...
Target File Name . . . . . .   *GEN1
Type of processing . . . . . .   *ADD          *ADD, *UPDATE, *FRESHEN ..
Compression Level  . . . . . .   *SUPERFAST    *NO, *FAST, *NORMAL, *MAX...
File Types . . . . . .   *DETECT       *DETECT *TEXT *BINARY ........
Zip Spool Files  . . . . . .   *SPL          *SPL
Archive Password . . . . . .
```

After defining what spool files are to be selected for compression, you will need to define the file format the spool file should be stored in the archive. At this time, there 3 formats of *SPLF (Spool File native mode), *TEXT (ASCII text document with 3 variation of how a new page is handled) and *PDF (Adobe Portable Document Format).

For use with *TEXT and *PDF there are 3 variation of storing the file name in the archive with the parameter SFTGFILE. SFTGFILE (*GEN1) will generate a very specific name using most of the Spool File name attributes to form the file name so that it will not be a duplicated. The name will be built as follows:

"Job-Name**/**User-Name**/#**Job-Number**/**Spool-File-Name**/F**spool- File-Number.Suffix"

For example: "MYJOB/BILLS/#152681/INVOICE/F0021.SPLF"

The suffix is dependent on the SFTARGET setting. *SPLF can only be stored as SFTGFILE (*GEN1).

SFTGFILE (*GEN1P) will generate the same specific name generated by *GEN1 except the '/' for folders will all be replaced by '.' to make the file name one lone name. For example:

For example: "MYJOB.BILLS.#152681.INVOICE.F0021.SPLF"

SFTGFILE (*GEN2) uses the spool file name and appends the spool file number followed by the suffix that is depended on the SFTARGET setting. Caution should be taken in that a duplicate file name in the archive could be created. An example of GEN2 is a spool file INVOICE with Spool File number of 21 that will be converted to a text file will generate a file name of INVOICE21.TXT.

In cases where a very specific name is desired for the file in archive name, SFTGFILE() can be code with the name. This is designed for selecting **only one** file at a time otherwise file names will be duplicated. Or you could add coding to the CVTNAME routine and use the CVTFLAG to generate the file name desired.

Chapter 8. - ZIP Archives

A ZIP Archive is the storage facility for files that are compressed (or simply stored) using the ***PKZIP®*** product. The basic archive can hold up to 65,535 files, which may have been compressed by up to 99% of their original size. Data integrity is validated by a Cyclic Redundancy Check (CRC) to maintain integrity of the data from compression through the extraction process. If the archive contains the ZIP64 archive format, the archive can support more than the 65,535 files and can be larger than 4 Gig (See Large Files Considerations). In addition to the data, file attributes are retained, allowing extraction of the same file characteristics without the need of control card specifications. An Archive can exist in three possible states during processing, described as "Old Archive," "Temporary Archive," and "New Archive." An explanation of the functions of each of these is described in the sections below.

A ZIP Archive is transferable between platforms; such as, files that are compressed by ***PKZIP®*** in one platform may be extracted by ***PKZIP®*** on a different platform, maintaining identical data.

This chapter describes the types of files used by ***PKZIP for iSeries*** and provides a description of the way in which they are accessed by ***PKZIP for iSeries*** ZIP archives.

PKZIP for iSeries (by default) creates a new archives in the *DB file system as members of PF-DTA files with 132-byte records. The archive file is given a text field of "File created by ***PKZIP for iSeries***." The archive member is given a text field of "Member created by ***PKZIP for iSeries***." If you wish to create your own archive (perhaps to have a larger record size, for performance) then you can do so, but try to adhere to the following:

- When you create the file, do not create any members in it.
- After having created the file, change the MAXMBRS parameter for the file from 1 to *NOMAX.

A ZIP archive holds files internally in one of several formats, which are compatible with other platforms supported by ***PKZIP®***. These formats are described here, and several commands are available for transforming files into one of these formats as they are compressed. You may specify in which format a file is stored using the FILETYPE(*BINARY) or FILETYPE(*TEXT) command parameters. OS/400 SAVF are always stored as *BINARY type. If you do not specify FILETYPE(*BINARY) or (*TEXT), then the PKZIP and PKUNZIP programs both will default to FILETYPE(*DETECT). For more information, see FILETYPE(*DETECT).

"Old" ZIP Archive

When the PKZIP or PKUNZIP command is run, the ZIP archive (which is specified by use of the ARCHIVE parameter is known as the old ZIP archive, except when the TYPE(*ADD) parameter is being used to create a new ZIP archive. The old ZIP archive may have been created by ***PKZIP for iSeries*** during an earlier operation or may have been created by ***PKZIP®*** on another platform and transferred from there. When a ZIP archive is being updated (or when PKUNZIP is extracting files from a ZIP archive), the necessary details are taken from the old ZIP archive. It should be noted that when ***PKZIP for iSeries*** is updating a ZIP archive, it takes the necessary data from the old ZIP archive, merges it with any new data, and transfers it to a new ZIP archive (in a temporary member in the same iSeries file as the old archive). When all updating is completed, ***PKZIP for iSeries*** deletes the old ZIP archive and then renames the new ZIP archive to the same name as the old ZIP archive. For this reason a file containing a ZIP archive should allow for at least one temporary member to be allocated. When ***PKZIP for iSeries*** creates an archive file, it uses MAXMBRS(*NOMAX).

“Temporary” Archive File

A temporary archive file refers to an archive work in progress. ***PKZIP for iSeries*** will always use a temporary archive file and its definition depends on the “file system.” If the file system type is IFS, then the temporary archive file will be in the same **directory** of the specified new archive. If the file system type is QSYS, the temporary archive file will become a **member** of the specified archive file. The temporary file or member will have a unique name PKnnnnnnnn (where nn represents an internal random number). When the file has been completed successfully, the temporary name will be renamed to the specified name in the ARCHIVE parameter. If this is a process in which an old archive is being updated, then (if successful) the old archive will be deleted before the rename. If a problem occurs, the temporary archive may stay with the temporary name. View the job log if this happens to determine the status of the archive.

“New” ZIP Archive

When the processing of the temporary dataset is finalized, ***PKZIP for iSeries*** creates a new ZIPPED Archive that is the modified “after” version of the old Archive. The modified name of the old Archive and specified allocation information is transferred automatically to the new Archive after updating, and the old Archive is deleted. A new ZIP archive is created when an old ZIP archive is updated, or when a TYPE(*ADD) parameter (see Chapter 9. - PKZIP Commands) is used with ***PKZIP for iSeries*** where there is no old ZIP archive.

Self Extracting Archive

The self extracting programs are held as binary entities in the file PKZIPSFX of the PKZIP for iSeries library. The appropriate member is loaded and the executable data copied to the beginning of the Archive as a preamble when requested.

The resulting Archive can still be processed by PKZIP for iSeries as a normal ZIP Archive.

When an input Archive containing a self-extraction preamble is passed to PKZIP for iSeries for PKZIP processing and no value is supplied by SELFXTRACT, the default of *MAINTAIN will keep any preamble if one exist. If the parameter SELFXTRACT(*REMOVE) is supplied then the PREAMBLE is removed when writing the new Archive.

A self-extracting Archive can be created from an existing Archive by using SELFXTRACT with a valid self-extractor. If the original Archive contained a preamble, it will be removed and the newly specified preamble will be inserted.

When transferring a self-extracting Archive to a target system, be sure to transfer the Archive in BINARY format and adhere to requirements for executables in that environment. (For example, a Windows program should be saved with an application extension of EXE, and a UNIX file attribute should have executable authorization set via the UNIX chmod command).

The self-extraction programs provided are at the 2.5 level of PKZIP. As such, the following restrictions apply to the operation of the self-extraction program(s). Care should be taken to control the creation of the self-extracting Archive within these restrictions, although the resulting Archive may still be processed with PKZIP programs at higher levels that support these features.

- The number of files in the archive should be limited to 65,535 or less.
- Strong Encryption is not supported.
- The size of the Archive should not exceed 4 gigabytes.
- The uncompressed size of individual files should be less than 4 gigabytes.
- Some target file systems (such as Windows FAT and UNIX Kernals earlier than release 2.4) do not support files greater than 2 gigabytes.

To assist in the usage of the self-extraction programs on the target systems, some of the command parameters are listed below. Note that some parameters may not be valid on all systems. By executing the transferred self-extracting Archive on the target system with “-help”, the commands syntax appropriate to that system will be displayed.

```
Usage: sfx.exe [options] [.ZIP archive] [files...]

Where sfx.exe = the name of the self-extracting executable file

Options:
after        extract files that are newer than or equal to a specified date
               suboptions:
                  "date specification" [format: mmddyy or mmddyyyy]
               e.g.: sfx.exe -aft=12311999 file.zip

before       extract files that are older than a specified date
               suboptions:
                  "date specification" [format: mmddyy or mmddyyyy]
               e.g.: sfx.exe -bef=12311999 file.zip

console      display the contents of specified archived files on your screen
               e.g.: sfx.exe -con= file.zip readme.txt

directories  recreate directory path while extracting including any
                  sub-directories
               e.g.: sfx.exe -dir file.zip

exclude      exclude specified files from being extracted
               e.g.: sfx.exe -exc=*.txt file.zip

extract      extract files from the .ZIP archive
               suboptions:
                  all [extract everything in archive]
                  freshen [extract if newer than destination copy]
                  update [extract if newer or not in destination directory]
               e.g.: sfx.exe -ext=all file.zip

help         display help screen
               e.g.: sfx.exe -help

Id         preserve original file uid/gid. Must be root/file owner (UNIX only)

include      include specified files for extraction
               e.g.: sfx.exe -inc=*.txt file.zip

larger       extract files that are the specified size (in bytes) and larger
               suboptions:
                  a numerical value (in bytes) that indicates a minimum desired
                  file size
               e.g.:sfx.exe -larger=400

license      displays license information
               e.g.: sfx.exe -lic

locale       reads and/or adjusts the locale variable for date and time format
             input
               suboptions:
                  environment [read system variable and apply accordingly]
                  "valid country name" [for example locale=germany]
               e.g.: sfx.exe -loc=us -aft=12311999 file.zip

lowercase    change filenames to lower case on extraction
               e.g.: sfx.exe -lowercase

mask         remove specified file attributes upon extraction
               suboptions:
                  archive [mask archive attribute from file(s)/folder(s)]
                  hidden [mask hidden attribute from file(s)/folder(s)]
                  system [mask system attribute from file(s)/folder(s)]
                  readonly [mask read-only attribute from file(s)/folder(s)]
                  none [do not mask attributes from file(s)/folder(s)]
                  all [mask all attributes from file(s)/folder(s)]
               e.g.: sfx.exe -mask=archive,readonly file.zip

more         display output one screen at a time
               e.g.: sfx.exe -more file.zip

newer        process only those files that are newer than a specified
```

```
            (calendar) day in the past
               suboptions:

                 a numerical value (in calendar days) that indicates some
                 date in the past relative to the current date
               e.g.: sfx.exe -newer=2

noextended  suppress the extraction of extended attributes
               e.g.: sfx.exe -noex file.zip

older       process only those files that are older than a specified
            (calendar) day in the past
               suboptions:
                 a numerical value (in calendar days) that indicates some
                 date in the past relative to the current date
               e.g.: sfx.exe -older=2

overwrite   overwrite existing files
                  prompt [prompt before overwriting]
                  all [always overwrite]
                  never [never overwrite]
               e.g.: sfx.exe -o=all file.zip

password    specify a decryption password
               e.g.: sfx.exe -pass=grendel file.zip

print       print the specified archived file
               suboptions:
                  "print device name" [for example print=lpt1]
               e.g.: sfx.exe -print=lpt2 file.zip readme.txt

silent      suppress warning messages when extracting
               e.g.: sfx.exe -silent file.zip

smaller     extract files that are the specified size (in bytes) and
            smaller
               suboptions:
                  a numerical value (in bytes) that indicates a maximum desire
                  file size
               e.g.:sfx.exe -smaller=400

sort        sort files when extracting
               suboptions:
                  crc [sort by crc value]
                  date [sort by date of the file]
                  extension [sort by file extension]
                  name [sort by file name]
                  natural [sort in the order that the file was archived]
                  ratio [sort by compression ratio]
                  size [sort by file size]
                  none [do not sort]
               e.g.: sfx.exe -sort=size file.zip

test        test the integrity of archived files
               suboptions:
                  all [test everything in archive]
                  freshen [test if newer than destination copy]
                  update [test if newer or not in destination directory]
               e.g.: sfx.exe -test=all file.zip

times       preserve specified file date/time stamp
               suboptions:
                  access [preserve accessed date/time stamp on extraction]
                  modify [preserve modified date/time stamp on extraction]
                  create [preserve created date/time stamp on extraction]
                  all [preserve all date/time stamps on extraction]
                  none [do not preserve date/time stamps on extraction]
               e.g.: sfx.exe -time=access,modify file.zip

translate   translate the end of line sequence for give operating system
               suboptions:
                  DOS [convert to DOS style line endings]
                  MAC [convert to MAC style line endings]
                  unix [convert to unix style line endings]
```

```
                e.g.:sfx.exe -translate=unix
version       display SFX version and return appropriate value to the shell
                suboptions:

                   major [return major version number]
                   minor [return minor version number]
                   step [return step or patch version number]
                e.g.: sfx.exe -ver=step

volume        restore the volume label when extracting
                e.g.: sfx.exe -vol file.zip

warning       prompt to continue after warning message
                e.g.: sfx.exe -warn file.zip
```

Chapter 9. - PKZIP Commands

PKZIP Command Summary with Parameter Keyword Format

If the OS/400 command prompt screen is to be used, the command format is simply: PKZIP. There also is a command PKZSPOOL which is the same command as PKZIP, but has the parameter TYPFL2ZP set to *SPL for spool files. The parameters are also re-sequenced for importance of Spool Files.

The command prompt screen is displayed when ENTER or PF4 is pressed. The parameter keywords are displayed on this screen, together with the available keyword options. The required options can be selected before PF4 is pressed to accept the selections. If the command and parameter keywords are entered together on the command line the required format is:

PKZIP keyword1(option) keyword2(option) . . . keyword*n*(option)

Keywords are demarcated by spaces. The keyword “ARCHIVE” is the only positional keyword where the keyword is not required. Whenever the word “path” is used, its meaning depends on the file system that is being used. If IFS is used, path refers to the openness true path type. If the Library systems or *DB is used, path means Library/file and then the file name refers to the member name.

TYPE(	*ADD {*UPDATE} {*FRESHEN} {*MOVEA} {*MOVEF} {*MOVEU} {*DELETE}	)
ADVCRYPT(	{ZIPSTD} {AES128} {AES192} {AES256}	)
ARCHIVE(	*Archive Zip File name* with path	)
ARCHTEXT(	{*NONE} *Archive File Text description*	)
COMPAT(	{*NONE} {*PK400}	)
COMPRESS(	{*FAST} {*MAX} {*STORE} {*TERSE)	)
CRTLIST(	{*NONE} *path/filename*	)
CVTDATA(	*External Pgm Conversion Extended Data*	)
CVTFLAG(	{*NONE} {*400} *External Pgm Conversion Flags*	)

CVTTYPE(	{*NONE} {*DROP} {*SUFFIX}	)
DATEAB(	*mmddyyyy*	)
DATETYPE(	{*NO} {*BEFORE} {*AFTER}	)
DBSERVICE(	({*NO} {*YES)	)
DELIM (	({CRLF} {CR } {LF } {LFCR }	)
DFTARCHREC(	{132} {decimal number}	)
DIRNAMES(	{*YES} {*NO}	)
DIRRECRS(	{*NONE} {*FULL} {*NAMEONLY}	)

EXCLFILE(	{*NONE} *path/filename*	)
EXCLUDE(	file_specification 1, file_specification 2, file_specification n	)

EXTRAFLD(	{*YES} {*NO} {*CENTRAL} {*LOCAL} {*BOTH same as *YES}	)
ERROPT(	{*END} {*SKIP}	)
FILES(	file_specification 1, file_specification 2, file_specification n	)
FILESTEXT(	{*NO} {*ALL} {*NEW} {*UPDATE}	)
FILETYPE(	{*TEXT} {*BINARY} {*EBCDIC} {*FIXTEXT} {*DETECT}	)
FTRAN(	{*INTERNAL} *Member Name*	)
GZIP(	{*YES} {*NO}	)
IFSCDEPAGE(	{*NO} *Code-page*	)
INCLFILE(	{*NONE} *path/filename*	)
MSGTYPE(	{*PRINT} {*SEND {*BOTH}	)
PASSWORD(	Archive Password	)
SELFXTRACT (	{*MAINTAIN} {*REMOVE} (WINDOWS} {AIX} {HP_UNIX} {SUN_UNIX} {LINUXINTEL}	)
SFUSER (	{*CURRENT} {user id 1} {user id 2} {user id 5)	)
SFQUEUE (	{*ALL} {Library/OUTQ }	)

SFFORM (	{*ALL} {*STD} {Spool File Form Type }	)

SFUSRDTA (	{*ALL} {Spool File User data}	)
SFSTATUS (	{*ALL} {*READY} (*HELD } {*CLOSED} {*SAVED } {*PENDING} {*DEFERRED}	)
SFJOBNAM (	{blanks } {*} {Job-name//Userr-Name/Job Number}	)
SFTARGET (	{*SPLF} {*TEXT} {*TEXT1} {*TEXT2} {*TEXTFC} {*PDF} {*PDFLETTER} {*PDFLEGAL}	)
SFTGFILE (	{*GEN1} {*GEN2} {*GEN1P} {path/filename }	)
STOREPATH(	{*NO} {*YES}	)
SPLFILE (	{*ALL} {Spool File Name }	)
SPLNBR (	{*ALL} {*LAST} {Spool File Number 1-9999}	)
STOREPATH(	{*NO} {*YES}	)
TMPPATH(	{*CURRENT} *Temporary Path*	)
TRAN(	{*INTERNAL} *Member Name*	)
TYPARCHFL(	{*DB} {*IFS}	)
TYPFL2ZP(	{*DB} {*IFS} {*IFS2} {*DBA} {*SPL}	)
TYPLISTFL(	{*DB} {*IFS}	)
VERBOSE(	{*NORMAL} {*NONE}	)

{*ALL} {*MAX}	VPASSWORD(Archive Verify Password)

PKZIP Command Keyword Details

TYPE

**TYPE(ADD|DELETE|EXTRACT|FRESHEN|MOVEA|MOVEF|MOVEU|
UPDATE |VIEW)**

The TYPE keyword specifies the type of action PKZIP should perform on the ZIP archive.

The possible actions are:

***ADD**	The *ADD option is the default and adds a selection of files to the archive file. If an archive is already present, it will be written over by the new archive file.
***UPDATE**	The *UPDATE option updates files which are already in the archive file with a newer version and will also add newly selected files that are not present in the archive file.
***FRESHEN**	The *FRESHEN option updates ONLY the files which already exist in an archive file. If the date/time of the file is newer than the date/time of the file in the archive, the file will be compressed and replace the one in the archive.
***MOVEA**	The *MOVEA (Move and Add option) option performs the *ADD option, and upon completion of a successful PKZIP command, the actual file will be deleted.
***MOVEF**	The *MOVEF (Move and Freshen option) option performs the *FRESHEN option, and upon completion of a successful PKZIP command, the actual file will be deleted.
***MOVEU**	The *MOVEU (Move and Update option) option performs the *UPDATE option, and upon completion of a successful PKZIP command, the actual file will be deleted.
***DELETE**	The *DELETE option removes entries from the archive file based upon the selection of FILES and EXCLUDE parameters. The format of the FILES and EXCLUDE parameters should be in the format of the files as seen in the archive.

ADVCRYPT

ADVCRYPT(ZIPSTD|ASE128|AES192|AES256)

When a ZIP action is requested to save a file in an Archive, and a password is provided, ***PKZIP for iSeries™*** will use an encryption method to protect the data. This command value specifies which algorithm to be
employed.

Possible Encryptions are:

ZIPSTD	This algorithm is the original algorithm used in PKZIP 2.x products and is compatible with other PKZIP 2.04g products that support standard encryption. Unless the installation defaults module has been tailored differently, this is the default value for the product, if you choose to encrypt a file.
AES128	A PKZIP exclusive implementation of the AES 128-bit key algorithm (AES stands for Advanced Encryption Standard; also known as Rijndael) will be used.
AES192	A PKZIP exclusive implementation of the AES 192-bit key algorithm (AES stands for Advanced Encryption Standard; also known as Rijndael) will be used.

AES256 A PKZIP exclusive implementation of the AES 256-bit key algorithm (AES stands for Advanced Encryption Standard; also known as Rijndael) will be used.

Usage Notes:

- PKUNZIP will detect automatically which encryption method was specified during the ZIP process and operate accordingly.
- During a PKZIP (ZIP) run, only 1 encryption method may be specified, and that method will be used for each file that is operated.
- By executing PKZIP at different times, various files within the Archive may be saved with differing levels (and types) of protection. That is, some files may not be protected at all, while others may have different methods and/or passwords.
- A "+" character is shown in a View to indicate Standard Encryption protection is used for a file.
- A "!" character is shown in a View to indicate Advanced Encryption (AES) protection is used for a file.

ARCHIVE

ARCHIVE(Archive Zip File name with path)

Specifies the path/file name or the library/file name of the ***PKZIP for iSeries*** archive to be processed. If the file exists, PKZIP will overwrite the file, otherwise PKZIP will create the file for you. Depending on which file system you choose, the Path or Library must exist. This is a required parameter.

The format depends on whether you will be using the Archive file in the Library File System, or the IFS (Integrated File System).

See parameter TYPARCHFL for file system type information.

Library File System Format is Library/File(Member). If Member is omitted, it will be created with the file name. If the file is not found, it will be created with a default length specified in parameter DFTARCHREC (which has a default of 132). If you would create a file manually to use a larger record length, create it with no members and with parameter MAXMBRS with *NOMAX, or with a high excepted limit. If the Library is not specified, the file name will be searched using *LIBL. If the file name is not found, the file will be created in the users *CURLIB.

If a Library is specified and does not exist, PKZIP will create the Library.

Integrated File System (IFS) Open system path followed by the archive file name. The path and file name can up to 256 characters and may contain imbedded spaces.

ARCHTEXT

ARCHTEXT(*NONE| Archive File Text description)

Specifies text that will be stored in the archive as the archive's file comment.

***NONE** No new archive comment will be stored.

***DEFAULT** The default PKWARE comment will be stored.

***CLEAR** Clear any commnet that may be stored in an archive.

Archive File Text description Up to 255 characters that are stored as the archive's file comments.

COMPAT

COMPAT(*NONE|*PK400)

Specifies that PKZIP will create and store extended data field information in another supported format or previous version. At this time, only "PKZIP Version 4.0 for OS/400" is supported.

The allowable values are:

NONE** The Extended Data fields will be in ***PKZIP for iSeries Versions 5.0 and above formats.

***PK400** The Extended Data fields will output to the archive in the format used by "PKZIP Version 4.0 for OS/400" product. This option should be used if the archive file will be extracted by "PKZIP Version 4.0 for OS/400" and the attributes are required to create the files. The files can be extracted without this option, but the files may have to be manually created in order to have the proper attributes (such as record length and text descriptions).

COMPRESS

COMPRESS(*FAST|*SUPERFAST|*MAX|*STORE|*TERSE)

Specifies the compression method or level to be used during a run of PKZIP. This is used to specify the amount and speed of compression that is required for any updated files.

The allowable values are:

***FAST** This selection provides ample compression at a fast rate.

***SUPERFAST** This is the default selection. This will compress in the fastest time, but will compress the files by the least amount.

***MAX** This level provides the maximum compression possible, but will also take the longest in time to process.

***NORMAL** The normal compression level provides good compression amount at a reasonable speed.

***STORE** No compression. Store will also be used if the other methods tried results in a file larger than the original.

***TERSE** This selection provides a terse compression algorithm provided with the iSeries by IBM as an API. This is much faster but is less efficient than FASTEST, and can only be decompressed on the iSeries. Do not use this option if you wish to unzip

the
archive on another platform.

CRTLIST

CRTLIST(*NONE| *path/filename*)

Specifies that PKZIP will create an output file with a list of entries that would have been compressed based upon the selection criteria in the FILES and EXCLUDE parameters.

See parameter TYPLISTFL for file system type.

***NONE** Default. No list file will be created.

path/filename Enter the file path and name of the file to create. The layout depends on which file system you want to create the file in.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

CVTDATA

CVTDATA(*External Pgm Conversion Extended Data*)

Specifies the extended data that is passed to the external program CVTNAME. When CVTFLAG is not *NONE, the contents of the parameter is passed to provide extended flexibility in controlling how the iSeries names are stored in the archive. See Appendix D - External Name Conversion Program - CVTNAME for more details on the external program.

External Pgm Conversion Extended Data
Specify up to 255 bytes of unedited data which is passed to the exit program CVTNAME to assist in controlling the program logic.

CVTFLAG

CVTFLAG(*NONE|*400| *Conversion Flags*)

Specifies the flags passed to the external program CVTNAME. These are used to control how the iSeries names are stored in the archive. See Appendix D - External Name Conversion Program - CVTNAME for more details on the external program.

***NONE** Conversion exit is not active.

***400** Use the included sample CVTNAME program section that does not change the names.

Conversion Flags Specify a 5-byte flag that is passed to the exit program CVTNAME to control the program logic. If the name passed back is blank, then conversion is referred back to the setting of the CVTTYPE parameter.

CVTTYPE

CVTTYPE(*NONE|*DROP|*SUFFIX)

Specifies how the iSeries Library and File names are stored in the archive. Since the length of the Library name, File name, and member name can each be up to 10 characters, and MS/DOS format requires a maximum of 8 characters with an optional extension, this option allows name compatibility.

The allowable values are:

***SUFFIX**	This forces any iSeries name with more than 8 characters to create a name of 8 characters and a period(.), followed by characters 9 and 10 to be considered an extension to suffix.
***NONE**	This leaves the iSeries name as the archive name.
***DROP**	This forces any iSeries name with more than 8 characters, to drop characters 9 thru 19.

DATEAB

DATEAB(mmddyyyy)

Used with DATETYPE parameter, DATEAB specifies the date to be used to compare with the files latest modification date for file selection. The format is mmddyyyy, where "mm" is a valid month (01-12), "dd" is valid day of the month, and "yyyy" is the four digits of the year (2001).

DATETYPE

DATETYPE(*NO|*BEFORE|*AFTER)

Specifies if PKZIP should select files based upon a file modification date.

The allowable values are:

***NO**	No date selection will take place.
***BEFORE**	Files with a modification date before the date in DATETYPE will be selected.
***AFTER**	Files with a modification date on or after the date in DATETYPE will be selected.

DBSERVICE

DBSERVICE (*NO|*YES)

Specifies if the iSeries special Database extended file attributes describing the database file, fields and keys are to be store in the archives. This will force the option EXTRAFLD(*YES). The database will also be stored in Binary mode. This mode can produce larger archive files.

The allowable values are:

***NO**	Does not store Database extended services attributes.
***YES**	Stores the Database extended service attributes in the archive file and treat non-SAVF as a database.

DFTARCHREC

DFTARCHREC(132|Record Length)

Specifies the record length to use when creating an archive file in the QSYS Library system. If the TYPARCHFL parameter is *DB, and the archive file does not exist, the archive file will be created with the record length specified in this parameter.

Note: A large record length will leave a high residual number if only one byte is use in the last record.

The allowable values are:

132 Default is record length of 132 to match previous versions.

Record Length A decimal number from 50 to 32000.

DELIM

DELIM(CRLF |CR |LF |LFCR)

When compressing a text file (not binary), the DELIM parameter specifies what characters are to be appended at the end of records to serve as delimiters. The delimiter is removed from the record when it is decompressed.

The allowable values are:

CRLF This is the default selection. Specifies for ***PKZIP for iSeries*** to use the default delimiter CR-LF (x'0D0A') at the end of each text record.

CR Appends an ASCII Carriage Return (hex 0D).

LF Appends an ASCII Line Feed character (hex 0A).

LFCR Appends an ASCII Line Feed character (hex 0A0D).

Note that transfers of MS-DOS records uses a CRLF for a delimiter, while UNIX records use a LF.

DIRNAMES

DIRNAMES(*YES|*NO)

Specifies to store directories as an entry. This is valid only for files in IFS.

***YES** Store the directories as entries in the archive.

***NO** Do not store directories as an archive entry.

DIRRECRS

DIRRECRS(*NONE|*FULL|*NAMEONLY)

IFS only. Specifies whether to search recursively through directories for file selection, or only search the current, specified directory.

The allowable values are:

***NONE** Search only the current, specified directory.

***FULL** Search through all directories by starting with the current, specified directory for selected files. If *FULL is used, and * is for file selections, all files found in all directories below the current directory will be selected.

***NAMEONLY** To be considered a hit, the full path and file name must match the selection statements exactly.

ERROPT

ERROPT(*END|*SKIP)

Specifies what action to take if an error occurs while processing (selecting or compressing) a Spool File.

The allowable values are:

***END** The PKZIP will end without completing the compression of the file. The archive is not updated.

***SKIP** The program will skip the file with the input error and continue to process all other files to completion. Message AQZ0022 will be issued at the end to indicate that an error occurred.

EXCLFILE

EXCLFILE(*NONE| *path/filename*)

This parameter specifies the file containing the list of files to be excluded. This can be used with or without the EXCLUDE parameter. See parameter TYPLISTFL for file system type information.

***NONE** No list file will be processed.

path/filename Enter the file path and name of the file to process. The layout depends on which file system you want the file created.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

EXCLUDE

EXCLUDE(file_specification1, file_specification2,... file_specification *n*)

Specifies the files and file specification patterns that will be excluded from the PKZIP run. One or more names can be specified. Each name should be in the OS/400 file system format, such as, QSYS is library/file(member) and IFS is directory/file, and can include wildcards "*" and "?."

Note: If TYPE(*VIEW) is being used, then the format for these names is the MS/DOS format.

The PKZIP program can also exclude file specifications by using the list file parameter EXCLFILE with a list of names to exclude.

Please refer to Chapter 4. - File Selection and Name Processing for details of file specification formatting.

The valid parameter values for the FILES keyword are as follows:

'file_specification1'
'file_specification2'
'file_specification n'

EXTRAFLD

EXTRAFLD(*YES|*NO)

Specifies if the basic ***PKZIP for iSeries*** extended file attributes should be stored in the archive. Some basic file attributes are record size, library text description, File text description, etc.

The allowable values are:

***YES**	Store the basic normal iSeries file attributes. See Appendix L - Extended Attributes for more definitions. This is the default and will be the same as coding *BOTH.
***NO**	Do not store any extended attributes.
***CENTRAL**	Stores the basic normal iSeries file attributes in only the Archive's Central Directory. This will reduce the overall archive size by only storing the attributes in the Central.
***LOCAL**	Stores the basic normal iSeries file attributes in only the Archive's Local Directory. *Warning:* PKUNZIP only utilizes the Central Directory for extended data attributes.
***BOTH**	Stores the basic normal iSeries file attributes in both the Archive's Central Directory and Local Directory.

Migration consideration: if the archive will be processed by an earlier release of PKZIP for OS/400™ and the attributes are required, then *BOTH should be coded.

FILES

FILES(file_specification1, file_specification2,... file_specification *n*)

Specifies the files and file specification patterns that will be selected in the PKZIP process. One or more names can be specified. Each name should be in the OS/400 file system format, such as, QSYS is library/file(member) and IFS is directory/file, and can include wildcard "*" and "?." For the IFS , the path and file name can up to 256 characters and may contain imbedded spaces.

Note: If TYPE(*VIEW) or TYPE(*DELETE) is being used, then the format for these names is the MS/DOS format.

The PKZIP program can also have file specifications selections to include by using the list file parameter INCLFILE with a list of names to select.

Files may also be excluded. See the EXCLUDE parameter.

Please refer to Chapter 4. - File Selection Process and Name Processing for details of file specification formatting.

The valid parameter values for the FILES keyword are as follows:

'file_specification1'
'file_specification2'...
'file_specification n'

FILESTEXT

FILESTEXT(*NO|*ALL|*NEW|*UPDATE)

Specifies if PKZIP allows the editing (and the type of editing performed) of a file's text comments that are stored in an archive.

The allowable values are:

***NO**	No comment editing (the default).
***ALL**	Add comments or edit comments for all files in the archive.
***NEW**	Add comments only for new files that are added to the archive.
***UPDATE**	Add or edit the comments of files that are added, updated, or freshened in the archive. Only file comments of files that are affected by a change are eligible for editing.

FILETYPE

FILETYPE(*BINARY|*DETECT|*EBCDIC|*FIXTEXT|*TEXT)

Specifies whether the files selected are treated as text or binary data. For text files added to an archive, trailing spaces in each line are removed, the text is converted to ASCII (based on the translation tables) by default, and a carriage return and line feed (CR/LF) are added to each line before the data is compressed into the archive. Binary files are not converted.

The default is *DETECT; where PKZIP attempts to make a determination based on the existing data type. The program will read in a portion of the data, evaluate it, and determine the appropriate process.

Note: This will lower performance time. A message will display the type used when compressing.

Use of text file options is usually faster because PKZIP has to process less data than with *BINARY, but more processing may also take place to perform the translation.

If the file is a SAVF or a Database File (with DBSERVICE(*YES)), then the file will be processed as BINARY, no matter what option is specified.

***BINARY**	Specifies that the files selected are binary files and no translation should be performed.
***DETECT**	The PKZIP program will try to determine the data type of Text or Binary.
***EBCDIC**	Specifies that the files selected are text files and leaves it in EBCDIC without performing any translation. This is good only if the files are to be used on an iSeries or IBM-type mainframe. If they are unzipped to a PC file, then a translation from EBCDIC to ASCII would be required.
***FIXTEXT**	Specifies that the files selected are text files with a fixed record length based on the iSeries file's record length and translation will be performed using the translate tables specified in the TRAN option. This means the compressed file will contain records with trailing spaces followed by a CR and LF. This is only valid for QSYS Library file types as files in the IFS do not contain a record length.
***TEXT**	Specifies that the files selected are text files and translation will be performed using the translate tables specified in the TRAN option.

FTRAN

FTRAN(*INTERNAL| *Member Name*)

Specifies the translation table for use in translating "File names, comments, and password" from the iSeries EBCDIC character set to the character set used in the archive file (normally ASCII character set). A default internal table is predefined. See Appendix F - Translation Tables for additional information.

***INTERNAL** Use the predefined internal table for translation. Using this is initially faster because PKZIP does not have to parse the override table.

membername Specify the member name in the file PKZTABLES that will be parsed and used to translate "File names and comments" files to the archive character set. The member should have the exact format of member UKASCII in file PKZTABLES. See Appendix F - Translation Tables for defining translation tables.

GZIP

GZIP(*YES|*NO)

If this option is set to *YES, PKZIP will create a compressed archive in the GZIP format. The GZIP format only allows for one file or member per archive and all text data is stored in ISO 8859- 1 (LATIN- 1) character set. The GZIP format is very different from the ***PKZIP for iSeries*** archive format, a program that can process ***PKZIP***® archives will not necessarily process a GZIP Archive correctly. The GZIP Archive created conforms to the GZIP specifications RFC1951 and RFC1952.

Do not use this option if the archive is to be unzipped on another platform where GZIP compatibility is not confirmed.

The allowable values are:

***YES** The PKZIP program will create a compressed archive in the GZIP format.

NO** The PKZIP program will create an archive in the ***PKZIP® format. This is the default.

IFSCDEPAGE

IFSCDEPAGE(*NO| *Code-Page*)

If this option is set to *NO, PKZIP will read IFS files using the code page that is registered for the file. Otherwise, PKZIP will read IFS files with the specified code page.

This parameter also controls the Spool File ASCII conversion for *TEST and *PDF documents. When *NO is specified for Spool Files, the conversion will use code page 819.

The allowable values are:

***NO** The PKZIP program will read IFS files with the code page registered for the file. If the file is a Spool Files, the code page 819 will be used. This is the default.

Code-Page The PKZIP program will read IFS files with the specified code page value. If the file is a spool file, it is the code page that a spool file will use for ASCII translation.

INCLFILE

INCLFILE(*NONE| *path/filename*)

This parameter specifies the file containing the list of files to be selected for inclusion. This can be used with or without the FILES parameter. See parameter TYPLISTFL for file system type information.

***NONE** No Include list file will be processed. This is the default.

path/filename Enter the file path and name of the file to process. The layout depends on which file system you want to create the file in.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

MSGTYPE

MSGTYPE(*PRINT|*SEND|*BOTH)

Specifies where the display of messages and information should be shown. The PKZIP program has the ability to send messages that appear on the log and/or the ability to print to stdout and stderr. If working interactively, stdout and stderr will show upon the dynamic screen. If submitted via batch, you can override them to print in an OUTQ or build a CL and save them to an outfile.

***SEND**	Send the information to the log with send message commands.
***PRINT**	Send the information to stdout and stderr.
***BOTH**	Send the information to the log with send message commands and also to stdout and stderr.

PASSWORD

PASSWORD(Archive Password)

Specifies a password for files being added to an archive. This password may be up to 64 characters in length and is case sensitive. All files selected for archiving will be encrypted using the specified password.

Note: There is no way to extract the password used from the archive data. If the password is forgotten, the file will become inaccessible. If files in an archive need to have different passwords, PKZIP must be run for each password required.

Since the password is entered in EBCDIC, the translation table referenced in the FTRAN parameter is used to translate it to ASCII. Care should be take when using the FTRAN override and when using a password. To use password-protected files, the same FTRAN override option is required.

SELFXTRACT

SELFXTRACT (*MAINTAIN| WINDOWS| UNIX| LINUX| *REMOVE)

This licensed feature specifies the action to take concerning Self Extracting Archives. The actions are to maintain the current archive as is, create the new archive with a self extracting preamble, or to remove the self extracting preamble if one exist in the archive.

The self extracting programs are held as binary entities in the PKZIP for iSeries library in the file PKZIPSFX. The appropriate member is loaded and the executable data copied to the beginning of the Archive as a preamble when requested.

The resulting Archive can still be processed by PKZIP for iSeries as a normal ZIP Archive.

The allowable values are:

MAINTAIN	If the current archive contains a self extracting preamble, it will be maintained at the beginning of the updated archive.
WINDOWS	The Windows version of the self extract preamble will be installed to archive. (`Microsoft Windows (95 and above)`)
AIX	The AIX version of the self extract preamble will be installed to archive. (`IBM AIX Version 4.0 and above`)
HP_UNIX	The HP Unix version of the self extract preamble will be installed to archive. (`HP/UX Version 9.0 and above`)

SUN_UNIX The Sun Unix version of the self extract preamble will be installed to archive. (Sun `Solaris 2.3 (SunOS 53) and above`)

LINUXINTEL The Linux version of the self extract preamble (if available) will be installed to archive. (`LINUX Kernel 2.x for Intel Note:libc-5 must be installed on the target system.`)

***REMOVE** If the current archive contains a self extracting preamble, it will be removed.

SFUSER

SFUSER (*CURRENT|*ALL |User Name List)

Specifies the user names that created spool files that will be selected. This value is ignored if SFJOBNAM is coded.

The allowable values are:

***CURRENT** Only files created by the user running this command are selected.

***ALL** Files created by all users are selected.

User Name Specify up to 10 user names. Only files created by those users are selected.

SFQUEUE

SFQUEUE (*ALL |Name)

Specifies the Output Queue that will be searched for the spool file selections. If no OUTQ Library is specified, it will default to *LIBL.

The allowable values are:

***ALL** Files on any device-created or user-created output queue are selected.

OUTQ The OUTQ that will be searched.

OUTQ Library The library where the OUTQ resides. Defaults to *LIBL.

SFFORM

SFFORM (*ALL | *STD| Form Type)

Specifies the spool file form type that is on the spool files that will be selected.

The allowable values are:

***ALL** Files for all form types are selected.

***STD** Only files that specify the standard form type are selected.

Form Type Only spool files with this specific form type will be selected.

SFUSRDTA

SFUSRDTA (*ALL| User Data)

The user data tag associated with the spool file to select.

The allowable values are:

***ALL**	Files with any user data tag specified are selected.
User Data	Only spool files with this specific user data tag will be selected.

SFSTATUS

SFSTATUS (*ALL |*READY|*HELD|*CLOSED|*SAVED|*PENDING|*DEFERRED)

Specifies the statuses of the spool files to be selected. Up to 4 statuses can be selected for one run.

The allowable values are:

***ALL**	All Spool File status will be considered for selection.
***READY**	Only spool files with a status of *READY will be selected.
***HELD**	Only spool files with a status of *HELD will be selected.
***CLOSED**	Only spool files with a status of *CLOSED will be selected.
***SAVED**	Only spool files with a status of *SAVED will be selected.
***PENDING**	Only spool files with a status of *PENDING will be selected.
***DEFERRED**	Only spool files with a status of *DEFERRED will be selected.

SFJOBNAM

SFJOBNAM(Blank|*|Spool File Jobname/User/Job Number)

Specifies the job name, user name, and job number that will be used to select spool files. If anything other than blanks are in SFJOBNAM parameter, it will be used as the primary selection criteria. If any of the three fields (Job Name, User Name, and Job Number) are specified, then all three fields must be entered and be valid.

The allowable values are:

Blank	This is the default selection. This will cause all other selection criteria to be used for spool files.
*****	The * will cause the current Job-name/User-Name/Job Number to be used to select spool files.
Job-name	Specify the name of the job to be selected. If no job qualifier is given, all of the jobs currently in the system are searched for the simple job name.
User-Name	Specify the name that identifies the user profile under which the job is run.
Job-Number	Specify the job number assigned by the system.

SFTARGET

SFTARGET (*SPLF|*TEXT|*PDF|*TEXT1|*TEXT2)

Specifies the format of the file that will be stored in the archive.

The allowable values are:

***SPLF**	This is the default selection. This will compress the spool file in a spool file format with all of the spool file attributes. This format is only valid on an AS/400. If is

extracted it will take on the latest spool file settings, such as, Job Name, User, Job Number, Spool File number etc. The suffix for this selection is SPLF.

***TEXT** The spool file will be saved in the archived as an ASCII Text document. The suffix for this selection is .TXT. Each New page will have a Form Feed control character.

***TEXT1** The spool file will be saved in the archived as an ASCII Text document. The suffix for this selection is .TXT. Each New page will have a Carriage Control and Line Feed control characters.

***TEXT2** The spool file will be saved in the archived as an ASCII Text document. The suffix for this selection is .TXT. Each page will have a Carriage Control and Line Feed control characters for blanks lines to fill out a page with the number lines required by the spool file attribute.

***TEXTFC** The spool file will be saved in the archived as an ASCII Text document. The suffix for this selection is .TXT. Each New page will have a Form Feed control character.

***PDF** The spool file will be saved in the archived as a PDF Text document. The suffix for this selection is .PDF. The size will be adjusted based upon the width and length of the Spool File.

***PDFLETTER** The spool file will be saved in the archived as a PDF Text document. The suffix for this selection is .PDF. The size will be adjusted based upon the width and length of the Spool File.

***PDFLEGAL** The spool file will be saved in the archived as a PDF Text document. The suffix for this selection is .PDF. The size will be adjusted based upon the width and length of the Spool File.

SFTGFILE

SFTGFILE (|*GEN1|*GEN2|*GEN1P|File Name)

Specifies the how the file name will be stored in the archive.

The allowable values are:

***GEN1** GEN1 is the default selection. This generate a very specific name using most of the Spool File name attributes to form the file name so that it will not be duplicated. The name will be built as follows:

"Job-Name**/**User-Name**/#**Job-Number**/**Spool-File-Name/Fspool-File-Number.Suffix"

"MYJOB/BILLS#152681/INVOICE/F0021.SPLF"

The suffix is dependent on the SFTARGET setting.

***GEN1P** *****GEN1P generates the same file name as *GEN1 except instead of a '/' separator, *GEN1P will use a '.' as name separator.

***GEN2** *****GEN2 uses the spool file name and appends the spool file number followed by the suffix that is depended on the SFTARGET setting. Caution should be taken in that a duplicate file name in the archive could be created. An example of GEN2 is a spool file INVOICE with Spool File number of 21 that will be converted to a text file will generate a file name of INVOICE21.TXT.

File Name This parameter should only be used when selecting one specific spool file where you want a specific file name.

SPLFILE

SPLFILE (*ALL| Spool File Name)

Specifies the Spool File Name that will be selected. This parameter is used along with all the other Spool File selection parameters to determine the spool files to select.

The allowable values are:

***ALL**	This is the default setting. *ALL indicates that spool file name is not important.
Spool File Name	A very specific Spool File Name that will be searched for and selected.

SPLNBR

SPLFILE (*ALL|*LAST| Spool File Number)

Specifies the number of the spooled file, from the job whose data records are to be selected. If *ALL is coded then all file numbers are considered. This parameter is only valid when the SFJOBNAM parameter or SPLFILE are used. This parameter is used along with all the other Spool File selection parameters to determine the spool files to select.

The allowable values are:

***ALL**	This is the default setting. *ALL indicates that spool file number is not important.
***LAST**	The spooled file with the highest number is used.
Spool File Number	A number 1-9999 to specify the number of the spooled file whose data records are to be selected.

STOREPATH

STOREPATH(*NO|*YES)

Specifies whether to store the full path and file name in the archive, or to just save the file name. If the file is an IFS file type, the path is all directories, from the current directory, to the directory of the file. In the Library System, the path is the Library and the File name. The Member name is considered to be the archive name.

The allowable values are:

***YES**	Store all paths and the filename in the PKZIP archive.
***NO**	Store only the filename in the PKZIP archive.

TMPPATH

TMPPATH(*CURRENT| *pathname*)

Specifies a directory or Library/File in which to build the temporary archive file. While PKZIP is compressing data into an archive, a temporary archive file name is used. The temporary file name is a 10-character name with a prefix of “PZ” followed by a time stamp (PZttttttt). If this option is *CURRENT, the temporary file is built in the same directory (for Library file systems it is same Library/File with temp member) in which the new archive will be stored and is then renamed at the end of the run to the archive name. If an override path is specified, the temporary archive file is built into that specified path, and the file is then copied to its final archive path at the end of the run. The temporary file name and path type will be the same as specified

for ARCHIVE. See parameter TYPARCHFL for file system type information. Special libraries (such as QTEMP) are used frequently.

***CURRENT** Specifies that the current archive path will be used (see ARCHIVE) to build the temporary archive file PZxxxxxxxx.

pathname Specifies a path name (if using IFS such as /PKZIP/tempdir) or a Library/File (if using the Library System).

NOTE 1: When using the QSYS library file system and specifying "qtemp" as the TMPPATH, a dynamic file name and member name is created in the library qtemp. At the end of the run, the file and member are removed. If any other combination of names is used, then a dynamic member name is created and only the member is removed.

NOTE 2: When using the QSYS library file system and specifying a TMPPATH, there may be a slight performance degradation because the archive file will have to be copied from one library/file to another library/file. Otherwise, if *CURRENT is used, the file member name will only be renamed.

TRAN

TRAN(*INTERNAL| *Member Name*)

Specifies the translation table for use with translating **data** from the iSeries EBCDIC character set to the character set used in the archive file (normally the ASCII character set). A default internal table is predefined (see Appendix F - Translation Tables).

***INTERNAL** Use the predefined internal table for translation. Using this is initially faster because PKZIP does not have to parse the override table.

Member Name Specifies the member name in the file PKZTABLES that will be parsed and used to translate data files to the archive character set. The member should have the exact format of member UKASCII in file PKZTABLES (see Appendix F - Translation Tables for defining translation tables).

TYPARCHFL

TYPARCHFL(*DB|*IFS)

Specifies the type of file system in which the archive file will exist (see parameters ARCHIVE and TMPPATH for additional information).

***DB** Archive files are to be in the QSYS Library file system.

***IFS** Archive files are to be in the Integrated Files System (IFS).

TYPFL2ZP

TYPFL2ZP(*DB|*IFS)

Specifies the type of file system that contains files to be zipped. Reflected for files in parameters FILES and EXCLUDE.

***DB** Files to be zipped are in the QSYS Library file system.

***IFS** Files to be zipped are in the IFS (Integrated Files System) -Case Sensitive selection.

***IFS2** Files to be zipped are in the IFS (Integrated Files System) –NON Case Sensitive selection.

***DBA** Files to be compressed are database files in the QSYS Library file system with Database Mode "DBSERVICE(*YES)", and the records are to processed in arrival sequence. This is only pertinent for Database files containing keys and when it is important to retain the arrival sequence of the data.

***SPL** Files to be zipped are Spool Files.

TYPLISTFL

TYPLISTFL(*DB|*IFS)

Specifies the “type of files system” that will be used for the input list file and/or the output list file of selected items.

To use input list files, see parameters INCLFILE (file section list) or EXCLFILE (file exclude list). To create an output list file of the selected file items, see parameter CRTLIST.

***DB** Files are in the QSYS Library file system.

***IFS** Files are in the IFS(Integrated Files System).

VERBOSE

VERBOSE(*NORMAL|*NONE| *ALL|*MAX)

Specifies how the detail will be displayed during a PKZIP run.

The allowable values are:

***NORMAL** Displays most informative message to show PKZIP is processing.

***NONE** Displays only major exception information.

***ALL** Displays all messages.

***MAX** Used only for debugging purposes.

VPASSWORD

VPASSWORD(Archive Verify Password)

Specifies a verification password against the entered password since the PASSWORD is not visible. This parameter is required for all encryption methods except ZIPSTD. VPASSWORD follows all the rules of PASSWORD and must match exactly to the archive password entered in PASSWORD parameter or the run will be terminated.

Chapter 10. - PKUNZIP Commands

PKUNZIP Command Summary with Parameter Keyword Format

If the OS/400 command prompt screen is to be used, the command format is simply: PKUNZIP.

The command prompt screen is displayed when ENTER or PF4 is pressed. The parameter keywords are displayed on this screen together with the available keyword options. The required options can be selected before PF4 is pressed to accept the selections. If the command and parameter keywords are entered together on the command line, the required format is:

PKUNZIP keyword1(option) keyword2(option) . . . keyword*n*(option)

Keywords are marked by spaces. The keyword “ARCHIVE” is the only positional keyword where the keyword is not required. Whenever the word “path” is used, its meaning depends on the file system that is being used. If IFS is used, path refers to the openness true path type. If the Library systems or *DB is used, path means Library/file, and then the file name refers to the member name.

TYPE(	*VIEW (*EXTRACT} {*NEWER} {*TEST}	)
ARCHIVE(	*Archive Zip File name* with path	)
CRTLIST(	{*NONE} *path/filename*	)
CVTDATA(	*External Pgm Conversion Extended Data*)	
CVTFLAG(	{*NONE} {*400} *External Pgm Conversion Flags*	)
CVTTYPE(	{*NONE} {*DROP} {*SUFFIX}	)
DFTDBRECLN(	{132} {decimal number}	)
DROPPATH(	{*NONE} {*ALL} {*LIB}	)
EXCLFILE(	{*NONE} *path/filename*	)
EXCLUDE(	*file_specification1,* *file_specification2,* *file_specificationn*	)
EXDIR(	{*CURRENT} *path*	)

FILES(	*file_specification1,* *file_specification2,* *file_specificationn*	)
FILETYPE(	{*TEXT} {*BINARY} {*EBCDIC} {*DETECT}	)
FTRAN(	{*INTERNAL} *Member Name*	)
IFSCDEPAGE(	{*NO} *Code-page*	)
INCLFILE(	{*NONE} *path/filename*	)
MSGTYPE(	{*PRINT} {*SEND} {*BOTH}	)
OVERWRITE(	{*NO} {*YES} {*PROMPT}	)
PASSWORD(	*Archive Password*	)
SFQUEUE (	{*DFT} {Library/Outq }SPLUSRID (	)
SPLUSRID	{*DFT} {User ID }	)
TRAN(	{*INTERNAL} *Member Name*	)
TYPARCHFL(	{*DB} {*IFS}	)

TYPFL2ZP(	{*DB}	)
	{*IFS}	
TYPLISTFL(	{*DB}	)
	{*IFS}	
VERBOSE(	{*NORMAL}	)
	{*NONE}	
	{*ALL}	
	{*MAX}	
VIEWOPT(	{*NORMAL}	)
	{*DETAIL}	
	{*BRIEF}	
	{*COMMENT}	

VIEWSORT(	{*ASIS}	)
	{*DATE}	
	{*DATER}	
	{*NAME}	
	{*NAMER}	
	{*PERCENT}	
	{*PERCENTR}	
	{*SIZE}	
	{*SIZER}	

PKUNZIP Command Keyword Details

TYPE

TYPE(*EXTRACT|*NEWER|*TEST |<u>*VIEW</u>)

The TYPE keyword specifies the type of action PKUNZIP should perform on the ZIP archive.

The possible actions are:

<u>*VIEW</u> Will output information about all files or selected files contained in an archive. This option is performed using PKUNZIP. The sequence (see *VIEWSORT) and type of list (*VIEWOPT) determines what information is displayed.

***EXTRACT** Extracts files from the archive (please refer to the DROPPATH, CVTTYPE, TO, and EXDIR parameters for controlling the conversion of file names extracted from the archive).

***NEWER** Extracts files in the archive that have a more recent date and time than the corresponding file on disk. If the files do not exist on disk, they will be extracted as newer. All other files will be skipped.

***TEST** Tests the integrity of files in the archive by extracting files without writing the data. As each file is extracted, a CRC is calculated. At the end of the file the calculated CRC is compared against the stored CRC in the archive file header to confirm that the data has not been corrupted.

ARCHIVE

ARCHIVE(Archive Zip File name with path)

Specifies the path/file name or the library/file name of the PKUNZIP archive to be processed.

This is a required parameter.

The format depends on whether you will be using the Archive file in the Library File System or the IFS (Integrated File System).

See parameter TYPARCHFL for file system type information.

Library File System: Format is Library/File(Member). If Member is omitted, it will use the file name for the member.

Integrated File System (IFS): Open system path followed by the archive file name. The path and file name can up to 256 characters and may contain imbedded spaces.

CRTLIST

CRTLIST(<u>*NONE</u>| *path/filename*)

Specifies that PKUNZIP will create an output file with a list of entries that will be compressed based upon the selection criteria in the FILES and EXCLUDE parameters. This parameter only works with the TYPE set to *VIEW.

See parameter TYPLISTFL for file system type information.

<u>*NONE</u> No list file will be created.

path/filename Enter the file path and name of the file to create. The layout depends on which file system you want to create the file in.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

CVTDATA

CVTDATA(*External Pgm Conversion Extended Data*)

Specifies the extended data that is passed to the external program CVTNAME. When CVTFLAG is not *NONE, the contents of the parameter is passed to provide extended flexibility in controlling how the iSeries names are stored in the archive. See Appendix D - External Name Conversion Program for more details on the external program.

External Pgm Conversion Extended Data
Specify up to 255 bytes of unedited data which is passed to the exit program CVTNAME to assist in controlling the program logic.

CVTFLAG

CVTFLAG(*NONE|*400|*Conversion Flags*)

Specifies the flags passed to the external program CVTNAME. These are used to control how the iSeries names are stored in the archive. See Appendix D - External Name Conversion Program CVTNAME for more details on the external program.

The allowable values are:

***NONE** Conversion exit is not active.

***400** Use the included sample CVTNAME program section that does not change the names.

Conversion Flags Specify a 5-byte flag that is passed to the exit program CVTNAME to control the program logic. If the name passed back is blank, then conversion is referred back to the setting of the CVTTYPE parameter.

CVTTYPE

CVTTYPE(*NONE|*DROP|*SUFFIX)

Specifies how the files names in the archive will be converted to a file name in the iSeries Library, File, and Member format. In the iSeries QSYS Library system, the length of each name in the QSYS format can only be up to 10 characters. In other platforms, the file name formats (including MS/DOS) may have an extension with a period (.) separator which is not valid in the iSeries DB Name. The file names in some cases may even exceed the 10-character limit. This parameter gives control over the file name conversion process.

Note: The conversion of file names may result in duplicate file names on the iSeries system. In this case, the rules for overwriting the files are in effect for duplicates (see the OVERWRITE option). If this is the case, using specific file inclusion and exclusion with multiple runs may be required to extract all of the files.

The allowable values are:

***SUFFIX** This forces the removal of the period(.) extension and stores name truncating characters over 10 characters.

***NAMEFILE** The extensions are considered to be file names or treated as a slash (/).

***DROP** Drops all characters after the period(.) extension separator, and stores the name truncating characters over 10.

DFTDBRECLN

DFTDBRECLN (132|Record Length)

Specifies the record length to use when creating a file in the QSYS Library system. If TYPFL2ZP parameter is *DB, and the file being extracted does not exist nor does extended attribute for the record length exist, the file will be created with the record length specified in this parameter.

The allowable values are:

132 Default is record length of 132 to match previous versions.

Record Length A decimal number from 50 to 32000.

DROPPATH

DROPPATH(*NONE|{*ALL| *LIB)

Used to drop the path(s) or libraries of files that are stored in the archives, therefore only using the file names in the archive. This is used along with the keyword EXDIR where the default paths are defined when dropping the paths on files in the archive.

For example, if the file in the archive is “path1/path2/filename” (IFS) or “Library/File/member” (QSYS), and if DROPPATH is *ALL, the file being extracted would be “filename” or “member”. If *LIB was used, the file being extracted would be path1/filename” or “File/member”.

See Example 1 - PKUNZIP Files to a New or Different Library in Appendix B - Examples for an example of using EXDIR and DROPPATH together.

The allowable values are:

***NONE** Do not remove paths and/or libraries in the archive.

***ALL** Remove all paths that are stored in the archive, leaving only an IFS file name or member name.

***LIB** Remove only the first path (which in most cases could be the library).

EXCLFILE

EXCLFILE(*NONE| *path/filename*)

This parameter specifies the file containing the list of files to be excluded. This can be used with or without the EXCLUDE parameter. See parameter TYPLISTFL for file system type information.

***NONE** No list file will be processed.

path/filename Enter the file path and the name of the file to process. The layout depends on which file system you want the file created.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

EXCLUDE

EXCLUDE(file_specification1, file_specification2,... file_specification *n*)

Specifies the files and file specification patterns that will be excluded from the PKUNZIP run. One or more names can be specified. Each name should be in the OS/400 file system format, such as, QSYS is library/file(member) and IFS is directory/file, and can include wildcards "*" and "?".

Note: If TYPE(*VIEW) is being used, then the format for these names is the MS/DOS format.

The PKUNZIP program can also exclude file specifications by using the list file parameter EXCLFILE with a list of names to exclude.

Please refer to Chapter 4. - File Selection and Name Processing for details of file specification formatting.

The valid parameter values for the FILES keyword are as follows:

'file_specification1'
'file_specification2'...
'file_specification n'

EXDIR

EXDIR(*CURRENT| *path*)

If there are no paths stored in the archive file name, EXDIR specifies the default path to store the files being extracted. The path definition depends on the "file system type" in parameter TYPFL2ZP. This will happen when the files come from a PC or if the files were compressed with ***PKZIP for iSeries*** using the STOREPATH(*NO) parameter.

If the "file system type" is IFS, EXDIR will be the paths defined for your iSeries open systems and the default path will be the current directory settings (issue the command DSPCURDIR to see the current directory settings).

If the "file system type" is the Library File System, the path will be either a Library or a Library/Filename. The default is *CURLIB/UNZIPPED and if the file UNZIPPED does not exist, then it is created with a record length of 132. It is best to create a default file with the record length of your choice, because if a text file is extracted with a record length greater than the file's record length, the record will be truncated to fit the record length.

If EXDIR is coded with keyword ?MBR and the file system is the QSYS Library system PKUNZIP, will use the member name for the file name. For example: EXDIR('newlib/?MBR') and DROPPATH(*ALL) parameters are coded and the file name in archive is "mylib/myfile/mymbr", the file will be extract to the file "newlib/mymbr(mymbr)". This is only valid for TYPFL2ZP(*DB) files.

EXDIR is also used when the archive file is a GZIP Archive and there is no file name stored in the archive. In this case, EXDIR becomes a required field.

***CURRENT** Current directory for IFS or *CURLIB/UNZIPPED for the QSYS Library file system.

path Enter the path or path/path/.. in which to extract. The layout depends on the file system in which the file is to be created.

Library File System: The format is "Library/File".

Integrated File System (IFS): The format is "path1/path2/../pathn".

FILES

FILES(file_specification1, file_specification2,... file_specification *n*)

Specifies the files and file specification patterns that will be selected in the PKUNZIP process. One or more names can be specified. Each name should be in the OS/400 file system format, such as, QSYS is library/file(member), and IFS is directory/file, and can include wildcard "*" and "?".

Note: If TYPE(*VIEW) is being used then the format for these names is the MS/DOS format.

The PKUNZIP program can also have file specification selections to include by using the list file parameter INCLFILE with a list of names to select.

Files may also be excluded. See EXCLUDE parameter.

Please refer to Chapter 4. - File Selection and Name Processing for details of file specification formatting.

The valid parameter values for the FILES keyword are as follows:

'file_specification1'
'file_specification2'
'file_specification n'

FILETYPE

FILETYPE(*TEXT|*BINARY|*EBCDIC|*DETECT)

Specifies whether the files selected are treated as text or binary data. For text files added to an archive, trailing spaces in each line are removed, the text is converted to ASCII (based on the translation tables) by default, and a carriage return and line feed (CR/LF) are added to each line before the data is compressed into the archive. Binary files are not converted at all.

There are attributes which indicate how a file was compressed (TEXT, BINARY, or a SAVF) in the Archive headers. The default setting (and recommended) is *DETECT, which analyzes the header to determine the file type. To view the attribute settings of a file, use the VIEWOPT(*DETECT).

If the file is a SAVF, then it will be processed as BINARY, regardless of any option that you select.

***DETECT**	Uses the attribute setting that is stored in the archive to determine the file type.
***TEXT**	Specifies that the files selected are text files and translation will be performed using the translate tables specified in the TRAN option.
***BINARY**	Specifies that the files selected are binary files and no translation should be performed.
***EBCDIC**	Specifies that the files selected are text files and leaves it in EBCDIC without performing any translation. This is good only if the files are to be used on an iSeries or IBM-type mainframe. If they are unzipped to a PC file, then a translation from EBCDIC to ASCII is required.

FTRAN

FTRAN(*INTERNAL| *Member Name*)

Specifies the translation table for use with translating "File names, comments, and password" from the iSeries EBCDIC character set to the character set used in the archive file (normally ASCII character set). A default internal table is predefined. See Appendix F - Translation Tables for additional information.

***INTERNAL** Use the predefined internal table for translation. Using this is initially faster because PKUNZIP does not have to parse the override table.

membername Specify the member name in the file PKZTABLES that will be parsed and used to translate "File names and comments" files to the archive character set. The member should have the exact format of member UKASCII in file PKZTABLES. See Appendix F - Translation Tables for defining translation tables.

IFSCDEPAGE

IFSCDEPAGE(*NO | *Code-Page*)

If this option is set to *NO, PKUNZIP will write IFS files with the code page that is registered for the file, or will use the default job code page if no code page is set in the file attributes. Otherwise, PKUNZIP will write IFS files with the specified code page.

Note: If files are to be extracted to a case sensitive file system, the case sensitive format of file names must be used before they can be selected.

The allowable values are:

***NO** The PKUNZIP program will read IFS files with the code page registered for the file. This is the default.

Code-Page The PKUNZIP program will write the IFS files with the specified code page value.

INCLFILE

INCLFILE(*NONE| *path/filename*)

This parameter specifies the file containing the list of files to be selected for including. This can be used with or without the FILES parameter. See parameter TYPLISTFL for file system type information.

***NONE** No Include list file will be processed. This is the default.

path/filename Enter the file path and name of the file to process. The layout depends on which file system you want the file created.

Library File System: The format is "Library/File(Mbr)".

Integrated File System (IFS): The format is "path1/path2/../pathn/filename".

MSGTYPE

MSGTYPE(*PRINT|*SEND|*BOTH)

Specifies where the display of messages and information should be shown. The PKUNZIP program can send messages which appear on the log, and also may print to stdout and stderr. If working interactively, stdout and stderr will display upon the dynamic screen. If submitted via batch, you can override them to print in an OUTQ, or you can build a CL and save them to an outfile.

***SEND** Send the information to the log with send message commands.

***PRINT** Send the information to stdout and stderr.

***BOTH** Send the information to the log with send message commands and also to stdout and stderr.

OVERWRITE

OVERWRITE(*NO|*YES|*PROMPT}

Controls how PKUNZIP reacts to files that are being extracted and the file already exists. To help prevent accidental overwriting of files, the default is *NO.

The allowable values are:

***YES** Always overwrite files. If the file exist, the file will be overwritten with no message or prompting.

***NO** Never overwrite files. If the file already exists then the archive file will be skipped and not extracted. This is the default.

***PROMPT** When a file being extracted already exists, PKUNZIP will issue the warning message AQZ0262 and prompt the user for the required action.

PASSWORD

PASSWORD(Archive Password)

Specifies a password to be used for files that were added to the archive with a password. This password may be up to 64 characters in length and is case-sensitive. All files selected for archiving will be checked for encryption using the specified password. Files in the archive may have different passwords. If so, PKUNZIP must be run once for each password.

Since the password in entered in EBCDIC, the translation table referenced in the FTRAN parameter is used to translate it to ASCII. Care should be take when using the FTRAN override and when using a password. To use password-protected files, the same FTRAN override option is required.

SFQUEUE

SFQUEUE (*DFT |Name)

Specifies the Output Queue that will be used as an override when extracting spool files. If no OUTQ Library is specified, it will default to *LIBL.

The allowable values are:

***DFT** The Output Queue that are in the spool file attributes will be used when extracting files.

OUTQ The specific OUTQ that will used when the spool file is extracted. It must be a valid Output Queue.

OUTQ Library The library where the OUTQ resides.

SPLUSRID

SPLUSRID (*DFT| User ID)

The user ID to use when extracting a Spool File. If *DFT is used the User ID belonging to the spool file will be used when building the spool file.

The allowable values are:

***DFT** Use User Id associated with spool file in the archive.

User ID Specify a valid user ID that the new extracted spool file will belong to. It must be a valid User ID on the OS/400.

Note on extracting Spool Files: To create or extract spool file with PKUNZIP, the user must have *USE authority to the API QSPCRTSP. The normal setting for the API QSPCRTSP is Authority PUBLIC(*EXCLUDE). The API authority is set this way so that system administrators can control the use of this API. This API has security implications because you can create a spooled file from the data of another spooled file. To allow user to extract spool files change the API authority on a need basis.

TRAN

TRAN(*INTERNAL| *Member Name*)

Specifies the translation table for use with translating **data** from the iSeries EBCDIC character set to the character set used in the archive file (normally the ASCII character set). A default internal table is predefined (see Appendix F - Translation Tables).

***INTERNAL** Use the predefined internal table for translation. Using this is initially faster because PKUNZIP does not have to parse the override table.

Member Name Specifies the member name in the file PKZTABLES that will be parsed and used to translate data files to the archive character set. The member should have the exact format of member UKASCII in file PKZTABLES (see Appendix F - Translation Tables for defining translation tables).

TYPARCHFL

TYPARCHFL(*DB|*IFS)

Specifies the type of file system in which the archive file will exist (see parameters ARCHIVE and TMPPATH for additional information).

***DB** Archive files are to be in the QSYS Library file system.

***IFS** Archive files are to be in the Integrated Files System (IFS).

TYPFL2ZP

TYPFL2ZP(*DB|*IFS)

Specifies the type of file system that contains the files to be unzipped. Reflected for files in parameters FILES and EXCLUDE.

***DB** Files to be unzipped are in the QSYS Library file system.

***IFS** Files to be unzipped are in the IFS (Integrated Files System).

TYPLISTFL

TYPLISTFL(*DB|*IFS)

Specifies the "type of files system" that will be used for the input list file and/or the output list file of selected items.

To use input list files, see parameters INCLFILE (file section list) or EXCLFILE (file exclude list). To create an output list file of the selected files items, see parameter CRTLIST.

***DB** Files are in the QSYS Library file system.

***IFS** Files are in the IFS(Integrated Files System).

VERBOSE

VERBOSE(*NORMAL|*NONE| *ALL|*MAX)

Specifies how the detail will be displayed during a PKUNZIP run.

The allowable values are:

***NORMAL** Displays most informative message to show PKUNZIP is processing.

***NONE** Displays only major exception information.

***ALL** Displays all messages.

***MAX** Used only for debugging purposes.

VIEWOPT

VIEWOPT(*NORMAL|*DETAIL|*BRIEF|*COMMENT)

Specifies the level of information produced when viewing the archive.

The allowable values are:

***NORMAL** Shows the Original File Length, Compression Method, Compressed Size, Compression Ratio, File Date and Time, 32-bit CRC value, and File Name for each file in the archive.

***DETAIL** Shows very detailed technical information about each file in the archive. It will also show all extended attribute (extra data fields) information that was stored in the archive produced by PKZIP (only if the PKZIP keywords EXTRAFLD(*YES) or DBSERVICE(*YES) were specified).

***BRIEF** Shows the Original File Length, File Date and Time, and File Name for each file in the archive.

***COMMENT** Same as the *NORMAL option, but also shows any file comments stored on a separate line after its details.

VIEWSORT

VIEWSORT(*ASIS|*DATE|*DATER|*NAME|*NAMER|*PERCENT|*PERCENTR| *SIZE|*SIZER))

Specifies the sequence of the viewing display.

The allowable values are:

***ASIS**	List the files in the sequence in which they are stored in the archive, such as, as is.
***DATE**	List the files in ascending order of the file's date & time as stored in the archive.
***DATER**	List the files in descending order of the file's date & time as stored in the archive.
***NAME**	List the files in ascending order of the file name as stored in the archive.
***NAMER**	List the files in descending order of the file name as stored in the archive.
***PERCENT**	List the files in ascending order of the compression percentage as stored in the archive.
***PERCENTR**	List the files in descending order of the compression percentage as stored in the archive.
***SIZE**	List the files in ascending order of the uncompressed file size as stored in the archive.
***SIZER**	List the files in descending order of the uncompressed file size as stored in the archive.

Chapter 11. - Processing with GZIP

Introduction to GZIP (GNU zip)

GZIP (GNU zip) is a compression utility designed to use a different standard for handling compressed data in an Archive. Its main advantages over other compression utilities are much better compression, and freedom from patented algorithms. It has been adopted by the GNU project and is now relatively popular on the Internet. GZIP was written by Jean-Loup Gailly (jloup@gzip.org) and Mark Adler (for the decompression code).

GZIP (GNU zip) utility program (available on a number of platforms including MVS, UNIX, and PC) can be used like ***PKZIP for iSeries*** to compress and extract data. ***PKZIP for iSeries*** in producing GZIP Archives implements two GZIP standard specifications:

- **RFC 1952:** GZIP file format specification Version 4.3, which documents the GZIP specifications and the format of a GZIP Archive file.
- **RFC 1951:** DEFLATE Compressed Data Format Specification Version 1.3, which documents the compression algorithm used by GZIP processing.

The RFC is a process to promote specifications and standards throughout the Internet community and can be found at www.faqs.org/rfcs. Both RFC 1952 and RFC 1951 specifications are platform-independent; therefore, data that was compressed on one platform, for example, UNIX, may be decompressed on another platform, for example, iSeries or MVS.

The one significant advantage of GZIP Archive files over ZIP archive files is the ability to handle larger (greater than 4 Gig) file sizes. The standard ZIP archive format restricts processing to uncompressed files that are less than 4 Gig and cannot create an archive containing multiple files that would meet or exceed the 4 Gig limit. These restrictions are due to the size of the specified fields (4 byte fields) that contain the file size information within the archive. The GZIP Archive format (see RFC 1952) can process files of any size. This format does not maintain a 'directory' of information for individual files and allows sizes to 'wrap' at 4 Gig, therefore, it does not suffer from size restriction.

GZIP Archive Files Used By *PKZIP for iSeries*

The term GZIP Archive file is used to describe the file that holds data that has been compressed by one of the GZIP programs and meets the specifications of RFC 1952. At the end of the GZIP Archive is a trailer that contains the file's compressed size, uncompressed size, and a CRC value for the file (which is used to verify that the data which is decompressed is identical to that originally compressed).

A GZIP Archive file can be transferred from one platform to another and can be decompressed by a GZIP-compatible application which is running on that platform. The internal format of a GZIP Archive is identical, no matter what platform compressed the file.

PKZIP for iSeries (by default) creates new archives as members of PF-DTA files with 132-byte records. The archive file is given a text field of 'File created by PKZIP iSeries.' The archive member is given a text field of 'Member created by PKZIP iSeries.' If you wish to create your own archive (perhaps because a larger record size would be convenient), then you can do so, but consider the following:

- When creating the file, do not create any members in it.
- After creating the file, change the MAXMBRS parameter for the file from 1 to *NOMAX.

A GZIP Archive holds files internally in either Text or Binary format, both of which are compatible with other platforms supported by GZIP. Because information held in a GZIP Archive is defaulted for binary processing, ***PKZIP for iSeries*** uses the parameter FILETYPE for Text or Binary processing. When transporting archives between machines that use different character sets for text, for example, EBCDIC and ASCII, the binary format may not be appropriate. Specifying that the file is to be compressed as FILETYPE(*TEXT) will allow ***PKZIP for iSeries*** to perform EBCDIC-to-ASCII conversion, as required. Specifying FILETYPE(*TEXT) may also be useful when PKUNZIP is used on the iSeries to extract data that has been compressed on an ASCII system. A GZIP Archive is similar to a ZIP archive, but normally only contains one compressed file or member. GZIP Archives, like ZIP archives, use the Lempel-Ziv algorithm (Inflate) to compress and decompress data. Unlike a ZIP archive, GZIP archives do not hold a lot of information in various information blocks throughout the archive; instead, they contain only one information block at the beginning of the archive and locates size information at the end of the archive. Some GZIP text data, for example, file name and comments, use the ISO 8859-1 (LATIN-1) character set and therefore will be converted to and from LATIN-1 as required. When a GZIP Archive is created, an information block is placed in the archive before the compressed version of the file. This information block includes the following information about the file:

- The compression method used on the file.
- The date and time of the last update to the file.
- A flag to indicate optional Extended Data Exist (these fields are usually operating system-dependant and may be ignored if identification code is unrecognized).
- The name of the file that was compressed.
- An archive text comment.
- Compressed data followed by the GZIP Trailer at the end of the archive. The trailer includes a CRC value and the original size of the uncompressed file.

Cross Platform Compatibility

Since GZIP Archive files adhere to RFC 1952, the files are compatible across all GZIP-supported platforms. If executable files and other platform-dependent objects are compressed on one platform and then decompressed on another, it is unlikely that they will work on the new platform. The same can be said about EBCDIC vs. ASCII. Because the extra information is platform dependent, most likely it will be ignored by another platform.

A major consideration for cross platform processing is when building the archive in the QSYS Library file system you may end up with pad bytes at the end of the archive due to files have record lengths and the end of the archive will be padded to the record length. Some GZIP products cannot handle the extra pad bytes at the end of the archive. In this case, the archive should be stored in the IFS where the archive will be a true stream file with no pad bytes at the end of the archive.

Special Note on GZIP Passwords

GZIP standard processing (RFC 1952) does not normally allow a password to be placed on a GZIP Archive. ***PKZIP for iSeries*** does allow this feature, but its use may cause compatibility issues with other platforms. ***PKZIP for MVS™*** does use the same password standard, so GZIP Archives with passwords can be exchanged between ***PKZIP for iSeries***, ***PKZIP for VSE™***, and ***PKZIP for MVS™***. Because GZIP Archives that are created with a password with ***PKZIP for iSeries*** or ***PKZIP for MVS™*** are not part of the GZIP standards, these files will probably appear to be corrupt on other platforms.

Processing GZIP Archives

PKZIP for iSeries can create and extract information from GZIP format archives similar to how it can be used to create and extract information from a ZIP archive. The creation of a GZIP Archive and other parameters is exactly like all other processes in ***PKZIP for iSeries***, including use of extended attributes. To create a GZIP Archive file, code the parameter GZIP(*YES). The difference is that the archive can only have one (1) file, and the archive cannot be updated. The PKUNZIP program will identify the GZIP Archive and process it accordingly.

The following are the specific GZIP Restrictions that pertain to the PKZIP and PKUNZIP programs:

GZIP Compressing

- The code used by ***PKZIP for iSeries*** follows the standards specified in the two applicable RFC's. Specifically, these are RFC 1952 (GZIP file format specification Version 4.3) and RFC 1951 (DEFLATE Compressed Data Format Specification Version 1.3). ***PKZIP for iSeries*** should always be able to create a GZIP compatible compressed file and extract data from a GZIP compressed file where the GZIP utility matches these two specifications.
- Parameter COMPRESS(*NO) cannot be used because all GZIP Archives must contain compressed data.
- Parameter COMPRESS(*TERSE) cannot be used because terse compression is a non-standard compression method for GZIP.
- Parameter FTRAN is not valid for GZIP because filenames have to be held in the ISO 8859-1 (LATIN-1) character set.
- Parameter TYPE (type of processing to be performed) cannot be specified because *DELETE, *UPDATE, *FRESHEN, *MOVEF, or *MOVEU as a GZIP Archive cannot be updated once created.
- Only one file is supported per GZIP Archive. When creating an archive, this means that only one file or member can be identified for inclusion in the archive.
- If ***PKZIP for iSeries*** is used to create and encrypt a GZIP Archive using a password, other platforms may not be able to decrypt the data. The encryption algorithm used by ***PKZIP for iSeries*** in a GZIP Archive is similar to that used by PKZIP, but it is not supported as part of the specifications for GZIP.
- Once an iSeries SAVF has been zipped into a GZIP Archive, the archive will extract on another platform but will not be available as a SAVF; it will just be a binary file and of no use on another platform.
- The file name stored in an archive created by ***PKZIP for iSeries*** will typically contain library, file, and member names (directory components) which relate to the qualifiers of the original iSeries name. According to the GZIP specifications, the file name stored should be the original name of the file
  being compressed, with any directory components removed. Most GZIP utilities support directory components, and the default ***PKZIP for iSeries*** processing will include library, file, and member names in the output file name. **Note:** It is not possible to create a GZIP Archive using ***PKZIP for iSeries*** that does not have the file name stored in the archive.

GZIP Extracting

- If a file name is present in the GZIP Archive, it will be held in the ISO 8859-1 (LATIN-1) character set.
- If there is no file name in the archive, one will use the default paths defined in the EXDIR parameter.

- If there is more than one compressed file in the archive, only the first file can be processed.
- When zipping a file into an archive, the archive cannot already exist. It is not possible to merge or update a GZIP Archive.
- VIEW processing may not show all of the details from the archive, since some information is not stored (or not stored in convenient locations) in the GZIP Archive. For example, the compressed and uncompressed file sizes will typically be shown as zero.
- To match the GZIP specifications, the time stored in the archive header will be in universal time format (seconds since 01/01/1970). Because of manipulation during processing, this time will have only a 2-second accuracy (will always be divisible by two) and therefore could be one second off from the original file time.
- There is no standard iSeries method to set the creation date of a file. As a result, the time in the GZIP Archive is ignored when creating the iSeries output file. It may be viewed by specifying the parameter TYPE(*VIEW).

Sample GZIP Processing

Compressing a file

The following example shows how to compress a file into a GZIP Archive. The PKZIP command is used to compress data into an archive. To select the GZIP format for the resulting archive, you must use the GZIP(*YES) option with the PKZIP command:

PKZIP ARCHIVE('MYLIB1/MYARCHFIL(GZ01)') FILES(' TESTLIB1/FILE1TXT') TYPE(*ADD) GZIP(*YES)

The command above will compress the text file TESTLIB1/FILE1TXT into the archive MYLIB1/MYARCHFIL(GZ01) in GZIP format. The archive must not already exist or an error message will be generated and the operation will fail.

The output from the above command should look like the following:

```
File MYARCHFIL created in library MYLIB1.
PKZIP for iSeries(tm) Data Compression Version 5.6,  2003/05/01
Copyright.  2003  PKWARE of Ohio, Inc.  All rights reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
EVALUATION Running
EVALUATION, Warning - This license will expire in 29 days on 2003/06/02
Contact your dealer with the following information
Machine ID = 0107X8WT, Processor Group =  P10
Scanning files for match  ...
File MYARCHFIL in library MYLIB1 with member GZ01 not found.
Found  1 matching files
Member GZ01 added to file MYARCHFIL in MYLIB1.
Member GZ01 removed from file MYARCHFIL in MYLIB1.
Member PZ3AF2447F added to file MYARCHFIL in MYLIB1.
Compressing TESTLIB1/FILE1TXT(FILE1TXT) in TEXT mode
Add  TESTLIB1/FILE1TXT/FILE1TXT  --   Deflating (31%)
Member PZ3AF2447F renamed to member GZ01.
Member GZ01 file MYARCHFIL in MYLIB1 changed.
PKZIP Compressed 1 files in GZIP Archive MYLIB1/MYARCHFIL(GZ01)
PKZIP Completed Successfully
```

Chapter 12. - Messages

PKZIP for iSeries messages are broken down into 3 types. When using *VIEW type, all displays of the archive and files information use messages AQZ0800 thru AQZ0899. The Licensing Install and Lisping validation use messages that start with AQZ9nnnn. All other messages start with AQZ0001 through AQZ0799.

There are two messages that are issued as Escape messages, and because they are issued as an Escape message, they can be monitored with the MONMSG command in CL programs. Escape messages indicates a condition causing PKZIP or PKUNZIP to end abnormally, without completing its work. By monitoring for escape messages, you can take corrective actions or clean up and end your procedure or program. To see the actual condition that may have caused the problem, you should review all previous message from the job log. If PKZIP or PKUNZIP issues one of these escape message, and the messages are not being monitored, then the iSeries will issue an inquiry message requiring a reply (normal iSeries processing).

The message numbers that are issued as Escape are:

AQZ0022 - PKZIP Completed with Errors.

AQZ0038 - PKUNZIP Completed with Errors.

There are four messages that are issued as Completion message types and are also issued as a Status message type. Because the messages are issued as a Status type, they can be monitored with the MONMSG command in CL programs. The message numbers that are issued as Status message types are:

AQZ0012 - PKZIP ending with Nothing to do for <&1>.

AQZ0020 - PKZIP Completed Successfully.

AQZ0024 - PKZIP Completing with a Warning. No Files Selected.

AQZ0037 - PKUNZIP Completed Successfully.

EXAMPLE:

```
 Msg ID
  |   Severity
  |     Code
  |      |     Message Text
  |      |       |
AQZ0109-00: Compressing &1 in &2 mode
```

Explanation: Informational: Compressing has started for the file with a specified mode. The modes are: 1) SAVF, 2) DATABASE, 3) BINARY, 4) TEXT, and 5) Text in EBCDIC.

The Message text &1 will be the file name and &2 will be the mode such as:

Compressing TESTLIB/TESTFILE(TESTFILE) in TEXT mode

AQZ0001 - AQZ0799 Messages

AQZ0001-00

PKZIP: &1 &2.

Explanation: This message is used in conjunction with other messages to provide extended information. See Messages preceding.

AQZ0002-40

Unexpected End of File: &1 &2.

Explanation: While reading an Archive file in PKZIP, an unexpected End of File was encountered. The file may not have been created properly or copied to completion. Try running the TYPE(*TEST) to see it can identify more information and review all other messages.

AQZ0003-40

Archive file Structure Error: &1 &2.

Explanation: This message is associated with the previous message for exact cause. This indicates a major archive failure for PKZIP to process. Review all messages and run PKUNZIP with TYPE(*TEST) to see if there are more details. This may indicate that current archive is incomplete or corrupted.

AQZ0004-40

Out Of Memory: &1 &2.

Explanation: While trying to allocate memory in PKZIP or PKUNZIP, a failure occurred and no memory was granted. Check other messages in Job Log possible causes and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0005-40

Logic Error: &1 &2.

Explanation: This message should never occur. If it does it may be due to a corrupted Archive file. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0006-40

Error opening temp archive file:<&1>.

Explanation: While trying to process PKZIP, a failure occurred opening the temporary Archive file<&1>. Review other messages with PKZIP and the iSeries for more details of why the failure occurred. Attributes used were=&2'.

AQZ0009-40

User interrupt or termination: &1 &2.

Explanation: PKZIP encountered a request to terminate. If it continues to be unknown, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0010-40

Error using temp file: &1 &2.

Explanation: While trying to process PKZIP, a failure occurred with the temporary Archive file. Review other messages with PKZIP and the iSeries for more details of why the failure occurred.

AQZ0011-40

Read or Seek error: &1 &2.

Explanation: While reading an Archive file in PKZIP, an unexpected Error was encountered. The file may not have been created properly, or copied to completion. Try running the TYPE(*TEST) to see it can identify more information and review all other messages.

AQZ0012-40>.

PKZIP ending with Nothing to do for <&1>.

Explanation: PKZIP found no files to process. Review the FILES EXCLUDE keywords for the selections. This message can be monitored using MONMSG.

AQZ0013-40

Missing or empty archive file: &1 &2.

Explanation: An Archive file to process *UPDATE, *FRESHEN, or *DELETE is empty. No TYPE was processed.

AQZ0014-40

Error writing to a file: &1 &2.

Explanation: An unexpected error was encountered while writing the Archive file. PKZIP is terminated and the temporary archive file is corrupted. Review other PKZIP message and iSeries messages to interpret why this message occurred.

AQZ0015-40

Could not open to write: &1 &2.

Explanation: PKZIP could not create a list file or ARCHIVE file. PKZIP is terminated. Review iSeries message for reason.

AQZ0018-40

Could not open a specified file to read: &1 &2.

Explanation: An archive, list, or extracted file could not be opened for reading. Review PKZIP messages and iSeries that occurred in run.

AQZ0019-00

PKZIP Compressed &1 files in Archive &2.

Explanation: &1 is the number of files that were compressed in this run of PKZIP storing them in the Archive file &2.

AQZ0020-00

PKZIP Completed Successfully.

Explanation: PKZIP Completed Successfully. This message can be monitored using MONMSG.

AQZ0021-10

There are no files to select from: Zip ending.

Explanation: Either no files were selected with Files parameters for all actions, or no files qualified for selection with *FRESHEN or *UPDATE TYPE. Review archive and selection parameters.

AQZ0022-40

PKZIP Completed with Errors.

Explanation: See job log for previous errors messages. An error occurred that caused PKZIP to complete unsuccessfully. AQZ0022 message will be issued as an Escape message type. Escape messages indicate a condition causing PKZIP to end abnormally, without completing its work. This message can be monitored using MONMSG.

AQZ0023-10

Invalid date {&1} for date select option &2.

Explanation: DATETYPE with *BEFORE or *AFTER was selected in PKZIP, but has an invalid date in DATEAB. Format must be mmddyyyy or yyyymmdd. The mm must be 1 thru 12. The dd must be 1 thru 31.

AQZ0024-00

PKZIP Completing with a Warning. No Files Selected for compression.

Explanation: PKZIP selected 0 files for compression and there were no other actions (such as *DELETE, Archive Comment update) to revise the archive. This message can be monitored using MONMSG.

AQZ0025-40

File name <&1> contains special characters.

Explanation: The file contains characters not valid for a file name.

AQZ0026-30

Invalid option(s) used with DELETE ; ignored.

Explanation: This message should never occur. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0027-30

Archive file is empty, cannot make it as old as latest entry.

Explanation: PKZIP run with archive file in the IFS file type. Cannot set date and time of Archive to latest time of files because Archive file is empty.

AQZ0028-30

Archive file has only directories, cannot make it as old as latest entry.

Explanation: PKZIP run with archive file in the IFS file type. Cannot set date and time of Archive to latest time of files. The archive file is OK.

AQZ0029-40

Name Not match: &1.

Explanation: A file in the FILES selection keyword could not be found or matched with a pattern. PKZIP will not process.

AQZ0030-00

Scanning files for match...

Explanation: Information only. The Search has started for file selections.

AQZ0031-00

Found &1 matching files.

Explanation: Information only. The number of selection files that was found is considered for processing.

AQZ0032-00

Copying temp &1 to &2.

Explanation: Copies a temporary Archive file to the requested Archive name.

AQZ0034-00

Searching Archive &1 for files to extract.

Explanation: Informational only. PKUNZIP has started searching the archive for files that meet the selection and excluding parameters.

AQZ0035-00

Extracting file &1.

Explanation: Informational only. PKUNZIP is in the process of extracting file &1.

AQZ0036-00

PKUNZIP extracted &1 files.

Explanation: Informational only. PKUNZIP reports the number of files that were extracted in this run.

AQZ0037-00.

PKUNZIP Completed Successfully.

Explanation: Informational Only. PKUNZIP completed with no errors. This message can be monitored using MONMSG.

AQZ0038-40

PKUNZIP Completed with Errors.

Explanation: During a run of PKUNZIP, errors were encountered. Some processing may or may not have completed. Review the preceding messages for further details. AQZ0038 message will be issued as an Escape message type. Escape messages indicates a condition causing PKUNZIP to end abnormally, without completing its work. This message can be monitored using MONMSG.

AQZ0039-00

PKUNZIP found &1 file(s) in Error.

Explanation: The run of PKUNZIP found &1 files with some type of error resulting in these files not being processed. Consult the job log to discover why these files were in error.

AQZ0040-10

PKUNZIP skipped &1 file(s).

Explanation: PKUNZIP skipped the files due to target files already existing on the system and the parameter to OVERWRITE was set to *NO, which informs the system not to overwrite any file that already exists.

AQZ0041-00

Searching GZIP Archive &1 for files to extract.

Explanation: Informational only. PKZIP is searching for the files to process with GZIP option.

AQZ0042-00

Scanning Archive &1 for &2 files in archive.

Explanation: PKZIP is scanning the input archive for files. Informational only when VERBOSE is set. For large archive with 10 of thousands of files it may take a while for this process.

AQZ0097-10

Input Archive was found to be Digitally Signed..

Explanation: The Archive was Digitally Signed. This PKZIP update will destroy the signature.

AQZ0101-00

Add &1 -- &2.

Explanation: Information providing the file and the type of compression method (Deflate, Terse, or Store) along with the percent of compression.

AQZ0102-00

Freshening: &1 with &2.

Explanation: Informational: PKZIP running with TYPE(*FRESHEN) is processing the file with compression method (Deflate, Terse, or Store) and the percent of compression.

AQZ0103-00

Updating: &1 with &2.

Explanation: Informational: PKZIP running with TYPE(*UPDATE) is processing file with compression method (Deflate, Terse, or Store) and the percent of compression.

AQZ0104-00

Deleting: &1.

Explanation: Informational: Delete file &1 from the archive.

AQZ0105-00

Translating to ASCII.

Explanation: Informational with VERBOSE(*max) indicating that data will translated to ASCII.

AQZ0106-00

Zip diagnostic: &1 <&2>.

Explanation: Zip diagnostic: &1 &2 File in archive is up to date with date & time, or the file in the archive is now missing. Only when Verbose(*max).

AQZ0107-40

Attempting to restore &1 to its previous state.

Explanation: A failure occurred while running PKZIP with an existing archive. An attempt is being made to set the archive to a previous state prior to the original processing start time.

AQZ0108-00

Excluding <&1>.

Explanation: In run PKZIP, a file found in the selection process was excluded due to a match excluding file parameter.

AQZ0109-00

Compressing &1 in &2 mode.

Explanation: Informational: Compressing has started for the file with a specified mode. The modes are: 1) SAVF, 2) DATABASE, 3) BINARY, 4) TEXT and 5) Text in EBCDIC.

AQZ0110-00

The Output List file <&1> is being created.

Explanation: Informational: to indicate that file &1 is being created, and will also be used to add list records selected in the run. See Keyword-CRTLIST for more details.

AQZ0111-00

&1 Records written to the Output List file &2.

Explanation: Informational: indicates that &1 records were outputted to the list file &2. See keyword-CRTLIST for more details.

AQZ0112-00

&1 records read from Include File &2.

Explanation: INCLFILE parameter was optioned in PKZIP or PKUNZIP. This message displays how many records were in the file.

AQZ0113-00

Include parameters supplied &1.

Explanation: Informational: How many include items found in FILES and INCLFILE.

AQZ0114-00

&1 records read from Exclude File &2.

Explanation: EXCLFILE parameter was optioned in PKZIP or PKUNZIP. This message displays how many records were in the file.

AQZ0115-00

Exclude parameters supplied &1.

Explanation: Informational: How many include items found in EXCLUDE and EXCLFILE.

AQZ0116-00

Zip diagnostic: &1cluding <&2>.

Explanation: Informational PKZIP: Display of file being included or excluded.

AQZ0117-00

File matches archive file – skipping.

Explanation: PKZIP running with a file selected, which is the current file, requested as the archive. This file is skipped from the selection process.

AQZ0118-40

Replace: cannot open &1.

Explanation: PKZIP is trying to copy the temporary archive file to the archive file. The archive file cannot be opened. Check the job log for messages as to why the open failed. If specifying parameter TMPPATH in the Library File system, verify that both Library and File name is specified (for example, MPPATH('MYLIB/TEMPARCH').

AQZ0119-40

Fcopy: Write error.

Explanation: PKZIP is trying to copy the temporary archive e file to the archive file. The archive encountered an error while writing the new archive file. Check the job log for messages to determine why the write failed.

AQZ0120-40

Fatal: Incorrect compressed size - &1.

Explanation: A failure occurred while compressing. The size reported by the compression engine is different from the offset determined in the archive file. This should never happen. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0121-00

Stats: &1 &2.

Explanation: Information in PKZIP with verbose on. Shows the size of the file being compressed and the size of the compressed file.

AQZ0122-00

Enter new Zip file Comment for file &1.

Explanation: FILESTEXT was set to edit a new comment for files being added to the archive. Enter the file's comment and press the Enter key.

AQZ0123-00

Enter Zip file Comment for file &1.

Explanation: An unexpected error was encountered while writing the Archive file. PKZIP is terminated, and the FILESTEXT was set to edit Comments for files being added or updated to the archive. Enter the file's comment, and press the Enter key.

AQZ0124-00

Zip Info: &1 &2.

Explanation: Informational PKZIP files when processing with enhanced DEBUG options.

AQZ0125-00

Zip: reading &1.

Explanation: Informational PKZIP: reading the file & in the current archive. Only is displayed with Verbose(*MAX) during fix option. This option is not available at this time.

AQZ0126-00

Zip Info: &1 for &2.

Explanation: Informational PKZIP: Compression statistic of file &2. Only Displays with Verbose(*MAX) during fix option. This option is not available at this time.

AQZ0127-00

Global settings are "No wildcard selection", but FILES contains an *.

Explanation: The Files Parameter contains a wildcard selection <&1>, but the wildcard global setting is coded for NO wildcards selections. Correct and rerun or see your administrator for the PKZIP settings.

AQZ0129-40

Local header not found for &1.

Explanation: The current, loaded archive file is corrupted. The Local header cannot be found for the compressed file &1. Processing of PKZIP will not continue. Review other messages. If file was created using PKZIP, please contact the Product Services Division at 1-937-847-2687 for assistance. If the file came from another source, review the creation of the file to verify if it was transferred properly to iSeries.

AQZ0130-00

Self Extracting code for &1 found in archive file.

Explanation: Informational PKZIP: Adjusting offsets in the archive file &1 for local headers. Only displays with Verbose(*max).

AQZ0131-00

Zip diagnostic: Deleting file &1.

Explanation: Informational PKZIP with TYPE(*DELETE): The file &1 was found in the archive and removed.

AQZ0132-00

Error in Deleting file &1.

Explanation: PKZIP with TYPE(*DELETE) could not delete the file &1 from the Archive. Review job log and other message that occurred during the PKZIP run to determine the cause.

AQZ0133-40

Deleting directory &1 (if empty).

Explanation: PKZIP with TYPE(*DELETE) found a match and it was an empty directory. The directory is being removed from the archive file.

AQZ0134-00

Zip Info: &1 &2.

Explanation: Informational PKZIP about archive files when processing with enhanced debug turned on.

AQZ0135-10

Zero-length name for entry # &1.

Explanation: An error occurred reading the archive file for entry # &1. The name of the file has a zero length. If processing continues, unpredictable results may occur. The source of the
archive file and its history of reaching it current state should be reviewed. Other files in the archive should be valid.

AQZ0136-00

Bad extended local header for.

Explanation: An archive file indicated it contained an extended local header. The header is corrupted. Processing will be terminated. Run PKUNZIP with TYPE(*View) and VIEWOPT(*Detail) to see if more information is available.

AQZ0137-10

Extended local header not found for &1.

Explanation: An archive indicated it contained an extended local header for file &1. The extended header could not be found nor processed. Run PKUNZIP with TYPE(*View) and VIEWOPT(*Detail) to see if more information is available.

AQZ0138-00

Missing end signature -- probably not a archive file (did you remember to use binary mode when you transferred it?).

Explanation: The archive file did not have an end signature, which usually indicates it is not an archive file or it is an incomplete archive file.

AQZ0139-00

Multiple disk information ignored.

Explanation: PKZIP and PKUNZIP do not support Multiple disk functionality.

AQZ0140-00

Name lengths in local and central differ for &1.

Explanation: The archive file is corrupted. The name lengths of the file name in the local directory are different than the name length in the Central directory.

AQZ0141-00

Names in local and central differ for &1.

Explanation: The Archive file corrupted. The file name & in the Local directory is different than the file name in the Central directory. Invalid Archive. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0143-00

Rename failure: New archive file left as: &1.

Explanation: PKZIP is trying to rename the temporary name for a requested archive and has failed. Review the job log for cause. The archive file &1 is still a good archive, but is saved as the temporary name. Rename the temporary file manually to the requested archive name.

AQZ0144-40

Rename failure: Tried to rename to &1.

Explanation: Part 2 of previous message AQZ0143-00.

AQZ0145-40

PKZIP could not open file for reading: &1.

Explanation: PKZIP could not open file &1 for reading to compress. Review Job Log and other messages for cause.

AQZ0146-40

Archive file and directory with the same name: &1.

Explanation: PKZIP found a file and directory to be selected with the same name. &1 will be bypassed and Job terminated.

AQZ0147-00

will just copy entry over: &1.

Explanation: PKZIP will copy the archive over the current archive file &1.

AQZ0148-40

Archive file empty.

Explanation: PKZIP process ended with no files in the archive file. Review selection process and preceding messages. The Archive is not valid and cannot be used to extract files.

AQZ0149-10

File is being skipped due to previous error.

Explanation: An error occurred processing file &1 and the parameter ERROPT was set to *SKIP the file.

AQZ0150-40

Archive File <&1> is not found or empty without ADD TYPE.

Explanation: PKZIP process is running without TYPE(*ADD); cannot find the specified archive file, or the Archive file is empty. For Actions other than *ADD, a valid existing archive must exist.

AQZ0151-00

<&1> include pattern or file could not find a match in search path.

Explanation: The Include pattern in the FILES or INCLFILE could not find a match during the PKZIP run. Review selection parameters, the library list, or the current directory.

AQZ0152-20

Cannot have duplicate names in Zip Include.

Explanation: PKZIP found entries in the FILES or INCLFILE that results in duplicate entries. Correct and rerun.

AQZ0153-00

Duplicates: 1st=<&1>, 2nd=<&2>.

Explanation: Part 2 of message AQZ0152 showing the duplicate files.

AQZ0154-20

Cannot read Archive file <&1> for *FRESHEN or *UPDATE.

Explanation: PKZIP is running with TYPE *FRESHEN or *UPDATE and could not read the archive file. Review Job Log for other messages to determine cause.

AQZ0155-30

No files found for <&1> in archive file for *DELETE TYPE.

Explanation: The file pattern specified for a PKZIP run with *DELETE found no files that matched in the archive.

AQZ0156-00

Searching for files to include...

Explanation: PKZIP searching for files that match entered file specifications.

AQZ0157-40

Library <&1> cannot not be found.

Explanation: PKZIP, in an attempt to find file selections, could not find a specified Library. Review selection criteria in FILES and INCLFILE. Run is terminated.

AQZ0158-40

File not found in Library &1.

Explanation: PKZIP, in an attempt to find file selections, could not find a specified file in a specific Library. Review selection criteria in FILES and INCLFILE. Run is terminated.

AQZ0160-40

SFX File &1 Cannot be opened. Run Terminated.

Explanation: SELFXTRACT was coded to build an archive with a self extracting preamble. While trying to open that SFX File &1, an error occurred. See previous detail job log for more details on the failure. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0161-40

Error occurred while reading SFX file &1.

Explanation: SELFXTRACT was coded to build an archive with a self extracting preamble. While trying to read that SFX File &1, an error occurred. See previous detail job log for more details on the failure. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0162-40

Error Occurred while coping SFX file &1 to archive file.

Explanation: SELFXTRACT was coded to build an archive with a self extracting preamble. While trying to copy SFX File &1 to the archive, an error occurred. See previous detail job log for more details on the failure. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0163-00

Self Extracting code <&1> was added to archive with &2 bytes

Explanation: SELFXTRACT(&1) was coded to build an archive with a self extracting preamble. This is informational only on the addition of the preamble &1 to the archive.

AQZ0164-00

Removing &1 bytes preamble code from Archive File &2.

Explanation: SELFXTRACT(*REMOVE) was coded to remove the self extracting preamble from the archive. This is informational only on the removal of the preamble.

AQZ0165-00

File &1 exceeds 4 Gig and PKZIP is not running with GZIP(*YES). File bypassed.

Explanation: PKZIP is trying to compress file &1 in a standard archive (parameter GZIP is *NO) and the sizes exceed 4 Gigabytes. Use GZIP(YES) to compress.

AQZ0166-40

Self Extraction Creation cannot be performed with Encryption.

Explanation: SELFXTRACT was coded to create a Self Extracting archive with Advanced Encryption. Advanced Encryption is not supported with self extracting archives.

AQZ0167-40

Self Extraction Creation cannot be performed with Large File Support.

Explanation: SELFXTRACT was coded to create a Self Extracting archive and it was determine that ZIP64 format will be required for Large File Support. ZIP64. Large File Support is not supported with self extracting archives.

AQZ0168-40

Self Extraction preamble &1 is not supported.

Explanation: SELFXTRACT was coded to create a Self Extracting archive with &1. It is not available on your current environment. Job will be terminated. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0201-00

PKUNZIP Archive: &1.

Explanation: PKUNZIP Informational: Currently processing Archive file &1.

AQZ0203-10

Caution: File name not matched: &1.

Explanation: PKUNZIP could not find the specified file name in the FILES and/or INCLFILE parameters. File is not extracted.

AQZ0204-10

Caution: Excluded file name not matched: &1.

Explanation: Based upon entered EXCLUDE and EXCLFILE parameters, PKUNZIP did not find any names to exclude. This is informational only.

AQZ0205-00

Please check that you have transferred or created the Archive File in the appropriate BINARY mode.

Explanation: PKUNZIP displays this message as a reminder of a common cause of an invalid archive file. This message appears with invalid archive messages.

AQZ0206-40

Archive name <&1> contains invalid characters.

Explanation: The archive name contains characters that are invalid and will cause the archive to fail to be created. Correct archive and Re-Run.

AQZ0208-00

Skipping: &1 unsupported compression method &2.

Explanation: PKUNZIP, while trying to extract file &1, found an unsupported compression method. The file is unable to be extracted. Run PKUNZIP with TYPE(*VIEW) and VIEWOPT(*Detail) for more information about the file and its compression method. This file is skipped in this extraction run

AQZ0211-00

&1 appears to use backslashes as path separators.

Explanation: The selection file &1 appears to have double \\ in the name for path separators. Will continue processing with normal separators. Review results to verify.

AQZ0212-00

Error: Invalid response [&1].

Explanation: An Invalid response was received for a reply message. Review the reply message request for a valid response, and once located, enter a valid response.

AQZ0213-00

At least one error was detected in &1.

Explanation: PKUNZIP encountered at least 1 error while processing. Review job log for messages.

AQZ0214-00

Caution: Zero files tested in &1.

Explanation: PKUNZIP found no files to test while processing with TYPE(*TEST). Review job log for other messages and review selection criteria. Run PKUNZIP with TYPE(*VIEW) and VIEWOPT(*Detail) to verify archive file.

AQZ0215-00

Skipping: &1 unable to get password.

Explanation: PKUNZIP is processing file &1 and it is encrypted, but no password was entered for this run. Re-run PKUNZIP with a correct password for file &1. Processing continues with other files.

AQZ0216-00

Skipping: &1 incorrect password.

Explanation: PKUNZIP is trying to process file &1 that is encrypted, but the password entered for this run was incorrect for this file. Processing Continues. Re-Run with a correct password.

AQZ0217-00

&1 file(s) skipped because of incorrect password.

Explanation: During this run of PKUNZIP, &1 file(s) were not processed due to incorrect passwords or missing passwords.

AQZ0218-00

(May instead be incorrect password).

Explanation: Part 2 of message AQZ0226 to remind that the failure may be due to an incorrect password. Review to see if file is encrypted by running PKUNZIP with TYPE(*VIEW) and VIEWOPT(*Detail) for the file and verify that encryption is on for the file.

AQZ0219-00

No errors detected in compressed data of &1.

Explanation: PKUNZIP encountered errors with Archive file &1 during run. Review job log for messages.

AQZ0220-00

No errors detected in &1 for &2 file(s) tested.

Explanation: PKUNZIP was run with TYPE(*TEST). No errors were found in testing the selected files in the archive file &2.

AQZ0221-00

&1 file(s) skipped because of unsupported compression or encoding.

Explanation: PKUNZIP encounter &1 files that were bypassed due to unsupported Compression methods. Review job log for details.

AQZ0224-00

Warning: &1 is probably truncated.

Explanation: While trying to extract file &1, some type of write error occurred, forcing an incomplete extraction. The file has probably been truncated. Review Job Log for more information about why this message occurred.

AQZ0225-20

&1: Unknown compression method.

Explanation: Compression method number &2 was found in the archive for file &1. This compression is unknown and is not supported by this PKUNZIP version. PKUNZIP will skip this file and then will attempt to extract the next file within the archive.

AQZ0226-00

&1: Bad CRC &2.

Explanation: While extracting file &1, the CRC calculated for the file was not the same as the CRC stored in the Archive for the file.

AQZ0227-00

&1: Compressed EA data missing (&2 bytes).

Explanation: In extracting file &1, it was identified to be created as an EF NTSD file. The EA data is missing and the file will be not be extracted.

AQZ0228-00

Field entry: EF block length (&1 bytes) exceeds remaining EF data (&2 bytes).

Explanation: Inconsistent Extra Field data was found; it will be ignored. The extracted file data should be good.

AQZ0231-00

&1: Bad CRC for extended attributes.

Explanation: A bad CRC was found in the extended attributes for an archive created by an OS/2 system. Extended Attributes are ignored.

AQZ0232-00

&1: Unknown compression method for EAs (&2).

Explanation: An unknown compression method found for an archive created by an OS/2 system. File is not extracted. Run PKUNZIP with TYPE(*VIEW) and VIEWOPT(*ALL) to see archive details.

AQZ0234-00

&1 archive&2 successfully processed.

Explanation: Informational: &1 is the number of files that were processed successfully in the archives.

AQZ0235-00

&1 archive&2 had warnings but no fatal error.

Explanation: Informational: &1 is the number of files processed in the archives that issued a warning. Review job log for their warnings.

AQZ0236-00

&1 archive&2 had fatal errors.

Explanation: There were &1 files in the archives that had a fatal error and could not be processed. Review job log for messages of the files that failed.

AQZ0240-00

No Archive files found.

Explanation: While searching for an Archive to extract from, no Archive file was found. Review to see that the archive exists.

AQZ0241-00

Cannot find archive file internal end directory in one of &1 or &2%s.zip, and cannot find %s, period.

Explanation: The Archive file may not be an archive, or it may be corrupted.

AQZ0242-40

Cannot find archive file &1, &2.

Explanation: The Archive file could not be found. Review command parameters.

AQZ0246-00

Note: &1 may be a plain executable, not an archive.

Explanation: The Archive file &1 is not an archive. Either it is corrupted, or it could be an executable object.

AQZ0247-00

Warning [&1]: &2 extra bytes at beginning or within Archive File (attempting to process anyway).

Explanation: The Archive &1 contains extra bytes within the archive. Will attempt continued extraction. Verify other messages within the job log and review source of archive file. If no failing message occurs, the extract file should be valid.

AQZ0250-00

Testing: &1.

Explanation: PKUNZIP Informational: TYPE(*TEST) is in the processing of testing file &1.

AQZ0251-00

Extracting: &1 &2.

Explanation: PKUNZIP Informational: TYPE(*EXTRACT) is in the process of extracting a Stored file &1 with mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0252-00

Unshrinking: &1 &2.

Explanation: PKUNZIP Informational: TYPE(*EXTRACT) is in the process of extracting file &1 that was compressed with the Shrink method, and with either a mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0253-00

Exploding: &1 &2.

Explanation: PKUNZIP Informational: TYPE(*EXTRACT) is in the process of extracting file &1 that was compressed with the Implode method, with either a mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0254-00

Inflating: &1 &2.

Explanation: PKUNZIP Informational: TYPE(*EXTRACT) is in the process of extracting file &1 that was compressed with the Deflate method, with either a mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0255-10

Member <&1> truncated to 10 characters.

Explanation: The member name of a file being extracted exceeds the 10-character limitation for an iSeries file member name. The name is truncated to 10 characters.

AQZ0256-10

Warning: Cannot set times for &1.

Explanation: After extracting a file to the Integrated File System, PKUNZIP failed to reset the date and time from the archive file to the file extracted to IFS. The file data is correct.

AQZ0257-00

When Creating an out list file, TYPE must be *VIEW with *NORMAL view.

Explanation: PKUNZIP has keyword CRTLIST specified and can only be used with TYPE(*VIEW) and VIEWOPT(*NORMAL). Correct and RE-RUN.

AQZ0258-10

Overriding Library <&1> is invalid.

Explanation: Parameter EXDIR was specified in PKUNZIP with TYPE(*EXTRACT) and TYPFL2ZP(*DB). The first path in EXDIR contains an invalid Library due to the name exceeding 10 characters. Run is aborted. Correct command and re-run.

AQZ0259-00

Target file exists. Skipping &1.

Explanation: **Warning:** PKUNZIP was extracting file &1, but the file already exists. Parameter OVERWRITE was either set to *NO, or it was set to *PROMPT and the response on the prompt was N.

AQZ0260-00

Target file newer. Skipping &1.

Explanation: **Warning:** PKUNZIP was extracting file &1 with TYPE(*NEWER). The current file is newer than the file in the Archive and will be skipped.

AQZ0261-00

File: &1 tested OK.

Explanation: Informational: PKUNZIP with TYPE(*TEST) tested file &1 and found NO problems.

AQZ0262-20

Warning! &1 already exists. Reply Y, N, A, R, y, n, or r.

Explanation: Y = Overwrite this file with the file extracted from the archive. N = The file will not be extracted from the archive. A = All files in the archive will be extracted and will overwrite any existing files of the same name. R = you will be prompted for a new file name. The default is N. If you wish to change this default then use the CHGMSGD command.

AQZ0263-00

Enter in the new File Name for &1.

Explanation: PKUNZIP issued message AQZ0262 for prompt for the over writing of file &1. The response received was "R" to rename the file. Enter in a valid name for the new file.

AQZ0264-20

Warning: EBCDIC Translation code x'15' must convert to x'0a' in &1(&2) override.

Explanation: Keyword TRAN (Data EBCDIC Translation Mbr) was selected with an override member. EBCDIC code x'15' must translate to x'0A' for iSeries EOL(End of Line) character. Correct and Re-run.

AQZ0265-00

Warning: The password has embedded blanks.

Explanation: A password was entered for PKZIP keyword PASSWORD() and was found to have blanks embedded. Some systems do not accept embedded blanks in the password. Run continues with the entered password.

AQZ0266-00

Tersing &1 in &2 mode.

Explanation: PKZIP Informational: File &1 is being compressed with the TERSE compression method with either a mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0267-00

unTersing &1 &2

Explanation: PKUNZIP Informational: TYPE(*EXTRACT) is in the process of extracting file &1 that was compressed by the TERSE method, with either a mode of empty, Binary, Text, Text in EBCDIC, SAVF, or Database File.

AQZ0268-10

Warning! Record length mismatch. Archived file record length = &1, output file record length = &2.

Explanation: PKUNZIP has detected that the output file and the file in the archive have different record lengths. PKUNZIP will still try to extract the data, but the format and appearance of the data is not guaranteed. Please consult the manual for more information.

AQZ0270-40

Password does not match the Verify Password. Re-enter and try again.

Explanation: The password entered in the VPASSWORD parameter does not match than the pass word entered in the PASSWORD parameter or VPASSWORD is missing when Advanced Encryption was selected. The run will be terminated. Re-issue the command and verify both PASSWORD and VPASSWORD contains the same values.

AQZ0271-40

Compressed Size Invalid for Advance Encryption.

Explanation: File &1 requires Advance Encryption, but has an invalid compress size of &2. Do a PKUNZIP TYPE(*VIEW) VIEWOPTION(*DETAIL) for more information about file. Extract or Test will continue.

AQZ0272-40

Advance Encryption is invalid for GIZP Archive.

Explanation: PKZIP is running with GZIP(*YES) and the option ADVCRYPT is specifying Advanced Encryption (other than *NONE and ZIPSTD. GZIP only supports ZIP Standard for iSeries and zSeries.
Note: Standard GZIP on platforms other than iSeries and zSeries do not support Encryption.

AQZ0273-40

Major Error Occurred with Advanced Encryption with New File size. Will Try to continue.

Explanation: PKUNZIP failed while extracting a file with invalid advanced encryption controls. The extracted size exceeds the uncompressed size &1 by &2 bytes. Contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0274-40

Encryption Method is not Supported.

Explanation: PKUNZIP cannot extract file due to the encryption method indicated on the archive. Perform PKZIP TYPE(*VIEW) VIEWOPT(*ALL) to check Encryption Method on message AQZ0872. Only valid encryption methods supported are: Zip Standard, AES 128, AES 192, and AES 256.

AQZ0275-40

A File in the Archive coded for Adavnced Encryption was found to be Invalid.

Explanation: PKUNZIP was extracting a file coded as being archived with advanced encryption. The file is invalid due to invalid encryption data. Perform PKZIP TYPE(*VIEW) VIEWOPT(*ALL) and check other messages in Job Log for possible causes. If this

continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0276-40

Security Types Not Supported for file Archive.

Explanation: PKUNZIP could not extract file. Archive indicates that it contains Security Types of Certificate Signature or Combination Password and Signature. PKUNZIP only supports password only. Do a PKUNZIP TYPE(*VIEW) VIEWOPT(*DETAIL) to see security type.

AQZ0280-40

Only TYPE(*ADD) or TYPE(*MOVE) are valid with Spool Files selection.

Explanation: When parameter TYPFL2ZP(*SPL) is used for Spool File selection, only *ADD and *MOVE are valid settings for the TYPE parameter. Correct and re-run.

AQZ0281-40

File Inlcudes and Excludes are invalid for Spool File selection.

Explanation: When parameter TYPFL2ZP(*SPL) is used for Spool File selection, the INCLUDE and EXCLUDE parameters are not allowed. Correct and re-run.

AQZ0282-40

Input list file for file includes and excludes are invalid with Spool File Selection.

Explanation: When parameter TYPFL2ZP(*SPL) is used for Spool File selection, the INCLFILE and EXCLFILE parameters for list input file includes and excludes are not allowed. Correct and re-run.

AQZ0283-40

Date selection (DATETYPE) with Before or After is Invalid with Spool File Selection.

Explanation: When parameter TYPFL2ZP(*SPL) is used for Spool File selection, the DATETYPE must be *NONE. Correct and re-run.

AQZ0284-40

STOREPATH(*NO) is invalid with Spool Selection.

Explanation: When parameter TYPFL2ZP(*SPL) is used for Spool File selection, the STOREPATH must be *YES. Correct and re-run.

AQZ0285-40

The Spooled File and/or Attributes cannot be retrieved. Error Code=&1 with &2.

Explanation: While trying to retrieve a Spool File or the Spool File attributes, an error (code=&1 with &2) occurred. Job will be aborted. It may be that the spool file was being modified. Try re-running. If problem continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0286-40

Job information cannot be retrieved. Job ending with Error Code (&1 - &2).

Explanation: Information about current job could not be retrieved. Job will be aborted. Try |re-running. If problem continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0287-40

Spool File Selection List could not generated. Job ending with Error Code (&1 - &2).

Explanation: While trying to select spool files for processing, the job has failed while generating a list of spool files. Try re-running. If problem continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0288-40

Abnormal Error Occurred while processing Spool Files.

Explanation: While processing a spool file, an internal error occurred with Error Codes (&1 - &2). Try re-running. If problem continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0289-40

Failure occurred during creation of Spool File.

Explanation: While trying to create a spool during extraction, a failure occurred with Error Codes (&1 - &2). Try re-running. If problem continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0291-40

SFJOBNAM in the PKZIP command is incomplete or Invalid.

Explanation: If the Spool File Job Name is "*", Spool File Job User and Spool File Job number must be blank. Otherwise all fields must contain valid name, user and number or they must all be blank. Correct and re-run.

AQZ0292-40

Error Occurred while transforming the spool file. Job is aborted.

Explanation: A major error has occurred while transforming a spool file to an ASCII file. Job is being aborted with exception code(&1-&2). Try re-running the job. If this failure continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0293-40

Parameter SFTARGET(*SPLF) requires SFTGFILE(*GEN1). Correct and re_run.

Explanation: SFTGFILE must be coded as *GEN1 when requesting a target type of Spool File SFTARGET(*SPLF). Correct and re-run.

AQZ0294-40

Spoof File (&1) is &2 type print file and Convert Format is TEXT. File bypassed.

Explanation: Only Spool File Types *SCS, *IPDS and *AFP are valid for Text conversion. &2 type cannot convert to TEXT format.

AQZ0295-10

Spool File &1 is being skipped due to being in OPEN Status.

Explanation: Spool Files that are in an Open Status are not eligible for selection. File will be skipped.

AQZ0296-10

Spool File "&1" Exceeds Maximum allowed size.

Explanation: The Spool File &1 Exceeded the maximum allowed of &2. File will be either skipped or job will end based on ERROPT settings.

AQZ0297-40

The Output Queue for Spool Files selection cannot be found.

Explanation: The output queue specified in parameter SFQUEUE is invalid or cannot be found in the libraries specified.

AQZ0298-40

Invalid Code Page for Spool File conversion.

Explanation: While trying to convert a Spool File to ASCII text, it was found the code page was invalid. Review parameter IFSCDEPAGE for overrides. Error occurred while using Page Code from &1 to &2.

AQZ0299-40

Spool File Target File Cannot be named GEN1 nor GEN2 nor start with an *.

Explanation: Spool file Parameter SFTGFILE cannot have file names GEN1 and GEN2. This is to prevent mistakes by leaving off the leading *. SFTGFILE cannot start with an * other than special names *GEN1, *GEN1P and *GEN2'.

AQZ0300-10

A Specific File name was specified in SFTGFILE, but more than 1 spool file was selected.

Explanation: When specifying a file name in SFTGFILE, only one spool file can be selected. Correct selections and re-run.

AQZ0301-40

Parameter SFTARGET(*SPLF) requires EXTRAFLD(*YES). Correct and re_run.

Explanation: In order to be able to extract a Spool File, extended attributes are required.

AQZ0302-40

FILETYPE parameter must be (*DETECT) when compressing Spool Files.

Explanation: When parameter TYPFL2ZP(*SPL) is coded to compress Spool Files, the FILETYPE parameter must be (*DETECT).

AQZ0303-10

Warning: Only Local Extended Attributes was selected to store iSeries Extended Attributes.

Explanation: This is a warning that the extra data fields will ONLY be stored in the Local Directory because of parameter EXTRADATA(*LOCAL). ***PKZIP for iSeries*** will not utilize the extended attributes.

AQZ0304-30

SFJOBNAM(&1) Job cannot not be found or is invalid.

Explanation: The parameter SFJOBNAM(&1) for job selection of a spool file could not be found or is an invalid JOB. Correct and Re-Run.

AQZ0401-40

Memory allocation failure while processing Parameters &1 &2.

Explanation: A failure occurred while trying to allocate memory. Check other messages in Job Log for possible causes and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0402-40

I/O Error: &1.

Explanation: An I/O error occurred. This message will help describe a previous message. Review all messages in job log to resolve problem.

AQZ0403-40

Open error occurred on Translate table <&1>.

Explanation: While opening file &1, an open error occurred. Review FTRAN and TRAN keyword entries and messages in the job log to determine the cause of problem. Current run is terminated.

AQZ0404-40

Read Error while reading Translate table <&1>.

Explanation: While reading file &1, an error occurred. Review FTRAN and TRAN keyword entries and messages in the job log to determine the cause of the problem. Browse file &1 and review the data to ensure it follows the requirements of a Translate Table. Current run is terminated.

AQZ0405-40

Translate Table must have 256 entries. File <&1> found &2 entries.

Explanation: A Translation Table is required to have 256 entries. Browse file &1 and review the data to ensure it follows the requirements of a Translate Table. Current run is terminated.

AQZ0406-40

Internal Error. Should not occur "&1".

Explanation: If this error occurs, review to ensure you have a valid archive. Please contact the Product Services Division at 1-937-847-2687 for assistance. Current run is terminated.

AQZ0407-40

&1 file size changed while zipping.

Explanation: PKZIP has determined that the size of file &1 has changed while compressing. Size Information (&2). File should be compressed again.

AQZ0408-40

Compression of input file &1 failed. Could not read input file.

Explanation: A read error occurred while trying to compress input file &1. Review all job log messages to determine cause of read error. Job is terminated.

AQZ0409-40

Failure during user space creation &1.

Explanation: A failure occurred while using an API, which required a user space that failed. See message &2 in job log for which API and why the failure occurred. Job is terminated.

AQZ0410-40

API list command failure due error &1 &2.

Explanation: Review error message &1 for more information why API list command &2 failed. Job is terminated.

AQZ0411-40

File &1. is not a database.

Explanation: A List Member API was issued for file &1. A return message, "CPF3C23," indicates this file is not a database file, which is allowed to have members. Review file &1 to ensure it is a valid file to compress. Job is terminated.

AQZ0412-00

API error &1. Get Member descriptions <&2>.

Explanation: While issuing a Get Member descriptions for file &2, error &1 occurred. Review message &1 for more information.

AQZ0413-00

API error &1. Database File descriptions <&2>.

Explanation: While retrieving the file descriptions for file &2, error &1 occurred. Review message &1 for more information.

AQZ0414-00

API error &1. Get Object descriptions <&2>.

Explanation: While retrieving object descriptions for file &2, error &1 occurred. Review message &1 for more information.

AQZ0415-00

API error &1. Change Object descriptions <&2>.

Explanation: While issuing a change to the object &2, error &1 occurred. Review message &1 for more information.

AQZ0416-00

API error &1. Process Command <&2>.

Explanation: While trying to process command &2, error &1 occurred. Review message &1 for more information. This may occur if the user does not have the proper authority for the command.

AQZ0417-40

Error writing List file &1.

Explanation: Keyword CRTLIST was specified and an error occurred while writing to file &1. Review all job log messages for more details. Job will be terminated.

AQZ0418-40

Error opening or closing list file &1.

Explanation: Keyword CRTLIST was specified and an error occurred while &2 was opening or closing file &1. Review all job log messages for more details. Job will be terminated.

AQZ0419-40

Cannot open file <&1> for Include selection.

Explanation: File &1 (specified in keyword INCLFILE) cannot be opened. See job log messages for more information. Verify that file &1 is a valid file for input to include selection. Job is terminated.

AQZ0420-40

Cannot open file <&1> for Exclude filtering.

Explanation: File &1 (specified in keyword EXCLFILE), cannot be opened. See job log messages for more information. Verify that file &1 is a valid file for input to exclude selection. Job is terminated.

AQZ0421-00

File <&1> failed while retrieving the members for the file. Review Job Log for previous message for details of failure.

Explanation: When trying to retrieve the members for file <&1> and record <&2>, the API failed. The job log will show more details in previous messages. One cause may be the file was remote and could not be accessed.

AQZ0450-40

Did not find end-of-central-dir signature at end of central dir.

Explanation: While scanning an archive for proper archive signatures, the end of central dir signature could not be found. Ensure it is a valid archive, or if the archive was sent via FTP; ensure Binary was used.

AQZ0451-40

Expected central file header signature not found (file #&1).

Explanation: While scanning an archive for proper archive signatures, the expected central file header signature could not be found. The archive may be corrupted, or it was transferred from another source incorrectly.

AQZ0452-40

Attempt to seek before beginning of Archive file &1.

Explanation: While scanning the archive for valid information, an offset caused PKZIP to attempt seeking functions prior to the beginning of the archive. The archive may be corrupted.

AQZ0453-40

&1: Bad file comment length.

Explanation: The archive indicated that a comment existed for file &1. The length was zero, or the comment length was wrong. Run PKUNZIP with Detail View for more information.

AQZ0454-40

Invalid option mixture - should never occur by using PKUNZIP Command.

Explanation: Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0455-40

Both Overwrite *NONE and *ALL specified; ignoring *ALL - This should never Occur.

Explanation: By using the PKUNZIP command this error should never occur. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0456-40

&1: bad filename length in &2 directory.

Explanation: File &1 was found to have a bad file name length in the archive &2 directory. Processing will continue at the next entry. Try running PKUNZIP with detail view to gather more information. The archive may be corrupted.

AQZ0457-40

&1: Bad Extra data length in &2 directory.

Explanation: File &1 was found to have a bad Extra data length in the archive &2 directory. Processing will continue at the next entry. Try running PKUNZIP with a detail view to gather more information. The archive may be corrupted.

AQZ0458-40

File # &1: Bad Archive File offset (&2): &3.

Explanation: PKUNZIP was trying to extract or test files in the Archive. File number #1 in archive contained a bad offset length (&2).

AQZ0459-20

Length warning: &1 bytes [&2].

Explanation: Length Warning: Based on the Local directory, the bytes required are less than the actual bytes used. Review the extracted file and ensure it is extracted correctly. Run PKUNZIP with View detail to check files sizes. The file is extracted and the process continues.

AQZ0460-40

Length Error: &1 bytes [&2].

Explanation: Length Failure: Based on the Local directory, the bytes required are more than the actual bytes used. Review the extracted file and ensure it is extracted correctly. Run PKUNZIP with View detail to check files sizes. The file is extracted and the process continues.

AQZ0461-40

Error: Not enough memory to Unshrink &1.

Explanation: While trying to allocate memory to Un-shrink file &1, an error occurred. All processing is terminated. Check other messages in Job Log for possible causes, and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0462-40

File #&1: Bad local header.

Explanation: File in the archive with sequence number &1 has an invalid Local Directory Header. File cannot be extract or tested. The Archive may be corrupted.

AQZ0463-40

(Attempting to re - compensate).

Explanation: An error occurred previously in the archive (see Message Log). An attempt will be made to compensate for the error.

AQZ0464-40

Skipping: &1 %2 volume label.

Explanation: An archive created under another system has Volume label appearing in the archive. iSeries cannot process this request and will skip file &1.

AQZ0468-40

Invalid compressed data to &1 &2.

Explanation: While attempting to uncompress file &2 using method &1, it was determined that the compressed data was invalid or corrupt. Review the log for other messages to help diagnose the problem. Do a PKUNZIP with VIEWOPT(*DETAIL). Verify the source of the archive that is being processed to verify if it was created using a compression product from PKWARE, Inc. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0469-40

Error: Not enough memory to &1 &2.

Explanation: While attempting to uncompress file &2 using method &1, PKUNZIP could not allocate enough memory to extract file. Check other messages in Job Log for possible causes, and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0470-40

&1: Out of memory while inflating EAs.

Explanation: While extracting file &1, an allocation of memory failure occurred while trying to extract and file with EAs attributes. This archive was created by another system type. Check other messages in the Job Log for possible causes, and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0471-40

&1: Unknown error on extended attributes.

Explanation: While extracting file &1, an unknown error occurred while trying to process the Extended attributes data. This archive was created by another system type.

AQZ0472-40

Unsupported extra-field compression type (&1)—skipping.

Explanation: The Extended data fields for file &1 was found, but is not recognized or is Invalid. File will be skipped.

AQZ0473-00

Error: Cannot allocate unzip buffers.

Explanation: While starting to extract the file, PKUNZIP could not allocate enough memory for IO buffers within the file. Job will be terminated. Check other messages in the Job Log for possible causes, and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0475-40

Warning [&1]: Archive File is disk %u of a multi-disk archive and this is not the disk on which the central Archive File directory begins.

Explanation: The Archive indicates that the archive part of a multi-disk archive is invalid for PKUNZIP. Review the source of the archive and ensure the archive is made up of one file.

AQZ0476-30

Warning [&1]: End-of-central-directory record claims this is disk &2 attempting to process anyway.

Explanation: The Central Directory has a setting that indicates this is a multi-archive disk file. It will attempt to process file &1. Expect "errors" and warnings; true multi-part support does not exist.

AQZ0477-30

Warning [&1]: Archive File claims to be last disk of a multi-part archive; attempting to process anyway.

Explanation: Expect \"errors\" and warnings; true multi-part support does not exist.

AQZ0478-30

Error [&1]: Missing &2 bytes in Archive file (attempting to process anyway).

Explanation: While extracting files from Archive file &1 with PKUNZIP, it was found to have &2 bytes missing according to the directories. PKUNZIP will continue attempting to extract the files and will complete the process.

AQZ0479-30

Error [&1]: NULL central directory offset (attempting to process anyway).

Explanation: While extracting from archive file &1, an error was found in the Central Directory offset. PKUNZIP will try to compensate and continue the process.

AQZ0480-30

Warning [&1]: Archive File is empty.

Explanation: PKUNZIP found that archive file &1 is empty and that there is nothing to process.

AQZ0481-30

Error [&1]: Start of central directory not found; Archive file is corrupt.

Explanation: PKUNZIP could not find the start of the central directory in Archive file &1. Archive file is corrupt and the job will be terminated.

AQZ0482-30

Warning[&1]: Reported length of central directory IS &2 bytes too long. Compensating...

Explanation: PKUNZIP found that the length of the central directory of the archive is too long by &2 bytes. PKUNZIP will try to compensate and continue.

AQZ0483-40

End-of-central-directory signature not found in archive &1.

Explanation: An End-of-central-directory signature not found. Either this file &1 is not an archive file, or it constitutes one disk of a multi-part archive. In the later case, the central directory and archive file comment will be found on the last disk(s) of this archive.

AQZ0484-00

Caution: Archive file &1 comment will be truncated.

Explanation: While displaying the Archive file comment, it was truncated due to excess length. This affects only the displayed information, not the data itself.

AQZ0485-30

Error: Cannot delete old &1.

Explanation: Cannot Delete File &1 on IFS system file type. See Job Message Log for more information.

AQZ0486-30

Error: PKZIP cannot open Archive File [&1].

Explanation: Archive File &1 cannot be opened for reading in PKZIP. See Job Message log for more information.

AQZ0487-30

Error: Cannot create &1.

Explanation: PKUNZIP was trying to create file &1 for extraction, but the create process failed. The file was not extracted. See Message Log for create failure.

AQZ0488-40

Error: Archive file &1 read error.

Explanation: PKUNZIP received a file error while reading Archive file &1. Job is terminated. See Message log for Error Reason.

AQZ0489-10

Warning: Filename too long -- truncating.

Explanation: The file name length (including path) found in the archive exceeds the length of 255 bytes allowed in PKUNZIP. File name is truncated.

AQZ0490-10

Warning: Extra field too long (&1). Ignoring.

Explanation: The extended attribute is too long for PKUNZIP to allocate Memory. The attributes will be ignored.

AQZ0491-40

Error: More than &1 simultaneous threads. Some threads are probably not calling DESTROYTHREAD().

Explanation: Internal Error. This should not occur. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0492-40

Error: Could not find global pointer in table Maybe somebody accidentally called DESTROYTHREAD() twice.

Explanation: Internal Error. This should not occur. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0493-40

Error: Global pointer in table does not match pointer passed as parameter.

Explanation: Internal Error. This should not occur. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0494-00

Warning: Cannot set UID &1 and/or GID %d for &2.

Explanation: After extracting a file successfully, PKUNZIP could change the UID attributes for a file in the Integrated File System to UID in the archive settings. See Job Log for more Information.

AQZ0495-00

Warning: Cannot set modification, access times for &1.

Explanation: After extracting a file successfully, PKUNZIP could change the modification time and the access time for a file in the Integrated File System to the time that is stored within the archives. Times are left as the time extracted. See Job Log for more Information.

AQZ0496-00

Warning: Cannot set permissions for &1.

Explanation: After extracting a file successfully, PKUNZIP could not set the authority permissions for the file in the Integrated File System. File has job user as owner. See Job Log for more information.

AQZ0498-40

&1: Write error was encountered while extracting.

Explanation: While extracting and writing file &1, an error occurred. It could be the allowable disk allocation for a user, a job, or a file. See Job Log for more information. Current extracted file is incomplete.

AQZ0499-50

Archive file probably corrupt (&1).

Explanation: A Handler signal occurred in PKUNZIP, which indicates an internal error occurred, or the archive file is corrupt. Please contact the Product Services Division at 1-937-847-2687 for assistance with the archive file and Job LOG. MAJOR Error.

AQZ0501-20

Warning: Cannot allocate wildcard buffers.

Explanation: PKUNZIP could not allocate Memory resources in do_wild for the Integrated File System wildcard buffers. Cannot select files. Check other messages in Job Log for possible causes and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0502-00

Creating directory: &1.

Explanation: Information only: PKUNZIP created directory &1 in the Integrated File System.

AQZ0503-30

Mapname: Conversion of &1 failed.

Explanation: PKUNZIP failed while converting (mapping) the internal file &1 name to an iSeries external file name. Check Log to see if file was extracted successfully.

AQZ0504-20

Checkdir: Cannot create extraction directory &1.

Explanation: PKUNZIP failed while trying to create directory &1 in the Integrated File System required extracting file.

AQZ0505-10

Warning: Cannot set UID &1 for &2.

Explanation: After extracting a file successfully, PKUNZIP could change the UID attributes for a file in Integrated File System to UID in the archive settings. See Job Log for more Information.

AQZ0506-10

Checkdir warning: Path too long; truncating &1.

Explanation: The path name in the archive file is longer than 255 bytes for an Integrated File System. Path will truncate for file extraction.

AQZ0508-10

Checkdir error: &1 exists but is not directory: unable to process.

Explanation: PKUNZIP was extracting a file to the Integrated File System where the Archive file name indicated it is a directory, but it already exists and it is not a directory. A default path will be used.

AQZ0509-10

Checkdir error: Cannot create &1 unable to process &2.

Explanation: PKUNZIP could not create directory &1 in the Integrated Files System. A default directory is used. See Job Log for information for the failure.

AQZ0510-10

Checkdir error: Path too long: &1.

Explanation: PKUNZIP determined that the path of file to extract exceeded 255 bytes. File will use default path.

AQZ0511-40

Failure to create file <&1> in Library <&2>.

Explanation: PKUNZIP was issuing a CRTPF command for a file that was using default settings before extracting and an error occur in the CRTPF command. See Job Log for more information about failure. Files are not extracted.

AQZ0512-40

Failure to create SAVF <&1> in Library <&2>.

Explanation: PKUNZIP was issuing a CRTSAVF command to create file &1 for extracting and the create process failed. See Job Log for more information about failure. Files are not extracted.

AQZ0513-10

Warning! Line &1 has been truncated.

Explanation: When writing a text file, the length of the line exceeded the record length of the output file and the line was truncated. Use an output file with the correct record length.

AQZ0514-30

Failure in creating Library &1 for Archive.

Explanation: PKUNZIP was trying to extract a file to library &1 and the library did not exist, and PKUNZIP failed when trying to create the Library. For more information see Job Log. This may be a security issue. File is not extracted.

AQZ0515-40

PKZIP Failure in file read request. Request bytes &1.

Explanation: While trying to read a selected file for compression, PKZIP experienced an error where the size of the data read failed the request. View the Job log to see if any messages pertain to the file. The file being used may cause this. The job will be terminated, and the archive will be corrupted.

AQZ0516-40

Could not allocate Memory for Terse extraction. PKUNZIP is ending immediately.

Explanation: PKZIP was compressing the file with compression type *TERSE, but could not allocate memory for the TERSE buffers. Check other messages in Job Log for possible causes, and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0517-40

Terse Extraction Failure. Work buffers are too small.

Explanation: PKUNZIP could not complete an extraction of a file compressed in TERSE mode because the buffers were two small. This should not occur for an archive created with this release. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0518-00

Warning! File &1 has associated triggers.

Explanation: PKZIP has detected that the database file being zipped has trigger programs associated with it. PKZIP does not store trigger program information in the archive. See Chapter
8. - Storing Database File Extended Attributes for more information. This message only appears when compressing database files. If you want to preserve the trigger information, first save the file and trigger programs to a SAVF and then add the SAVF to the archive.

AQZ0519-00

Warning! File &1 has associated constraints.

Explanation: PKZIP has detected that the database file being zipped has constraints associated with it. PKZIP does not store constraint information in the archive. See Chapter 8. - Storing Database File Extended Attributes for more information. This message only appears when compressing database files. If you wish constraint information to be preserved, first save the file to a SAVF and then add to the archive.

AQZ0520-00

Warning! File &1 uses an alternative collating sequence.

Explanation: The file being compressed uses an alternative collating sequence (the ALTSEQ keyword was specified in the DDS for the file). PKZIP does not store alternative collating table information in an archive. When unzipped, the file will not have the same access path. This message only appears when compressing database files. If you want to preserve the alternate sort order, first save the file in a SAVE and then add to the
archive.

AQZ0521-00

Warning! File &1 has NULL capable fields.

Explanation: PKZIP has detected that one or more fields in the file are NULL capable. PKZIP will translate NULL numeric fields as zeros and NULL character fields as blanks. See Chapter 8. - Storing Database File Extended Attributes for more information.
This message only appears when compressing database files. If you want to preserve NULL fields, first save the file in a SAVF, and then perform SAVF to the archive.

AQZ0522-00

Database process selected but missing DB extended data record - File &1.

Explanation: An extended data file indicates that Database attributes are present in order to create a DB that is not currently present, but the attributes could not be found. The archive may be corrupt. Will try with default settings.

AQZ0523-00

Failure building DDS source to gen database - File &1.

Explanation: While building the DDS source for creating file &1, an error occurred in routine build_ddssrc with number &2. See job log for other messages. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0525-00

Warning! File &1 has Public Authorization List.

Explanation: PKZIP has detected that the database file being zipped has a Public Authorization List associated with it. PKZIP does not store Public Authorization List information in the archive. See Chapter 8. - Storing Database File Extended Attributes for more information. This message only appears when compressing database files. When extracting the file, it is up to the user to provide Public Authorization List information. If you wish to preserve Public Authorization List information, first save the file to a SAVF, and then add the file to the archive.

AQZ0526-30

Extended attributes exceed 64K

Explanation: The extended attributes for the file being compressed exceeded the maximum length that can be accommodated in a PKZIP archive. Action: Either compress the file with DBSERVICES(*NO) or first save the file in SAVE file and then add to the archive.

AQZ0527-10

Warning: Text Record too long for buffers. Text Record length=&1. Buffer Length=&2

Explanation: A Text Record length exceeded the maximum supported buffer length of &2. Record is truncated. File extract should be reviewed for completeness. Probable cause could be the creations of the file did not insert proper CR or LF characters.

AQZ0600-00

Archive &1 Identified as a GZIP archive.

Explanation: PKZIP was run, specifying existing archive &1 and that the archive is identified as a GZIP archive. GZIP Archive files cannot be updated.

AQZ0601-40

Option GZIP cannot zip to an existing archive.

Explanation: PKZIP GZIP(*YES) parameter has been specified, and the archive indicated on the PKZIP command already exists. A GZIP Archive cannot be updated; you can only compress in GZIP format to a new archive. Specify an archive name that is not in use.

AQZ0602-40

Option GZIP only allows 1 file per archive.

Explanation: GZIP processing has identified more than one file to include in the archive, but only one file is allowed per GZIP archive. Specify a file name or wildcard combination that identifies only one file, or be more specific and specify the library, the file, and the member names.

AQZ0603-40

Invalid TYPE used with GZIP. TYPE(*ADD,*MOVEA,*VIEW) is only valid option for GZIP.

Explanation: A PKZIP TYPE other than *ADD, *MOVEA or *VIEW was specified with option GZIP(*YES). This is an invalid combination. Job terminated.

AQZ0604-00

PKZIP Compressed &1 files in GZIP Archive &2.

Explanation: &1 is the number of files that were compressed in this run of PKZIP, storing them in the GZIP Archive File &2.

AQZ0605-40

GZIP cannot allocate memory for GZIP extraction.

Explanation: PKUNZIP was trying to extract the file from an archive identified as a GZIP archive, but could not allocate memory. Job is terminated. Check other messages in Job Log for possible causes and refer to Appendix C about memory errors. If this continues, please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ0606-40

GZIP Archive File &1 probably corrupt.

Explanation: PKUNZIP was trying to extract from archive file &1, and it was identified as a GZIP archive. It was determined that it could be corrupt based on the setting in the header or trailer.

AQZ0607-40

Cannot Use compress(*STORE or *TERSE) with GZIP.

Explanation: GZIP parameter *YES cannot be used when COMPRESS parameter is set to *STORE or *TERSE.

AQZ0608-40

GZIP *YES cannot be used with FTRAN table.

Explanation: FTRAN table override cannot be used with option GZIP(*YES) due to compliant issues.

AQZ0609-00

Warning! GZIP Archive has non-compliant Flag settings.

Explanation: The archive being processed has reserve bits in the Flag byte set. This may indicate the presence of new fields or options in the archive that prevent the file being extracted from being extracted correctly.

AQZ0610-40

GZIP Archive &1 contains no filename and Default Path was not specified.

Explanation: The GZIP Archive that is being extracted does not contain a file name. To get a file name, use EXDIR keyword and enter a path and filename. If IFS, use path/filename; For Library file system use library/filename.

AQZ0611-40

TEXTEDIT parameter is invalid with GZIP(*YES).

Explanation: GZIP archives do not support file comments. TEXTEDIT must be *NO.

AQZ0800 - AQZ0899 *VIEW Displays Messages

AQZ0800-00

Filename: &1.

Explanation: Displays the Archive file name being viewed or processed.

AQZ0801-00

Archive Comment: "&1".

Explanation: Displays the Archive Comment (if one exists) when TYPE(*VIEW) is used with parameters *NORMAL, *DETAIL, *COMMENT for keyword VIEWOPT().

AQZ0802-00

Length, Date, Time, Name.

Explanation: This message will appear at the top of the file information listing when the option VIEWOPT(*BRIEF) is used with TYPE(*VIEW).

AQZ0803-00

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

Explanation: This message will appear after message AQZ0802 for the file information listing when the option VIEWOPT(*BRIEF) is used with TYPE(*VIEW).

AQZ0804-00

&1 &2 &3.

Explanation: File information listing when the option VIEWOPT(*BRIEF) is used with TYPE(*VIEW). Length=uncompressed size, modifications Date and time of file, and the filename.

AQZ0805-00

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

Explanation: This message will appear after the brief file information listing as a separator when the option VIEWOPT(*BRIEF) is used with TYPE(*VIEW).

AQZ0806-00

&1 &2 file&3.

Explanation: This message is the totals for file information listing when the option VIEWOPT(*BRIEF) is used with TYPE(*VIEW). Total: Uncompressed size=&1, and Number of files in listing=&2.

AQZ0807-00

File Comment: "&1".

Explanation: Displays the file comment text when the option VIEWOPT(*COMMENT or *DETAIL) is used with TYPE(*VIEW).

AQZ0808-00

Length, Method, Size, Ratio, Date, Time, CRC-32, Name.

Explanation: This message displays the 1st heading in the file information listing when either of the parameters *DETAIL, *COMMENT, or *NORMAL are used with the VIEWOPT() option.

AQZ0809-00

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

Explanation: This message displays the 2nd heading in the file information listing when either of the parameters *DETAIL, *COMMENT, or *NORMAL are used with the VIEWOPT() option.

AQZ0810-00

&1 &2 &3 &4 &5 &6 &7 &8.

Explanation: This message displays the file attributes in the file information listing when either of the parameters *DETAIL, *COMMENT, or *NORMAL are used with the VIEWOPT() option. Uncompressed file Length=&1, Compression method used=&2, Compressed Size=&3, Compression Ratio=&4, File Date/ Time=&5, CRC-32=&6, for File Name &7.

AQZ0811-00

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

Explanation: This message displays the totals separator in the file information listing when either of the parameters *DETAIL, *COMMENT, or *NORMAL are used with the VIEWOPT() option.

AQZ0812-00

&1 &2 &3 &4 file&5.

Explanation: This message displays the totals in the file information listing when either of the parameters *DETAIL, *COMMENT, or *NORMAL are used with the VIEWOPT() option. Total Uncompressed size=&1, Total Compressed size=&2, for &3 number files in listing.

AQZ0813-00

Archive: &1 &2 bytes &3 file&4.

Explanation: Archive statistics for Archive File &1, is &2 bytes is length with &3 files archived Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0814-00

Created by: &1 &2.

Explanation: Displays the PKZIP operating system and the PKZIP algorithm version which the file was added to the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0816-00

Minimum file system compatibility required: &1 &2.

Explanation: Information with TYPE(*VIEW) and VIEWOPT(*DETAIL) and only with debug option on. Shows detail Operating System (&1) with version (&2).

AQZ0817-00

Minimum to Extract: &1 &2.

Explanation: Displays the minimum version of the PKZIP algorithm required to extract the file from the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0818-00

Compression method: &1 &2.

Explanation: Displays the Compression Method used to compress the file. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0821-00

Extended local header: &1.

Explanation: Shows the setting of the general-purpose bit to indicate an Extended local header exists. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0822-00

Date and Time &1.

Explanation: Displays the modification Date mm-dd-yyyy and Time hh:mm of files attributes Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0823-00

32-bit CRC value (hex): &1.

Explanation: Displays the value of the Cyclical Redundancy Check of a file in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0824-00

Compressed size: &1 bytes.

Explanation: Displays the compressed size of a file in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0825-00

Uncompressed size: &1 bytes.

Explanation: Displays the Uncompressed size of a file in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0826-00

Length of filename: &1 bytes.

Explanation: Displays the length of the file name of a file in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0827-00

Length of extra field: &1 bytes.

Explanation: Displays the length of the Extended Data field in the Archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0828-00

Length of file comment: &1 characters.

Explanation: Displays the Length of a file comment if it exists in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0829-00

Size of sliding dictionary (implosion): &1K.

Explanation: Displays the size of the sliding dictionary that was used to compress a file in the archive with the Implosion compression method. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0830-00

Number of Shannon-Fano trees (implosion): &1.

Explanation: Displays the number of Shannon-Fano trees that were used to compress a file in the archive with an Implosion compression method. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0831-00

Detected File type: &1 &2.

Explanation: Based on internal Attributes; displays the file type that was detected for a file in the Archive. The Types are: Binary, Text, Text in EBCDIC, SAVF, UNKNOWN. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0834-10

Caution: Archive File comment truncated.

Explanation: The Archive file comment exceeded the buffer size and will be truncated. This is only a warning that affects the display of the comment. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0835-00

Unknown Extended Attribute TAG &1 with length of &2.

Explanation: An Unknown Extended Attribute Key was found; PKUNZIP does not handle this and will be bypassed. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0836-00

Extended attributes: &1, [Length = &2].

Explanation: If extended attributes are stored in the archive for this file, then [Yes] is displayed, otherwise, [No] is displayed. The size of these extended attributes are &2 bytes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0837-00

A sub field with ID &1 with a length of &2 bytes.

Explanation: This is only used in DEBUG mode only. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0838-00

- -

Explanation: Displays a View file separator for reading purposes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0839-00

Found &1 file&2, &3 bytes uncompressed, &4 bytes compressed: &5.

Explanation: Displays the total number of files found in the archive (&1). Displays the total bytes uncompressed(&3) and the total bytes compressed(&4) for the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0840-00

IFS: Code Page &1.

Explanation: Displays the Code Page of a (Integrated File System) File in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0841-00

IFS: Creation Time &1.

Explanation: Displays the Creation Date and Time of a (Integrated File System) File in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0842-00

IFS: Access Time &1.

Explanation: Displays the Last Access Date and Time of a (Integrated File System) File in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0843-00

IFS: Modification Time &1.

Explanation: Displays the Last Modification Date and Time of a (Integrated File System) File in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0844-00

Lib Text Desc: &1.

Explanation: Displays the text associated with a library that contains the file that has been stored within the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0845-00

File Text Desc: &1.

Explanation: Displays the text associated with a file that has been stored within the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0846-00

Mbr Text Desc: &1.

Explanation: Displays the text associated with the member of the file that has been stored in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0847-00

Record Length: &1.

Explanation: Displays the record length of the file in the Archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0848-00

Source or Data: &1.

Explanation: Displays whether the file stored in the archive is a PF-DTA file, or is a PF-SRC file type. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0849-00

Source Type: &1.

Explanation: Displays the source type, for example, CLP, RPG, CMD, etc., for a PF-SRC member. Displays Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0850-00

Unknown Database type: &1.

Explanation: The Extended Data field specifying the file is a database or is not invalid. Database options ignored. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0851-00

Database File Attributes-.

Explanation: This indicates that a file contains Database File attributes in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0852-00

AUT(&1) MAXMBRS(&2) DLTPCT(&3) SIZE(&4 &5 &6) ALLOCATE(&7) CONTIG(&8) LVLCHK(&9) REUSEDLT(&10) Access path type: &11.

Explanation: Displays Database File Attributes:- File Authority:
&1 Maximum no. members in file:
&2 Max % deleted records in file:
&3 Initial size of member:
&4 Size of increment:
&5 Max no. increments:
&6. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0853-00

Format Name(&1) No. Flds(&2) No. Key Flds(&3).

Explanation: Displays Database File Field Format Attributes: - File Format Name: &1 Number of fields in format:

&2 No. key fields in format:
&3 Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0854-00

-Field descriptions:-

Explanation: Displays that a heading of Field descriptions will follow. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0855-00

Num, Name, Width Gen. U D N V K CCSID.

Explanation: DB Field Format Attributes: - Num : Field Number Name : Internal Field name Width : Field width (Number of digits/characters) Gen. : Decimal Pos./Allocated Length/Time or Date format U : Usage (B = Both, I = Input, O = Output, N = Neither D : Field Data type N : Null field capable indicator V : Variable length field indicator K : Key Field indicator CCSID : CCSID of field (N/A numeric fields). Information with TYPE(*VIEW)/VIEWOPT

AQZ0856-00

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

Explanation: Displays heading separator for the Field Description details. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0857-00

&1 &2 &3 &4 &5 &6 &7 &8 &9 &10.

Explanation: This field has the following attributes: - Num : Field Number = &1 Name : Internal Field name = &2 Width : Field width = &3 General: = &4 U : Usage = &5 D : Field Data type = &6 N : Null capable = &7 V : Variable length field = &8 K : Key Field = &9 CCSID : CCSID of field = &10. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0858-00

Text: &1.

Explanation: Displays the Fields text description for this field is: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0859-00

Alias: &1.

Explanation: Displays the alternative field name (alias) for this field: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0860-00

Default: &1.

Explanation: Displays the default value for the field is: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0861-00

Col Heading 1: &1.

Explanation: Displays the First column heading for the field is: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0862-00

Col Heading 2: &1.

Explanation: Displays the Second column heading for the field is: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0863-00

Col Heading 3: &1.

Explanation: Displays the Third column heading for the field is: &1. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0864-00

MAINT(&1) RECOVER(&2) FRCACCPTH(&3) ACCPTHSIZ(&4) Order: &5

Explanation: Displays the Keyed access path attributes: - Order = Duplicate key sorting order. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0865-00

File text: &1.

Explanation: Displays the text associated with the file that has been stored in the archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0866-00

Archive identified as a GZIP archive.

Explanation: Archive identified as a GZIP archive. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0867-00

VMS file attributes (&1 octal): 2.

Explanation: If present for a file in the archive, displays VMS file attributes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0868-00

Amiga file attributes (&1 octal): &2.

Explanation: If present for a file in the archive, displays Amiga file attributes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0869-00

Unix file attributes (&1 octal): &2.

Explanation: If present for a file in the archive, displays UNIX file attributes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0870-00

Non-MSDOS external file attributes: &1 hex.

Explanation: If present for a file in the archive, displays non-MSDOS external file attributes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0871-00

MS-DOS file attributes (&1 hex): &2.

Explanation: If present for a file in the archive, displays MS-DOS file attributes. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0872-00

Advanced Encryption &1. &2.

Explanation: Displays the Advanced Encryption method the file was archived. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0873-00

Algorithm Key &1, Security type &2.

Explanation: Displays the Advanced Encryption algorithm key text description and the Security type (Password, Certificate Signature, or Combination Password with Signature) that was used to encrypt the file. Information with TYPE(*VIEW) and VIEWOPT(*DETAIL).

AQZ0874-00

Spool File Type: &1, Target File:&2, (CdPg=&3), Nbr Of pages(&4).

Explanation: Displays the Spool File type, the Target Format File Type, and the number of pages in the Spool File. If the target is Text or PDF then (CdPg=&3) will be displayed to show the code page that was used to translate the EBCDIC to ASCII.

AQZ0875-00

SPLF Desc: &1.

Explanation: Displays the spool file name attributes with / separation with format: **"**Job-Name**/**User-Name**/#**Job-Number**/**Spool-File-Name/Fspool-File-Number.Suffix".

AQZ0876-00

OS390 File Type &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0877-00

OS390 Record Format

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0878-00

OS390 Block Size &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0879-00

OS390 VSAM Record Size &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0880-00

OS390 VSAM Max Record Size &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0881-00

OS390 VSAM Cluster Name &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0882-00

OS390 VSAM Data Name &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0883-00

OS390 VSAM Index Name &1

Explanation: Z390 MVS or VSE Extended Attributes.

AQZ0884-00

ZIP64 Extended Information: Uncompressed size=&1; Compressed Size=&2

Explanation: Extended data to support large files. Original Uncompressed size=&1; Compressed Size=&2; Offset=&3.

AQZ0885-00

ZIP 64 Large File Support Found

Explanation: PKZIP.

AQZ0886-00

Archive has been Digitally Signed.

Explanation: The Archive contains a Digital Signature for security purpose.

AQZ0887-00

Spool File OUTQ(&1) and User(&2)

Explanation: Displays the OUTQ that the spool file was compressed from and the user Id of the Spool File

AQZ9xxx LICENSE Messages

AQZ9000-00

PKZIP for iSeries(TM) Compression Utility Version &1, &2.

Explanation: Displays the Version &1 of PKZIP for iSeries(tm) with version release date of &2.

AQZ9001-00

PKZIP for iSeries(tm) running under Beta release &1.

Explanation: Displays if PKZIP is running under an Alpha or a Beta Release.

AQZ9002-00

PKZIP is being terminated due license validation failure.

Explanation: PKZIP could not validate the License. Obtain proper Licensing keys from PKWARE, Inc., and run INSTPKLIC.

AQZ9003-00

Machine ID = &1, Processor Group = &2.

Explanation: Displays the Machine CPU Serial number and the Processor Group Information.

AQZ9004-00

EVALUATION Running.

Explanation: PKZIP is running under DEMO and Evaluation mode. A License Key needs to be obtained.

AQZ9005-00

Enterprise Registered.

Explanation: ***PKZIP for iSeries (tm)***is Licensed with Enterprise Registered.

AQZ9006-00

Registered.

Explanation: ***PKZIP for iSeries***(tm)has been registered with a valid License Key.

AQZ9007-00

Copyright. 2003 PKWARE of Ohio, Inc. All Rights Reserved.

Explanation: ***PKZIP for iSeries(tm)*** Banner lines. Displays when PKZIP or PKUNZIP starts.

AQZ9008-00

THIS LICENSE HAS EXPIRED. To renew your license.

Explanation: During PKZIP start up processing, it was determined that your license has expired. See Following Messages on how to update your Licenses.

AQZ9009-00

Contact your dealer with the following information.

Explanation: Informing that the next message provides information required to obtain a license update.

AQZ9010-00

Make sure you have run the Install License program INSTPKLIC.

Explanation: A problem has occurred when trying to open and process the licensing. This is probably due to not running the Install License program.

AQZ9011-50

Could not open License file for <&1> -aborting &2.

Explanation: PKZIP or PKUNZIP could not open the License file. Job will terminate. This is probably due to not running the Install Program.

AQZ9012-50

Could not read License file base for <&1>, Length returned =&2.

Explanation: An error occurred while reading the PKZIP Licensing File. Review Job Log for more details. Job is being terminated.

AQZ9013-50

Could not read License file Feature for <&1>, Length returned =&2.

Explanation: An error occurred while reading the PKZIP Licensing File. Review Job Log for more details. Job is being terminated.

AQZ9014-00

&1, Warning - This license will expire in &2 days on &3.

Explanation: A warning that your license will expire soon. Contact your supplier to obtain a new license.

AQZ9015-00

UNREGISTERED Copy Running,

Explanation: ***PKZIP for iSeries*** is running without a valid license. It will run temporarily for 30 days. Contact your supplier to obtain a new license with your Serial Number and Processor Group.

AQZ9016-00

Unregistered Hardware Found Exception. &1 Grace days allowed.

Explanation: Serial Number and/or Processor Group was not found for ***PKZIP for iSeries***. A temporary license will be granted for &1 days. Contact your supplier to obtain a new license with your Serial Number and Processor Group.

AQZ9017-00

PKZIP (R) is a registered trademark of PKWARE (R), Inc.

Explanation: Trademark notes.

AQZ9018-40

Beta Release Has Expired Contact PKWARE, Inc. for Final Beta Release.

Explanation: You are running a Beta Version of PKZIP for iSeries(tm) that has expired. Contact PKWARE, Inc., for a newer version of PKZIP for iSeries (TM).

AQZ9050-50

Could not create License file - aborting.

Explanation: While running INSTPKLIC, the licensing did not exist, so one needed to be created. An error occurred and INSTPKLIC could not be created. See Job Log for more detail information. Job will be terminated.

AQZ9051-50

Could not open License file for update-aborting.

Explanation: ***PKZIP for iSeries*** could not open the License file for updating. Job will terminate. See Job Log for details.

AQZ9052-50

Could write License file for update-aborting.

Explanation: ***PKZIP for iSeries*** could not write to the License file with updates. Job will terminate. See Job Log for details.

AQZ9054-50

Input File <&1> cannot be open for input.

Explanation: The PKZIP License Install program could not open the "Library/File(member)" - &1 input file that contains the licensing update keys. Job is terminated. See Job Log for more Details.

AQZ9055-50

Error Reading Input Control file <&1>.

Explanation: The PKZIP License Install program could not read the "Library/File(member)" - &1 input file that contains the licensing update keys. Job is terminated. See Job Log for more Details.

AQZ9056-00

&1 Evaluation set to expire in &2 days on &3.

Explanation: Feature Code &1 has been set for Evaluation and will expire in &2 Days on the date of &3.

AQZ9057-00

Nbr of Feature Control Records Type-&1 was &2.

Explanation: The total of Inputted Feature control records with type &1 was processed. This is only with debug turned on.

AQZ9058-00

Rec - &1 &2.

Explanation: Displays Detail input control records being processed by INSTPKLIC (includes * comments).

AQZ9059-00

License File &1 Updated successfully.

Explanation: The PKZIP License File &1 was updated successfully with supplied control records.

AQZ9060-50

License file NOT Updated.

Explanation: Due to errors wile running the License Install program, the License file was not updated. Review Job Log for previous message, which indicates why the file was not updated.

AQZ9061-40

Error found in processing input parameters.

Explanation: Errors were found while processing the input control records in the License Install program. License file was not updated. Review Job Log for previous message, which indicates why the file was not updated.

AQZ9062-40

More than 1 control code 55 record, Run terminated.

Explanation: Only one Customer Control Record beginning with 55 is allowed in the License Control Program. Job will be terminated. Review Input Control file for control records.

AQZ9063-40

Control Record 55 must be 1st non-comment record.

Explanation: The first control record (non-comment) in the License Control input file must the Customer Control Record beginning with 55. Job will be terminated.

AQZ9064-40

Invalid Feature Code <&1> in Position 1 & 2.

Explanation: In positions 1 and 2 of the License control record is the Feature Code. The License Install Program found the invalid Feature code of &1. The License File will not be updated. (See Licensed Product Features in Chapter 3. - *PKZIP for iSeries* Installation).

AQZ9065-40

Invalid Date <&1> on record &2.

Explanation: An Invalid Feature Date on control record number &2 was found. Date must be YYYYMMDD format with valid month and day. The License File will not be updated.

AQZ9066-40

An error has occurred in validating the Customer Number &1 - Contact your Dealer.

Explanation: The Customer Number &1 was found to be Invalid. Contact your dealer for correct codes. The License File will not be updated.

AQZ9067-40

An error has occurred in validating the License - Contact Your Dealer.

Explanation: The validation of the license was found to be Invalid. Contact your Dealer for correct codes. The License File will not be updated.

AQZ9068-40

An error has occurred in validating the License. Contact Your Dealer.

Explanation: The validation of the license was found to be Invalid. Contact your Dealer for correct codes. The License File will not be updated.

AQZ9069-40

Customer name cannot be blank for Control 55.

Explanation: In the Customer Control record (starts with 55), the Customer Name should contain the Customers Name in Positions 23 thru 72 and must not be blank.

AQZ9070-40

Duplicate hardware<%s> found for Feature %2.

Explanation: Only one unique hardware (Serial Number and Processor Group) is allowed per Feature Code. The License File will not be updated.

AQZ9071-40

Embedded blanks in hardware <&1> for feature &2.

Explanation: The Hardware codes (Serial Number and Processor Group) must all be non-blank characters. The License File will not be updated.

AQZ9072-40

Max hardware exceeded &2 for Feature &2.

Explanation: All Hardware available entries are in use. There is no room to add another hardware entry for the product. Contact your Dealer.

AQZ9073-40

&1 Request to setup DEMO rejected. DEMO is already running.

Explanation: A request in the License Install Program to setup a Demo license was issued, but a DEMO is currently active. A DEMO request cannot be run until current DEMO expires. The License File is not updated.

AQZ9074-40

Cannot do VIEW option until file has been Installed.

Explanation: A request in the License Install program to view the current license setting was issued, but the VIEW option cannot be used until one valid install is processed.

AQZ9075-40

Invalid or Missing Member Name.

Explanation: Special characters or a blank member name was submitted in the command INSTPKLIC for parameter INMBR.

AQZ9076-40

Feature Code &1 is invalid for Customer Number &2.

Explanation: The Customer Number &2 on the input control record is invalid for the specified Feature Code &1. Contact your Dealer for correct codes. The License File will not be updated.

AQZ9077-40

License Keys have invalid version setting.

Explanation: The version inputted keys are a mismatch to PKZIP for iSeries version. Review input file for keys.

AQZ9079-00

****

Explanation: Displays a Print Separator for INSTPKLIC viewing.

AQZ9080-00

A License Report requested on &1 from CPU Serial# &2.

Explanation: The Licensing Install program was run with TYPE(*VIEW), and the report of the current status of the License will be following for this CPU.

AQZ9081-00

&1 Product Licensed to Customer # &2 -&3.

Explanation: Displays the Version, Customer Number and Customer Name currently on license file. Information of INSTPKLIC with TYPE(*VIEW).

AQZ9082-00

&1-DEMO with &2 Days remaining (&3).

Explanation: Displays the hardware &1 is running as a demo and has &2 days remaining. Information of INSTPKLIC with TYPE(*VIEW).

AQZ9083-00

Enterprise Licensed. All Products Features are available on all CPUs. Expiration Date is &1.

Explanation: Display that all CPUs are being run with the Enterprise license; also displays the expiration date. Information of INSTPKLIC with TYPE(*VIEW).

AQZ9084-00

&1 Licensed --Expires &2 for processors:.

Explanation: Displays the Feature Codes (&1) and their expiration date(&2) that is on current License File. Information of INSTPKLIC with TYPE(*VIEW).

AQZ9085-00

Serial# &1 Processor Type &2.

Explanation: Displays the list of hardware that is active under the Feature Code (AQZ90084). Information of INSTPKLIC with TYPE(*VIEW).

AQZ9086-00

Serial# &1 is Not Licensed. Use Of The Product will CEASE in &2 days (&3).

Explanation: Displays the hardware that is not currently licensed for the above Feature Code (AQZ9084) and display when it will expire. Information of INSTPKLIC with TYPE(*VIEW).

AQZ9087-00

Contact PKWARE of Ohio, Inc. for Licensing.

Explanation: Informational message to contact your support for Licensing Update. Please contact the Product Services Division at 1-937-847-2687 for assistance.

AQZ9100-40

Compression Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Compression License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9101-40

Decompression Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Decompression License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for contact information.

AQZ9102-40

GZIP Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the GZIP License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9103-40

IFS File Handlers Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the IFS File Handlers License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9104-40

Database File Handlers Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Database File Handlers License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9105-40

Advance Encryption Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Advance Encryption License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9106-40

Spool File Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Spool File License Feature. Obtain proper Licensing keys from your reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9107-40

Beta Release Has Expired Contact PKWARE, Inc. for Final Beta Release.

Explanation: You are running a Beta Version of PKZIP for iSeries that has expired. Contact PKWARE, Inc., for a newer version of PKZIP for iSeries.

AQZ9108-00

Large File Support Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP/PKUNZIP encountered an action that re quires the Large File Support feature. Reason code &1. The Large File Support Feature could not validate this Licensed Feature. Obtain proper Licensing keys from reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

AQZ9109-00

Self Extraction Support Feature is not licensed for this Serial Number. Run will not be processed.

Explanation: PKZIP could not validate the Self Extraction Support Feature. Obtain proper Licensing keys from reseller and run INSTPKLIC. See Message AQZ9087 for more contact information.

Appendix A - Performance Considerations

This Appendix lists a few performance considerations when running ***PKZIP for iSeries***. Most performance related issues can be controlled by the PKZIP/PKUNZIP parameters. However, it should be noted that PKZIP data compression is CPU intensive by its very nature, and that PKZIP/PKUNZIP parameters can only help so much. Therefore, it should be expected that a *reasonable* amount of CPU resources will be needed for such operations.

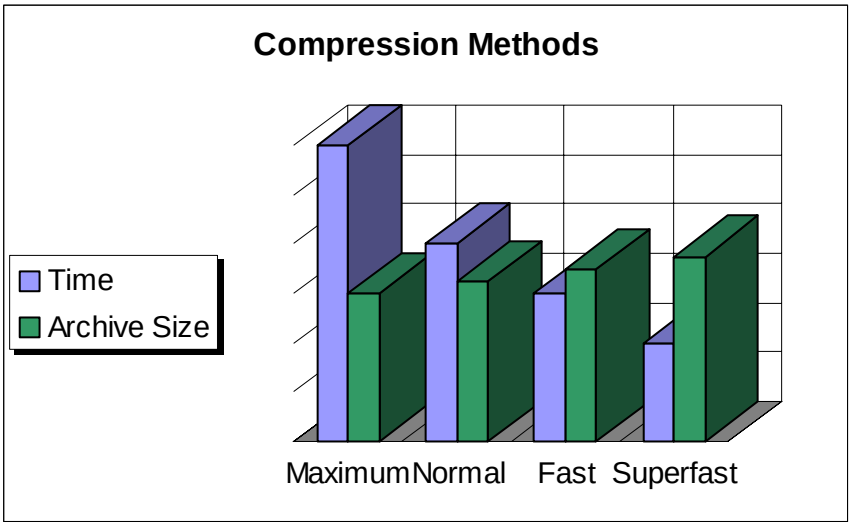
Interactive Performance

When compressing large size files, PKZIP will sometimes use as much CPU as the system will allow. With this in mind, processing very large files may perform best as a submitted job. Yet, some iSeries environments have constraints on running interactive jobs. If those interactive jobs run long in duration and use a high amount of CPU, the system will start slowing down and may issue the message CPI1479 "Interactive activity approaching capacity of installed feature". In this case, review the details of this message. This usually means that the interactive systems are using more resources than the iSeries was configured to use.

Compression Type Performance

Selecting a compression method is a way to get the smallest compressed file with the relationship to the CPU usage and run times. Sometimes, to get the best results, you may have to run several tests with the data to balance the compression ratio to the length of the run time. Running with *MAX will usually get the best compression ratio but will also run the longest. In most of our test cases, *MAX would run 30%-40% longer than *NORMAL and might only gain less than 1% better ratio. This is why we recommend using SUPERFAST (the default) unless your testing implies otherwise.

To minimize the overhead needed to ZIP, the best thing (and the easiest) is to select a compression method other than *MAX. PKZIP's default compression method is SUPERFAST. Below you will see a graph that properly depicts how the COMPRESSION METHOD parameter works.



As you can see from the above bar graph, when using the compression method of Maximum, you are only compressing the data by another 1-8% over a job that might use the SUPERFAST compression method. The archive file size change is minimal. However, the time difference between a Maximum and a SUPERFAST job can be measured in hours if the file is big enough!

You may read more about the compression levels by prompting the Compression Level parameter (F1).

```
Compression Level  . . . . . *SUPERFAST    *FAST, *NORMAL, *MAX...
```

Data Type Selection

Getting the best performance from your iSeries machine with regards to a PKZIP job can truly depend on the parameters you have selected for the job. In many cases, the compressed size of a file depends on the type of data (Binary vs. Text), and the compression type selected. Text will usually compress more since it has a higher probability of repeated characters.

Knowing the target platform of the data will help you resolve how PKZIP is to treat the data during the compression process. However, PKZIP treatment of data defaults to *DETECT. *DETECT means that PKZIP will scan the data (up to 97% of the input file) to determine whether the data that it is going to compress should be treated as TEXT or BINARY. This can be an especially painful process if you are selecting large files for compression. However, to get around the *scanning* overhead, if you know you are sending the archive or ZIP file to a PC or to a UNIX box, you know that the data will need to be converted to TEXT (or ASCII). Therefore, you should select File Types(*TEXT). If the data is targeted for another iSeries machine, then you should select *BINARY. *DETECT should only be used when you do not know the nature of the data.

You may read more about the data types by prompting the File Types parameter (F1).

```
File Types . . . . . .   *DETECT  *DETECT *TEXT *BINARY ....
```

Archive Placement (IFS or in a Library)

For best performance try to store the archives in the IFS. By placing the archive in the IFS instead of in a library/file reduces the overall CPU usage and in some cases can reduce the run times as much as 30%-40%.

It is recommended when using the ZIP process for large files that the ZIP Archive be stored in the IFS. This method provides the best performance and makes the most efficient use of storage space for both ZIP Archives and ZIP temporary files.

ZIP64 Processing Considerations

When processing very large files or high volumes of files, the processing characteristics of PKZIP may vary depending on the phase of processing involved. Some common processing phases and their run-time characteristics are:

- ZIP File Selection: When selecting a very large number of files thru many directories and/or libraries, the initial selection requires IO time and memory per file to analyze and manage each of the file's properties. The more files to select, the more memory and initial startup overhead. Each site will have to discover their practical limits based on their environments and resources.
- ·Archive Directory Read processing: When updating an existing archive that contains a very large number of files, time and memory again are used to manage the archives and its directory. Or when using PKUNZIP to view the files, the more files in the archive, the more memory is required and the more time involved when sorting the files in archive properties before displaying or printing the contents.
- ·Archive updating: When updating a large archive with large file sizes, there will be overhead to copy the files from the previous archive, before adding or updating new files to the archive. For example, if you have a 10 GB archive with 5 files that are each compressed, down to 2 GB, overhead will be required to copy the compressed files from the old archive to the new archive resulting in time to copy. This is another reason for storing the archive in the IFS, which can help reduce resources rather than storing the archive in a file in a library.
- ·When compressing large size files, PKZIP will sometimes use as much CPU as the system will allow. With this in mind, processing very large files may perform best as a submitted job. Some iSeries have constraints on running interactive and if interactive jobs run long in duration and use high amounts of CPU interactive, their system will start slowing down and may issue the message CPI1479 "Interactive activity approaching capacity of installed feature." In this case they should review the details of this message, which usually means that their interactive systems are using more resources than the iSeries was configured to use.

Encryption Performance

When using Advanced Encryption versus no encryption, there will be a slight increase in the overall size of the Archive that contains the AES overhead (maybe around 300 bytes per file in archive). The increase in size will be same whether you use AES 128, AES 192, or AES 256.

AES 256 being the most secure encryption algorithm, will also consume the most CPU usage. AES 128 on average could use around 9% more CPU than running with no encryption. AES 256 averages about 3.4% more usage when compared with AES 128 (or around 12.5% versus no encryption).

Extended Attributes Selections

The extended attributes naturally attribute some overhead to the archive but is minimal, unless you are compressing a database file in the QSYS Library file system with the parameter DBSERVICE(*YES). This size then depends on the definitions of the database (fields, headings, etc), but also is very important in rebuilding a DB2 database where it does not exist.

These extended attributes can be stored in two places called the Local Header and Central Header directories. PKZIP for iSeries 5.6, and other PKWARE products now only use the extended attributes from the Central directory. PKZIP for OS/400 5.5 required some of the attributes in the Local Header.

To help reduce the archive overhead the parameter EXTRAFLD in PKZIP has been expanded to select where you want to store the attributes. By using EXTRAFLD(*Central), you reduce the size of each file in the archive by the size of the extended attributes. **One word of caution is if the attributes are required on another iSeries not running on 5.6 or above, you should use the option EXTRAFLD(*BOTH) or EXTRAFLD(*YES).**

Appendix B - Examples

Example 1 - PKUNZIP Files to a New or Different Library

To extract the files in the archive to a new library. An example of the files in the archives are:

```
PKUNZIP ARCHIVE('atest/qz/tstchg')
Archive:  ATEST/QZ(TSTCHG)   88572 bytes   6 files
 Length  Method   Size  Ratio   Date   Time  CRC-32    Name
-------- ------ ------- -----   ----   ----  ------    ----
      259  Defl:F      178  01-16-01 08:24 b5dbf80c TESTLIB1/BEN/BEESON
   449664  Defl:F    48720  01-16-01 14:46 94c7506c
TESTLIB1/MYSPLFTM.P/AQZIP123.4X
   205693  Defl:F    29687  01-16-01 14:46 e2473ea4
TESTLIB1/MYSPLFTM.P/LISTDBOB.X
    27488  Defl:F     6771  01-16-01 14:46 c264817a TESTLIB1/MYSPLFTM.P/QPRINT1X
     3352  Defl:F      800  01-16-01 14:46 3e485445
TESTLIB1/MYSPLFTM.P/T4RSTLIB.XY
      256  Stored      256  01-16-01 15:22 29058c73 TESTLIB1/TEST1/HEX
 --------          -------  ---                            -------
   686712            86412                              6 files
```

To extract to a new Library, use the keyword EXDIR and DROPPATH

```
PKUNZIP ARCHIVE('atest/qz/tstchg') TYPE(*EXTRACT) EXDIR(mynewlib) DROPPATH(*LIB)

     PKUNZIP Archive: ATEST/QZ(TSTCHG)
     Searching Archive ATEST/QZ(TSTCHG)  for files to extract
     Extracting file TESTLIB1/BEN/BEESON                      NOTE path
     Library MYNEWLIB created.                                NOTE Library created
     File BEN created in library MYNEWLIB.
     Member BEESON added to file BEN in MYNEWLIB.
     Member BEESON file BEN in MYNEWLIB changed.
     Inflating: MYNEWLIB/BEN(BEESON)     Text                 NOTE new path
     Extracting file TESTLIB1/MYSPLFTM.P/AQZIP123.4X
     File MYSPLFTMP created in library MYNEWLIB.
     Member AQZIP1234X added to file MYSPLFTMP in MYNEWLIB.
     Member AQZIP1234X file MYSPLFTMP in MYNEWLIB changed.
     Inflating: MYNEWLIB/MYSPLFTMP(AQZIP1234X)     Text
```

Since the library mynewlib did not exist, it was created.

Example 2 - CLP with Override for Stdout and Stderr to an OUTQ

The following is an example of overriding the PKZIP and PKUNZIP program output, and then redirecting the output to an OUTQ. This also provides an example of using mixed file systems, such as having the archive file in the IFS and selecting files from the QSYS Library file system.

```
 ZIPEXPL01:   PGM        PARM(&OUTQ)
/* Program:   ZIPEXPL01            Example                          */
/*************************************************************/
/* Abstract: This is a example CL program has on parameter for   */
/*    the OUTQ for the processing of PKZIP for iSeries.  If it is     */
/*    *none or *NONE no overriding to QPRINT will take place.    */
/*    1. Will add the PKZIP for iSeries Library to Library List       */
/*       If it is already part of LIBL then note so as to not    */
/*        remove at the end.                                     */
/*    2. Set the Current Library (only required if parameters of */
/*        PKZIP leaves out the Library and a default is needed)  */
/*    3. An example of setting the current directory is IFS      */
/*    4. Check input OUTQ.  If none keep processing              */
/*    5. Override the Stdout and Stderr to input outq            */
/*        This is where PKZIP will send messages when the        */
/*        MSGTYPE is (*PRINT) or (*BOTH).                        */
/*    6. Run Test01 of PKZIP                                     */
/*    7. Run Test02 of PKZIP with archive in IFS                 */
/*    8. Run Test03 of PKZIP with IFS system                     */
/*    9. Run Test04 of PKUNZIP  to view archive                  */
/*   10. If the PKZIP Library was not present at the beginning   */
/*       remove it from *LIBL.                                   */
/*                                                               */
/*************************************************************/

 OUTQ:       DCL        VAR(&OUTQ) TYPE(*CHAR) LEN(10)
 PKZIPLIB:   DCL        VAR(&PKZIPLIB) TYPE(*CHAR) LEN(10) +
                          VALUE(PKZ560510)
   /* if PKZIP library is in Libl do not remove it at the end */
 LIBLCHG:    DCL        VAR(&LIBLCHG) TYPE(*CHAR) LEN(1) VALUE('Y')
 CURLIB:     DCL        VAR(&CURLIB) TYPE(*CHAR) LEN(10) VALUE(MYLIB)
 ZIPDIR:     DCL        VAR(&ZIPDIR) TYPE(*CHAR) LEN(10) +
                          VALUE('/mydir')
     /* Add the PKZIP for iSeries library to library List  */
             ADDLIBLE   LIB(&PKZIPLIB)
             MONMSG     MSGID(CPF2103) EXEC(CHGVAR VAR(&LIBLCHG) +
                          VALUE('N'))
     /* Set Current Directory to MYLIB  */
                /*(not Required for this test Just an example)*/
             CHGCURLIB  CURLIB(&CURLIB)
     /* Set Current Directory to zip     */
                /*(not Required for this test Just an example)*/
              CD         DIR(&zipdir)

     /* Check input outq to see overides required   */
CHKOUTQ:     IF         COND((&OUTQ *EQ '*none') *OR (&OUTQ *EQ +
                          '*NONE')) THEN(GOTO CMDLBL(NOOUTQ))
  /* change Stdout and Stderr to my outq  */
             OVRPRTF    FILE(STDOUT) TOFILE(*LIBL/QSYSPRT) OUTQ(&OUTQ)
             OVRPRTF    FILE(STDERR) TOFILE(*LIBL/QSYSPRT) OUTQ(&OUTQ)
NOOUTQ:
```

```
    /* Test basic PKZIP  */
TEST01:     PKZIP      ARCHIVE('ATEST/PKZ2(MYSL04)') +
                         FILES('TESTLIB/MYSPLF(*ALL)') +
                         EXCLUDE('TESTLIB/MYSPLF(Q*)')

    /* Test basic PKZIP with archive in IFS and print only messages  */
TEST02:     PKZIP      ARCHIVE('/mydir /tmpsave/itest01') +
                         FILES('TESTLIB/TEST') FILETYPE(*BINARY) +
                         TYPARCHFL(*IFS) MSGTYPE(*PRINT)

    /* Test PKZIP with all files in IFS  */
TEST03:     PKZIP      ARCHIVE('/mydir/tmpsave/itest02.zip') +
                         FILES('/mydir/test1/basetest') +
                         TYPARCHFL(*IFS) TYPFL2ZP(*IFS)

    /* Test PKUNZIP view of archive from test02  */
TEST04:     PKUNZIP    ARCHIVE('/mydir/tmpsave/itest01') +
                         TYPARCHFL(*IFS)  MSGTYPE(*PRINT)

    /* If PKZIP Library was added to LIBL then remove it */
ENDPGM:     IF         COND(&LIBLCHG *EQ 'Y') THEN(RMVLIBLE +
                         LIB(&PKZIPLIB))
            ENDPGM
```

Example 3 - Creating an Archive in Personal Folders (QDLS)

The following is an example of creating and processing the archive in the Document Library Services File System (QDLS). First, assume a folder in QDLS with a name of MYFOLDER where the archives will be stored. To view the folders, issue the command WRKLNK '/QDLS/*' (you could use WRKDOC and WRKFLR, but WRKLNK is better to use since PKZIP will be using /QDLS).

```
                          Work with Object Links

Directory  . . . . :   /qdls

Type options, press Enter.
  3=Copy   4=Remove   5=Next level   7=Rename   8=Display attribu
  11=Change current directory ...

Opt   Object link            Type             Attribute    Text
      .                      FLR
      ..                     FLR
      MYFOLDER               FLR
      QBKBOOKS               FLR
```

Run the PKZIP command:

PKZIP ARCHIVE('/QDLS/MYFOLDER/MYARCH1.ZIP') FILES('testlib/ben') TYPARCHFL(*IFS)

The suffix .ZIP was added to help identify the file as an archive file.

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/08/21
Copyright.  2003.  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP is a registered trademark of PKWARE, Inc.
EVALUATION Running
EVALUATION, Warning - This license will expire in 29 days on 2003/09/20
Contact your dealer with the following information
Machine ID =     , Processor Group =  P05
Scanning files for match  ...
Found  1 matching files
Compressing TESTLIB/BEN(BEESON) in TEXT mode
Add  TESTLIB/BEN/BEESON  --   Deflating (32%)
PKZIP Compressed 1 files in Archive /QDLS/MYFOLDER/MYARCH1.ZIP
PKZIP Completed Successfully
Press ENTER to end terminal session.
```

To see the file in the folders, run WRKLNK '/QDLS/MYFOLDER/*'

```
                          Work with Object Links

 Directory  . . . . :   /QDLS/MYFOLDER

 Type options, press Enter.
   3=Copy   4=Remove   5=Next level   7=Rename   8=Display attributes
   11=Change current directory ...

 Opt   Object link            Type             Attribute    Text
       .                      FLR
       ..                     FLR
       MYARCH1.ZIP            DOC
```

Next, to view the contents, run:

PKUNZIP ARCHIVE('/QDLS/MYFOLDER/MYARCH1.ZIP') TYPARCHFL(*IFS)

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/08/21
Copyright.  2003.  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP is a registered trademark of PKWARE, Inc.
Archive:  /QDLS/MYFOLDER/MYARCH1.ZIP   551 bytes   1 file
  Length  Method    Size  Ratio  Date   Time  CRC-32    Name
 -------- ------  ------- -----  ----   ----  ------    ----
      259 Defl:F      177  32% 11-27-00 15:32 b5dbf80c TESTLIB/BEN/BEESON
 --------         -------  ----                         -------
     259              177  32%                          1 file
PKUNZIP extracted     0 files
PKUNZIP Completed Successfully
Press ENTER to end terminal session.
```

Example 4 - Processing Archive on a CD (QOPT)

The following is an example of processing an archive that exists on a CD and using PKUNZIP to view or extract. Because the archive file is on a CD, and the file system QOPT controls the CD, this archive basically exists in the IFS.

First, check and ensure the archive is on the CD by doing a WRKLNK (you can use WRKOPTDIR, but using WRKLNK will show the actual paths required). Remember, the volume of the CD is also a directory in QOPT file system. If the file names are longer than 8, the file name will be changed, much like you see in DOS
systems. It will contain an "~" followed by a number for files found with excessive name lengths.

WRKLNK '/QOPT/*'

```
                          Work with Object Links

Directory  . . . . :   /QOPT

Type options, press Enter.
  3=Copy   4=Remove   5=Next level   7=Rename   8=Display attributes
  11=Change current directory ...

Opt   Object link            Type             Attribute    Text
      MYTESTLABEL            DDIR
```

The above screen shows that the volume label of the CD is "MYTESTLABEL". Using the "5" for the next level option, you can navigate through the directories. You will then see the files and directories on the root of the CD. For example:

```
                          Work with Object Links

Directory  . . . . :   /QOPT/MYTESTLABEL

Type options, press Enter.
  3=Copy   4=Remove   5=Next level   7=Rename   8=Display attributes
  11=Change current directory ...

Opt   Object link            Type             Attribute    Text
      ARCHIVE.ZIP            DSTMF
      GZIPPW.GAR             DSTMF
      OS_400~3.DOC           DSTMF
      PKZCVT~2.DOC           DSTMF
      PKZ55~1.SAV           DSTMF
      PKZ55~1.ZIP           DSTMF
```

To view the archive PKZ55~1.ZIP (which is really the long name PKZ560510.ZIP) contents, use PKUNZIP with *VIEW.

Use the command:

PKUNZIP ARCHIVE('/QOPT/MYTESTLABEL/PKZ55~1.ZIP') TYPARCHFL(*IFS) TYPE(*VIEW)

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/08/22
Copyright.  2003.  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.
Archive:  /QOPT/MYTESTLABEL/PKZ55~1.ZIP   1373026 bytes   1 file
```

```
  Length  Method    Size  Ratio  Date   Time  CRC-32    Name
 -------- ------  ------- -----  ----   ----  ------    ----
  6044544 Defl:N  1372902  77% 08-16-01 21:17 d73f09cf PKZ560510.sav
 --------         -------  ----                          -------
 6044544          1372902  77%                           1 file
PKUNZIP extracted     0 files
PKUNZIP Completed Successfully
```

Example 5 - Compressing files from a CD (QOPT)

Using the document (.DOC) files on the CD shown in Example 4, we can compress the files and store them in the archive in my archive library ATEST under the file V509 archives with an archive file member named CDTEST01.

PKZIP ARCHIVE('atest/v509/cdtest01') FILES('/QOPT/MYTESTLABEL/OS_400~3.DOC' '/QOPT/MYTESTLABEL/PKZCVT~2.DOC') TYPFL2ZP(*IFS)

```
PKZIP for iSeries(tm) Compression Utility Version   5.6,  2003/08/22
Copyright.  2003.  PKWARE of Ohio, Inc.  All Rights Reserved.
PKZIP (R) is a registered trademark of PKWARE (R), Inc.             Scanning
files for match  ...
Found  2 matching files
Compressing /QOPT/MYTESTLABEL/OS_400~3.DOC in BINARY mode
Add  /QOPT/MYTESTLABEL/OS_400~3.DOC  --   Deflating (77%)
Compressing /QOPT/MYTESTLABEL/PKZCVT~2.DOC in BINARY mode
Add  /QOPT/MYTESTLABEL/PKZCVT~2.DOC  --   Deflating (79%)
PKZIP Compressed 2 files in Archive ATEST/V509(CDTEST01)
PKZIP Completed Successfully
```

Because you would not be able to extract them to the CD, you may want to use the parameter STOREPATH(*NO) so that only file names OS_400~3.DOC and PKZCVT~2.DOC are stored in the archive.

Example 6 - Compressing CL with MSG Checking

The following brief example demonstrates using PKZIP in a CL passing the archives library, file, and member as variables and then monitoring for errors from the PKZIP run.

```
ZIPEXPL03:  PGM        PARM(&ZIPLIB &ZIPFILE &ZIPMBR)

/* Program:   ZIPEXPL03            Example                    */
/*************************************************************/
/* Abstract: This is a example CL program that has 3 paramters   */
/*    that specifies the archive's Library, File and Member.     */
/*    1. Will add the PKZIP for iSeries Library to Library List      */
/*       If it is already part of LIBL then note so as to not    */
/*        remove at the end.                                     */
/*    2. Build the archive file namefor PKZIP by concatenating   */
/*       the inputted library, file and member names.            */
/*    3. Compress all files in TESTLIB with PKZIP Command        */
/*    4. Monitor for error messages from PKZIP.                  */
/*        If errors send message.                                */
/*    5. If the PKZIP Library was not present at the beginning   */
/*       remove it from *LIBL.                                   */
/*                                                               */
/*************************************************************/
 /* &PKZIPLIB contains the current PKZIP library           */
            DCL        VAR(&PKZIPLIB) TYPE(*CHAR) LEN(10) +
                         VALUE(PKZ560510)
 /* if PKZIP library is in Libl do not remove it at the end */
            DCL        VAR(&LIBLCHG) TYPE(*CHAR) LEN(1) VALUE('Y')
 /* &ZIPLIB is Library where archive will be stored  */
            DCL        VAR(&ZIPLIB)  TYPE(*CHAR) LEN(10)
 /* &ZIPFILE is File for the archive                */
            DCL        VAR(&ZIPFILE) TYPE(*CHAR) LEN(10)
 /* &ZIPMBR is Member of the archive file           */
            DCL        VAR(&ZIPMBR)  TYPE(*CHAR) LEN(10)
 /* Archive file for PKZIP built with concatenation  */
            DCL        VAR(&ZARCHF)  TYPE(*CHAR) LEN(36)
 /* Add the PKZIP for iSeries library to library List  */
            ADDLIBLE   LIB(&PKZIPLIB)
            MONMSG     MSGID(CPF2103) EXEC(CHGVAR VAR(&LIBLCHG) +
                         VALUE('N'))

 /*Concatenate the libraries, files and members for PKZIP of the archive*/
            CHGVAR     VAR(&ZARCHF) VALUE(&ZIPLIB *TCAT '/' *TCAT +
                         &ZIPFILE *TCAT '/' *TCAT &ZIPMBR)

 /* Compress all files in the library TESTLIB  and     */
 /* store them in the archive specified in the         */
 /* calling of the CLP program                         */
 /* If messages AQZ0022 "PKZIP Completed with Errors." */
 /* or AQZ0012 "PKZIP ending with Nothing to do"       */
 /* are returned                                       */
 /* from PKZIP, send message indicating an error       */
 /* Occured.                                           */
            PKZIP      ARCHIVE(&ZARCHF) FILES('TESTLIB/*all(*all)') +
                         COMPRESS(*NORMAL) ARCHTEXT('This the text +
                         of the archive for example 3')
            MONMSG     MSGID(AQZ0022) EXEC(SNDPGMMSG MSG('PKZIP +
                         Ended with MONMSG for AQZ0022'))
            MONMSG     MSGID(AQZ0012) EXEC(SNDPGMMSG MSG('PKZIP +
```

```
                          Ended with MONMSG for AQZ0012'))

    /* If PKZIP Library was added to LIBL then remove it */
ENDPGM:     IF         COND(&LIBLCHG *EQ 'Y') THEN(RMVLIBLE +
                         LIB(&PKZIPLIB))
EOJ:        ENDPGM
```

Example 7 – Compressing Spool Files Samples

The following are several samples demonstrating the selection of spool file for compression.

Sample 1: Select a specific spool file (MYSPLFFILE) for the specific job (jobname-WSSSPL, User-WSS and job number 11) in all output Qs (the default of *SFQUEUE*) and convert the spool file to a PDF format *SFTARGET(*PDFLETTER)* to fit a letter format. Store the archive in the IFS with *TYPARCHFL(*IFS)* .

```
PKZSPOOL ARCHIVE('/pkzshare/bills/splftest01.zip') TYPARCHFL(*IFS)
         SPLFILE(MYSPLFILE) SFUSER(*ALL) SFJOBNAM(11/WSS/WSSSPL)
         SFTARGET(*PDFLETTER) SFTGFILE(*GEN1P)
```

Sample 2: Select all spool files belonging to users WSS and TAIT (*SPLUSERID*) that resides in the OUTQ QPRINTS (*SFQUEUE*) and compress them as spool files with *SFTARGET(*SPLF)*. This might be done to save the spool files for later review since this OUTQ is purged on a regular basic.

```
PKZSPOOL ARCHIVE('/pkzshare/bills/splftest02.zip') TYPARCHFL(*IFS)
         SFUSER(WSS TAIT) SFQUEUE(QPRINTS) SFTARGET(*SPLF) SFTGFILE(*GEN1)
```

.

Sample 3: Using the archive from sample 2, we want to restore or extract the spool files in order to print them again. Except in this case we want them to belong to the user MAS with *SPLUSERID* and place the spool files in the OUTQ MASQ (*SFQUEUE*) located in the library DEVPLIB.

```
PKUNZIP ARCHIVE('/pkzshare/bills/splftest02.zip') TYPARCHFL(*IFS)
        TYPE(*EXTRACT) SPLUSRID(MAS) SFQUEUE(DEVPLIB/MASQ)
```

Sample 4: Select the spool file QPRINTS (*SPLFILE*), spool file number 17 (*SPLNBR*), user MAS (*SFUSER*) and convert the file to a TEXT file with *SFTARGET(*TEXTFC).* In this case the file is needed to read into a PC program and the user wants the ANSI control characters in position 1 of each line.

```
PKZSPOOL ARCHIVE('/pkzshare/bills/splftest04.zip')  TYPARCHFL(*IFS)
    SPLFILE(QPRINTS) SFUSER(MAS) SPLNBR(17)
    SFTARGET(*TEXTFC) SFTGFILE(*GEN1P)
```

Sample 5: Now we want to extract the Text file created in sample 4 to one of our share drives areas ('/PKZSHARE/PCFILES') that our PCs can access. In this case the normal extraction would identify the file as a Text file and would convert it to EBCDIC. Since the file will be used by a PC program that is expecting the data to be in ASCII, we will have to extract the file as binary since the internal file is already in ASCII. By specifying FILETYPE(*BINARY), this ensures that no translation of the data takes place.

```
PKUNZIP ARCHIVE('/pkzshare/bills/splftest04.zip') TYPARCHFL(*IFS) TYPFL2ZP(*IFS)
TYPE(*EXTRACT) FILETYPE(*BINARY)
EXDIR('/PKZSHARE/PCFILES') DROPPATH(*ALL)
```

Example 8 – PKZSPOOL the last spool file of current Job

The following brief CLP example demonstrates using PKZSPOOL to compress to a PDF, only the LAST spool file that was written out by the current job.

```
 ZIPEXPL07:  PGM
/* Program:   ZIPEXPL07            Example                      */
/*************************************************************/
/* Abstract: This is an example CL program that perform several */
/* task that prints reports.  Then compresses only the LAST     */
/* spool file created to a PDF file in a archive                */
/*                                                               */
/*************************************************************/


/* display the properties of the files start with "i" in QIBM folder   */
             DSPLNK     OBJ('/QIBM/i*') OUTPUT(*PRINT) +
                          DETAIL(*EXTENDED) DSPOPT(*ALL)


 /* display all libraries that start with Q and print              */
             DSPOBJD    OBJ(*LIBL/Q*) OBJTYPE(*LIB) DETAIL(*BASIC) +
                          OUTPUT(*PRINT)



/* Compress the ONLY the last spool file created to a PDF file       */
             PKZSPOOL   ARCHIVE('/pkzshare/bills/PKZdata1.zip') +
                          SFJOBNAM(*) SPLNBR(*LAST) +
                          SFTARGET(*PDFLETTER) SFTGFILE(*GEN1P) +
                          TYPARCHFL(*IFS)
 ENDOFJOB:   ENDPGM
```

Example 9 - CL to submit a job to compress all spool files for a job to a PDF

The following brief example demonstrates using PKZIP in a CL passing the archives library, file, and member as variables and then monitoring for errors from the PKZIP run.

```
 ZIPEXPL08:  PGM
/* Program:   ZIPEXPL08            Example                        */
/*************************************************************/
/* Abstract: This is an example CL program that perform several */
/* task that prints reports.  Then submits a job at the end of   */
/* the job that will compress all spool files to PDF files. The  */
/* reason the job was submitted was to also compress the job log */
/* to a PDF                                                      */
/*                                                               */
/*************************************************************/
         /*    Current Job Name     */
             DCL        VAR(&MYJOBNM) TYPE(*CHAR) LEN(10) VALUE(' ')
         /*    Current Job User     */
             DCL        VAR(&MYJUSER) TYPE(*CHAR) LEN(10) VALUE(' ')
         /*    Current Job Number   */
             DCL        VAR(&MYJNBR) TYPE(*CHAR) LEN(10) +
                          VALUE('000000')
/* retrieve the job name, user, and job number for later use       */
             RTVJOBA    JOB(&MYJOBNM) USER(&MYJUSER) NBR(&MYJNBR)
/* this job to get a full job log                                  */
             CHGJOB     LOG(4 00 *SECLVL) LOGCLPGM(*YES)

/* display the properties of the files start with c in my folder   */
             DSPLNK     OBJ('/pkzshare/bills/c*') OUTPUT(*PRINT) +
                          DETAIL(*EXTENDED) DSPOPT(*ALL)
             DSPAUT OBJ('/pkzshare/BILLS') OUTPUT(*PRINT)


/* create an archive with one file                                  */
             PKZIP      ARCHIVE('/pkzshare/bills/PKZtest1.zip') +
                        FILES('/pkzshare/bills/chartest2.zip') +
                        TYPARCHFL(*IFS) TYPFL2ZP(*IFS) STOREPATH(*NO)

/*  display detail attributes for the new archive created  */
             DSPLNK     OBJ('/pkzshare/bills/PKZtest1*') OUTPUT(*PRINT) +
                          DETAIL(*EXTENDED) DSPOPT(*ALL)

/* submit a job to create all spool Files including the job log into a */
/*   PDF file and place them in  archive  PKZdata1                  */
             SBMJOB     CMD(PKZSPOOL +
                          ARCHIVE('/pkzshare/bills/PKZdata1.zip') +
                          SFJOBNAM(&MYJNBR/&MYJUSER/&MYJOBNM) +
                          SFTARGET(*PDFLETTER) SFTGFILE(*GEN1P) +
                          TYPARCHFL(*IFS)) JOB(&MYJOBNM) +
                          INLLIBL(*CURRENT)
 ENDOFJOB:   ENDPGM
```

Appendix C - Memory Errors

In some rare cases, ***PKZIP for iSeries*** may experience a memory error condition. While running ***PKZIP for iSeries*** the program allocates memory for sorts, buffers, and other storage requirements. There are diagnostic checks in the system to validate memory requests. If a request fails, the job will terminate and send a message to the message queue indicating a memory allocation failure. Some systems limit the memory allocation for certain user/jobs which may cause this problem. In some cases, programs can have what is called “memory leaks” where memory is never freed. Signing off or ending the job will free the allocated memory. If a message indicates that a memory allocation error occurred, retrieve the failed job log and contact the Product Services Division at 1-937-847-2687 for assistance.

Appendix D - External Name Conversion Program - CVTNAME

PKZIP for iSeries provides an exit that calls a compiled CLP program named CVTNAME. PKZIP can change the names of iSeries file names to a ZIPPED file name to be used in the archive. PKUNZIP can change a ZIPPED file name in an archive to an iSeries file name. This is activated by using the parameter CVTFLAG and not being set to *NONE. ***PKZIP for iSeries*** will call the first CVTNAME program found in the library list.

PKZIP for iSeries will call the program and pass three fields to CVTNAME and will expect a return field. The first field passed is a 256-byte field that contains the input name (in PKZIP, it is the iSeries name, and in PKUNZIP, it is the ZIPPED name in the archive) that can be changed. The second field passed is a 5-byte field that can be used to help code specific logic to control the name change process. The third field is up to 256 bytes of extended data from the command to provide more flexibility in controlling the conversion of file names.

The program will pass back up to 256 bytes of a file name that will be used as the output name in the archive.

The exit can be divided into different conversion routines by assigning each routine a 5 character name, which is passed in as the CVTFLAG parameter value. This routine works for both the PKZIP and PKUNZIP programs, but normally you will need a different conversion routine or CVTFLAG for archiving or extraction. When the CVTNAME returns, the returned name should either hold the new name, or remain blank to indicate that the default conversion rules should be used.

Note: The PKZIP and PKUNZIP programs perform no validity checking on the name that is passed back and assumes that the name is valid and unique. If the name is not valid or unique, open errors or missing files may occur.

For PKZIP, the name exit receives the OS/400 file name and the returned name is the name that will be used for the file name in the archive. Ensure that names returned are unique; otherwise, only the first of the duplicated files will be compressed.

For PKUNZIP, the CVTNAME exit receives the name of the file that is in the archive and expects to return the name of a file on the OS/400 as ‘library/file(member)’. If this is invalid, then file open errors will occur. If they are valid but did not exist before, PKUNZIP will create the library, file, and member.

In the supplied library, there is a sample CVTNAME CLP that contains logic for the following flags:

- If the flag is *400, there is no conversion taking place, and the input name is passed back as a new name.
- If the flag is O4MS, the program will convert an iSeries file name “library/file(member)” to a MS-DOS name with ‘/’ (such as “library/file/member”).
- If the flag is MSO4, the program will take a MS-DOS file name “/dir1/dir2/dir3/name.xxx” and make it an iSeries name (such as “dir2/dir3{namexxx)”) where dir2 will be the library, dir3 will be a file name, and namexxx will be up to a 10-byte character member name.
- If the flag is *MSD, the program will convert an iSeries file name "library/file(member)" to a MS-DOS name with suffix .TXT to the file "library/file.TXT".
- If the flag is *MSM, the program will convert an iSeries file name "library/file(member)" to an MS-DOS name with a suffix .TXT added to the member name, such as "library/file/member.TXT".
- If the flag is *IFS, the program will format a file name "library/file(member)" for the iSeries QSYS.LIB Integrated File System. For example "/QSYS.LIB/LIBRARY.LIB/FILE.FILE/MEMBER.MBR".

- If the flag is SPLF1, the program will take a spool file name "job-name/useriud/jobnumber/splfname/splfnbr.suffix" build each in it own DCL field. Then taking the first 10 bytes of the CVTDATA passed from the command will build a new name that looks like “CVTDATA.splfnbr.suffix”.

Of course the correct flag should be used with the appreciate file system.

Changes can be made to CVTNAME to fit the customer's site requirements. The use of (and changes to) the CVTNAME program is the customer's responsibility.

The following is an example of the CVTNAME CLP source code included in the ***PKZIP for iSeries*** library:

Sample CVTNAME

```
/* Program:   CVTNAME              Sample VERSION 5.6.0          */
/*************************************************************/
/* Abstract: This is a sample CL program that can be used by     */
/*    PKZIP for iSeries product as an exit program to change the */
/*    names from iSeries to an archive name and visa versus.     */
/*    There are 4 parameters:                                    */
/*     1. The input name that is supplied by PKZIP or PKUNZIP    */
/*        to this program to analyze (up 255 bytes).             */
/*     2. The Name that this program can change and passes back  */
/*        to the calling programs to use (up to 255 bytes).      */
/*        If this is passed back with the 1st position blank,    */
/*        PKZIP/PKUNZIP will revert back to settings of the      */
/*        CVTTYPE parameter.                                     */
/*     3. A 5 byte field that represents a control field or      */
/*        or flags, that controls how CVTNAME should process the */
/*        name.                                                  */
/*     4. A 255 byte field that is user defined in the command to*/
/*        provide more information to assist controlling the     */
/*        logic in processing the name.  For Example it may      */
/*        time stamp prefix of the file names                    */
/*                                                               */
/*     This sample CVTNAME CLP has three flags accepted:         */
/*                                                               */
/*     1.  If the flag is *400 there are no conversion taking    */
/*     place and the input name is passed back as new name.      */
/*     2.  If the flag is O4MS,  the program will convert an     */
/*     iSeries name "library/file(member)"  to a MS-DOS          */
/*     name with '/' such as  "library/file/member".             */
/*     3.  If the flag is MSO4, the program  will take a MS-DOS  */
/*     file name  "/dir1/dir2/dir3/name.xxx"  and make it an     */
/*     iSeries name such as  "dir2/dir3{namexxx)"  where dir2    */
/*     will be the library, dir3 will be a file name and         */
/*     namexxx will be up to a 10 byte character                 */
/*     member name.                                              */
/*     4.  If the flag is *MSD,  the program will convert an     */
/*     iSeries file name "library/file(member)"  to a MS-DOS     */
/*     name with suffix .TXT to the file "library/file.TXT".     */
/*     5.  If the flag is *MSM,  the program will convert an     */
/*     iSeries file name "library/file(member)"  to a MS-DOS     */
/*     name with suffix .TXT to the member name                  */
/*     such as "library/file/member.TXT".                        */
/*     6.  If the flag is *MST,  the program will convert an     */
/*     iSeries file name "library/file(member)"  to a MS-DOS     */
/*     name with suffix .TXT to the member name only             */
/*     such as "member.TXT".                                     */
/*     7.  If the flag is *IFS, the program will format a        */
/*     file name "library/file(member)" formatted for the        */
/*     iSeries QSYS.LIB Integrated File System. for example      */
/*      "/QSYS.LIB/LIBRARY.LIB/FILE.FILE/MEMBER.MBR"             */
/*                                                               */
/*     8.  If the flag is SPLF1, the program will format a       */
/*     spool file name formated as:                              */
/*     "jobname/useriud/jobnumber/splfname/splfnbr.suffix"       */
/*     and rebuild a new name with data from the CVTDATA         */
/*     parameter.                                                */
/*                                                               */
/*                                                               */
/*NOTE: PKZIP and PKUNZIP performs no validity checking on the   */
/*  name that is passed back and assumes that the name is valid  */
/*  and unique.  If the name is not valid or unique, open errors */
/*  or missing files may occur.                                  */
/*                                                               */
/*                                                               */
/*************************************************************/
 CVTNAME:    PGM        PARM(&INNAME &OUTNAME &CVTFLAGS &CVTDATA)
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*    Parameter  Program Variables                                   */
/*      Input    Name that is used examine for change              */
 INNAME:     DCL        VAR(&INNAME) TYPE(*CHAR) LEN(256)
/*      Changed name that is passed back to PKZIP for iSeries      */
 OUTNAME:    DCL        VAR(&OUTNAME) TYPE(*CHAR) LEN(256)
/*     Flags  from PKZIP for iSeries for controlling logic flow   */
 CVTFLAGS:   DCL        VAR(&CVTFLAGS) TYPE(*CHAR) LEN(5)
/*     data   from PKZIP command for controlling logic flow       */
 CVTDATA:    DCL        VAR(&CVTDATA) TYPE(*CHAR) LEN(256)
/*                                                                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*     Program Variables                                          */
/* iSeries QSYS file name represented like "libnm/filenm(mbrnm)" */
 LIBNM:      DCL        VAR(&LIBNM) TYPE(*CHAR) LEN(10) /* Library +
                          Name */
 FILENM:     DCL        VAR(&FILENM) TYPE(*CHAR) LEN(10) /* File +
                          name */
 MBRNM:      DCL        VAR(&MBRNM) TYPE(*CHAR) LEN(10) /* Member +
                          name */
             DCL        VAR(&IDX)    TYPE(*DEC)  LEN(3)
             DCL        VAR(&IDXPRV) TYPE(*DEC)  LEN(3)
             DCL        VAR(&LEN)    TYPE(*DEC)  LEN(3)
          /* DCL        VAR(&MSGVAR) TYPE(*CHAR) LEN(300)               */
          /* DCL        VAR(&MUVAR)  TYPE(*CHAR) LEN(6) VALUE('..OUT=') */

/*   Spool  file name represented like             */
/* "JOBNAM/USERID/JOBNBR/SPLFNAM/SPLFNBR.SPLSUFIX" */
 JOBNAM:     DCL        VAR(&JOBNAM) TYPE(*CHAR) LEN(10) /* Job +
                          Name     */
 USERID:     DCL        VAR(&USERID) TYPE(*CHAR) LEN(10) /* User +
                          Id   */
 JOBNBR:     DCL        VAR(&JOBNBR) TYPE(*CHAR) LEN(7) /* Job +
                          Number  */
 SPLFNAM:    DCL        VAR(&SPLFNAM) TYPE(*CHAR) LEN(10) /* Spool +
                          File name  */
 SPLFNBR:    DCL        VAR(&SPLFNBR) TYPE(*CHAR) LEN(5) /* Spool +
                          File Number  */
 SPLSUFIX:   DCL        VAR(&SPLSUFIX) TYPE(*CHAR) LEN(4) /* +
                          Suffix      */
/* blank out name uncase no hit */
             CHGVAR     VAR(&OUTNAME) VALUE('    ')
/*                                                                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*  Flag set to *400                                               */
             IF         COND(&CVTFLAGS *EQ '*400') THEN(DO)

             CHGVAR     VAR(&OUTNAME) VALUE(&INNAME)

             GOTO       CMDLBL(ENDCHG)
             ENDDO
/*                                                                  */
/* * * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * * */
/*  Flag set to O4MS                                               */
/*  Change iSeries  Library/File(Member)  to                       */
/*         MSDOS        library/File/Member                        */

 O4MS:       IF         COND((&CVTFLAGS *EQ 'O4MS') *OR (&CVTFLAGS +
                          *EQ '*MSD') *OR (&CVTFLAGS *EQ '*MSM') +
                          *OR (&CVTFLAGS *EQ '*MST')  +
                          *OR (&CVTFLAGS *EQ '*IFS')) +
                          THEN(DO)
             CHGVAR     VAR(&IDX)    VALUE(0)
             CHGVAR     VAR(&IDXPRV) VALUE(1)
/*    Find a Library     Note:max length must be 11  */
 DOLIB1:     CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(DOLIB1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&LIBNM) VALUE(%SST(&INNAME &IDXPRV &LEN))
/*    Find a File        Note:max length must be 11  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
```

```
 DOFILE1:    CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '(')       +
                          THEN(GOTO CMDLBL(DOFILE1))

             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&FILENM) VALUE(%SST(&INNAME &IDXPRV &LEN))
/*    Find a Member      Note:max length must be 11  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
 DOMBR1:     CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE ')')       +
                          THEN(GOTO CMDLBL(DOMBR1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&MBRNM) VALUE(%SST(&INNAME &IDXPRV &LEN))
             CHGVAR     VAR(&OUTNAME) VALUE(&INNAME)
  /* Save the new name lib/file/mbr */
DOOUT1:
 AMSD:       IF (&CVTFLAGS *EQ '*MSD')         +
               DO
                  CHGVAR VAR(&OUTNAME) VALUE(&LIBNM *TCAT '/' +
                         *TCAT &FILENM *TCAT '.TXT ')
                  GOTO       CMDLBL(ENDCHG)
             ENDDO
 AMSM:       IF (&CVTFLAGS *EQ '*MSM')         +
               DO
                  CHGVAR VAR(&OUTNAME) VALUE(&LIBNM *TCAT '/' +
                         *TCAT &FILENM *TCAT '/' *TCAT &MBRNM +
                         *TCAT '.TXT ')
                  GOTO       CMDLBL(ENDCHG)
             ENDDO
 AMST:       IF (&CVTFLAGS *EQ '*MST')         +
               DO
                  CHGVAR VAR(&OUTNAME) VALUE(&MBRNM +
                         *TCAT '.TXT ')
                  GOTO       CMDLBL(ENDCHG)
             ENDDO
 AIFS:       IF (&CVTFLAGS *EQ '*IFS')         +
               DO
                  CHGVAR VAR(&OUTNAME) VALUE('/QSYS.LIB/' *TCAT +
                         &LIBNM *TCAT '.LIB/' *TCAT &FILENM *TCAT +
                         '.FILE/' *TCAT &MBRNM *TCAT '.MBR')
                  GOTO       CMDLBL(ENDCHG)
             ENDDO

             CHGVAR     VAR(&OUTNAME) VALUE(&LIBNM *TCAT '/'      +
                        *TCAT &FILENM  *TCAT  '/'                 +
                        *TCAT &MBRNM   *TCAT ' ')
ENDO4MS:     GOTO       CMDLBL(ENDCHG)
          ENDDO    /* end of O4MS do flag */
/*                                                                */
/* * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * */
/*  Flag set to MSO4                                              */
/* Change a MSDOS file /dir1/dir2/../dirn/file.xxx  to            */
/*                       dir2/dirn(filexxx)  iSeries file         */

MSO4:     IF         COND(&CVTFLAGS *EQ 'MSO4') THEN(DO)
             CHGVAR     VAR(&IDX)    VALUE(0)
             CHGVAR     VAR(&IDXPRV) VALUE(1)
             CHGVAR     VAR(&LIBNM)  VALUE('          ')
             CHGVAR     VAR(&FILENM) VALUE('          ')
             CHGVAR     VAR(&MBRNM)  VALUE('          ')
/*    first find 1st blank and work backwards                     */
FINDLST2:    CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             if         COND(&IDX *GT 250) THEN(GOTO CMDLBL(ENDCHG))
             IF         COND(%SST(&INNAME &IDX 1) *NE ' ')       +
                          THEN(GOTO CMDLBL(FINDLST2))
/*    Find a Member      Note:max length must be 11  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX - 1)
 DOMBR2:     CHGVAR     VAR(&IDX) VALUE(&IDX - 1)
             IF         COND(&IDX *LT 1) THEN(GOTO CMDLBL(ENDCHG))
```

```
             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(DOMBR2))
             CHGVAR     VAR(&LEN) VALUE(&IDXPRV - &IDX)
             IF         COND(&LEN > 11)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(11))

             CHGVAR     VAR(&IDXPRV) VALUE(&IDX - 1)

             CHGVAR     VAR(&MBRNM) VALUE(%SST(&INNAME &IDXPRV &LEN))
          /* REMOVE . IF ANY AND FORCE LENGTH TO 10  */

/*    Find a File        Note:max length must be 10  */
 DOFILE2:    CHGVAR     VAR(&IDX) VALUE(&IDX - 1)
             IF         COND(&IDX *LT 1) THEN(GOTO CMDLBL(SETNAME2))

             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(DOFILE2))
             CHGVAR     VAR(&LEN) VALUE(&IDXPRV - &IDX)
             IF         COND(&LEN > 11)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(11))
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX - 1)
             CHGVAR     VAR(&FILENM) VALUE(%SST(&INNAME &IDXPRV &LEN))
/*    Find a Library        Note:max length must be 10  */
 DOLIB2:     CHGVAR     VAR(&IDX) VALUE(&IDX - 1)
             IF         COND(&IDX *LT 1) THEN(GOTO CMDLBL(ENDCHG))

             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(DOLIB2))
             CHGVAR     VAR(&LEN) VALUE(&IDXPRV - &IDX)
             IF         COND(&LEN > 11)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(11))
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX - 1)
             CHGVAR     VAR(&LIBNM) VALUE(%SST(&INNAME &IDXPRV &LEN))

  /* Save the new name lib/file/mbr */
SETNAME2:
             IF     COND(&LIBNM *NE ' ') THEN(DO)
                      CHGVAR     VAR(&LIBNM) VALUE(&LIBNM *TCAT '/')
                      ENDDO
             IF     COND(&FILENM *NE ' ')  THEN(DO)
                       CHGVAR     VAR(&FILENM) VALUE(&FILENM *TCAT '/')
                       ENDDO
             CHGVAR     VAR(&OUTNAME) VALUE(&LIBNM *TCAT &FILENM      +
                          *TCAT &MBRNM   *TCAT ' ')
ENDMSO4:     GOTO       CMDLBL(ENDCHG)
          ENDDO    /* end of O4MS do flag */


/*                                                                   */
/* * * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * * */
/*  Flag set to SPLF1  Sample                                        */
/* Change  "jobname/useriud/jobnumber/splfname/splfnbr.suffix"       */
/*      and build new name with 10 bytes of the CVTDATA field        */
/*      with a '.' separator followed by splfnbr.suffix              */
/*      for CVTDATA.splfnbr.suffix                                   */
/*  all fields from the Spool File passed file name are built        */
/*  care should be taken not to build duplicate file names in        */
/*  Archive                                                          */
/* * * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * * */


 SPLF1:       IF         COND((&CVTFLAGS *EQ 'SPLF1')) +
                          THEN(DO)
             CHGVAR     VAR(&IDX)    VALUE(0)
             CHGVAR     VAR(&IDXPRV) VALUE(1)
/*    Find a Jobname     Note:max length must be 10  */
 JOBNAM1:    CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '/') THEN(GOTO +
                          CMDLBL(JOBNAM1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&JOBNAM) VALUE(%SST(&INNAME &IDXPRV &LEN))
/*    Find a User ID     Note:max length must be 10  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
```

```
 USERID1:    CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(USERID1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&USERID) VALUE(%SST(&INNAME &IDXPRV &LEN))
/*    Find a Job Number     Note:max length must be 6  */

             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)

 JOBNBR1:    CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(JOBNBR1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 7)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(7))
             CHGVAR     VAR(&JOBNBR) VALUE(%SST(&INNAME &IDXPRV &LEN))

/*    Find a SPLF Name      Note:max length must be 10  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
 SPLFNAM1:   CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '/')       +
                          THEN(GOTO CMDLBL(SPLFNAM1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&SPLFNAM) VALUE(%SST(&INNAME &IDXPRV &LEN))

/*    Find a SPLF nbr      Note:max length must be 5  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
 SPLFNBR1:   CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE '.')       +
                          THEN(GOTO CMDLBL(SPLFNBR1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 5)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(5))
             CHGVAR     VAR(&SPLFNBR) VALUE(%SST(&INNAME &IDXPRV &LEN))

/*    Find a Suffix      Note:max length must be 4  */
             CHGVAR     VAR(&IDXPRV) VALUE(&IDX + 1)
 SPLSUFIX1:  CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&INNAME &IDX 1) *NE ' ')       +
                          THEN(GOTO CMDLBL(SPLSUFIX1))
             CHGVAR     VAR(&LEN) VALUE(&IDX - &IDXPRV)
             IF         COND(&LEN > 4)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(4))
             CHGVAR     VAR(&SPLSUFIX) VALUE(%SST(&INNAME &IDXPRV &LEN))

/*    Set CVTDATA       Note:max length must be 4  */
             CHGVAR     VAR(&IDX) VALUE(0)
 CVTDATA1:   CHGVAR     VAR(&IDX) VALUE(&IDX + 1)
             IF         COND(%SST(&CVTDATA &IDX 1) *NE ' ') +
                          THEN(GOTO CMDLBL(CVTDATA1))
             CHGVAR     VAR(&LEN) VALUE(&IDX)
             IF         COND(&LEN > 10)                          +
                          THEN(CHGVAR VAR(&LEN) VALUE(10))
             CHGVAR     VAR(&LIBNM) VALUE(%SST(&CVTDATA 1 &LEN))
             IF         COND(&LEN > 1)                           +
                         THEN(CHGVAR VAR(%SST(&CVTDATA &LEN 1))  +
                          VALUE('.'))

  /* Save the new name lib/file/mbr */
             CHGVAR     VAR(&OUTNAME) VALUE(&LIBNM *TCAT &SPLFNBR +
                          *TCAT '.' *TCAT &SPLSUFIX *TCAT ' ')


ENDSPLF1:     GOTO       CMDLBL(ENDCHG)
         ENDDO    /* end of SPLF1 do flag */
/*                                                              */
ENDERR:      CHGVAR     VAR(&OUTNAME) VALUE('   ')
/*                                                              */
/* * * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * */

/* End of CVTNAME for return                                    */
```

```
ENDCHG:/*     CHGVAR     VAR(&MSGVAR) VALUE(&INNAME *BCAT &MUVAR + */
       /*                  *BCAT &OUTNAME)                        */
       /*   SNDPGMMSG  MSGID(AQZ0000) MSGF(*LIBL/PKZIPMSG)        */
       /*                MSGDTA(&MSGVAR) TOPGMQ(*PRV)             */
       /*   DMPCLPGM    CL dump of variable to help debug         */

ENDEXIT:   ENDPGM
```

Appendix E - List Files and their Usage

The list file capabilities provided in the PKZIP and PKUNZIP commands can be a powerful tool for maintaining detailed selection and to exclude files. PKZIP and PKUNZIP commands also allow creating a list of files that are located in a particular archive.

Creating List Files

Both PKZIP and PKUNZIP can create a text format file of file names that meet criteria entered within FILES and EXCLUDE parameters. In PKZIP, the output files contain the names of all files in the OS/400 format, depending on if the files are from the QSYS file system or IFS. The PKUNZIP program will produce a list of names in the format of the archive. To create an output list file, place the output file name in the parameter CRTLIST(). The default value is CRTLIST(*NONE).

Depending on the value of the TYPLISTFL parameter, the output file can be put in either the QSYS file system or IFS.

TYPLISTFL(*DB): When the file system is QSYS, the output file will create a physical file (PF-DTA) with a record length of 132. For the file format in CRTLIST, you can use any of the following formats: library/file, library/file(mbr), file, or file(mbr). When a member is not entered, the member name will be the same as the file name. You should use the utility that your organization uses to edit data files.

TYPLISTFL(*IFS): When using the IFS, the output file will create a stream file (*STMF object type). Most organization uses EDTF. For the file format in CRTLIST, you can use any of the following formats: file, file. suffix, dir1/file, dir1/dir2/../dirn/file or /dir1 etc. When the path does not start with '/', then the path starts in your current directory.

When creating a file manually, follow the creation attributes described above.

Using List Files as Input

Both PKZIP and PKUNZIP programs can use list file parameters for both selections of files: (INCLFILE('file name')) and/or the excluding of file (EXCLFILE('file name'). The file name of parameters depends on the setting of TYPLISTFL (*DB or *IFS) and should follow the guidelines in Creating List Files above.

When using PKZIP, the format of files in the list file should be in the format of the iSeries files that will be processed. See the parameters FILES and EXCLUDE for specifications.

When using PKUNZIP, the format of the files in the list file should be in the format of the archive. See the parameters FILES and EXCLUDE for specifications.

PKUNZIP also has an option to create a list file in expanded mode, which will create the date and time along with the file names. This is accomplished by having a '>' character being the in the first position of the CRTLIST parameter. See the two examples below.

Create normal list file:➔ PKUNZIP ARCHIVE('atest/v560/listf') CRTLIST('atest/listfile(demo)')

```
 Edit File: ATEST/LISTFILE(DEMO)
 Record :       1   of       4 by   8           Column :    1    132 by  74
CMD ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
               ************Beginning of data**************
    TESTLIB/MYFILE/MYMBR
    TESTLIB/MYFILEGE.R/MYMBR
    TESTLIB/MYFILETE.XT/MYMBR
    TESTLIB/MYFILE27.3/MYMBR
               ************End of Data********************
```

Create an expanded list file:➔ PKUNZIP ARCHIVE('atest/v560/listf') CRTLIST('>atest/listfile(demo)')

```
 Edit File: ATEST/LISTFILE(DEMO)
 Record :       1   of       4 by   8            Column :    1    132 by  74
CMD ....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
              ************Beginning of data*************
    DT(05-20-03 16:16)  TESTLIB/MYFILE/MYMBR
    DT(02-14-03 16:44)  TESTLIB/MYFILEGE.R/MYMBR
    DT(02-14-03 16:44)  TESTLIB/MYFILETE.XT/MYMBR
    DT(02-14-03 16:44)  TESTLIB/MYFILE27.3/MYMBR
              ************End of Data********************
```

Appendix F - Translation Tables

Text files (such as program source code) are usually held within an archive using the ASCII character set for compatibility with other versions of ***PKZIP***®. For these to be usable on OS/400, they must be converted to the IBM EBCDIC character set. ***PKZIP for iSeries*** uses one of two possible internal translation tables, which should be suitable for most customers. These translation table members are used by parameters FTRAN and TRAN in both the PKZIP and PKUNZIP programs. Included (as part of the distribution) are a series of override translation tables. Some users may wish to define their own table.

The override Translation Tables included are stored as source members in file PKZTABLES ***PKZIP for iSeries*** Resources Tables. By referencing the members in parameters TRAN and FTRAN, ***PKZIP for iSeries*** will access the selected member in the PKZTABLES file and parse them to an internal hexadecimal table for use in translation.

The following translation tables are included:

Table Name	Translation from	Translation to	Explanation
ASCIIISO	EBCDIC	ASCII - iso	Translate Table
LATIN1	EBCDIC	ASCII	Latin Translate Table
NOOP	NO-OP		Translation Table Straight Hex
UKASCII	EBCDIC	UK	ASCII Translate Table
UKASCIIE	EBCDIC	UK	ASCII Translate Table-Euro
USASCII	EBCDIC	USA	ASCII Translate Table (Internal Default)
USASCIIE	EBCDIC	USA	ASCII Translate Table-Euro

Standard Code Page Support with Tables

Three data translation tables are available to assist with one or more of the latest standard EBCDIC text translation to ASCII. These tables were built to relate directly to IBM code pages numbers.

Code Page Tables available are:

Table Name	ASCII Code Page	EBCDIC Code Page	Explanation
PKZ819037	819	037	ASCII-819 <-> EBCDIC-037 Translation
PKZ819500	819	500	ASCII-819 <-> EBCDIC-500 Translation ISO8859-1
PKZ850037	850	037	ASCII-850 <-> EBCDIC-037 Translation

International Code Page Support

Some data-interchange environments require specialized multi-language character translation support. ***PKZIP for iSeries*** provides tables for character based data translation through translate tables that are also included in the PKZTABLES.

The tables for the following international code pages are provided in the ***PKZIP for iSeries*** PKZTABLES as members TRTxxyy (where xx = "from" and yy = "to").

Language	EBCDIC	ASCII	EURO/ ASCII	FROM	TO	EURO
German	273	850[1]	858	EB	AA	AI
Spanish	284	850	858	EJ	AA	AI
Portuguese	282	850	858	EI	AA	AI
Italian	280	850	858	EG	AA	AI
Danish	277	850	858	EE	AA	AI
Norwegian	277	850	858	EE	AA	AI
Swedish	278	850	858	EF	AA	AI
Finnish	278	850	858	EF	AA	AI
French	297	850	858	EM	AA	AI

These members are provided "as is." It is the responsibility of the user to ensure that data translation mapping is in accordance with their business needs.

Translation Table Layout

There are two translation tables in PKZTABLES. The first table is a translation from ASCII to EBCDIC. The second is EBCDIC to ASCII.

In each table there are 256 entries representing hex values from x'00' thru x'FF'.

Each entry is represented as a 4-character field such as 0x00 and 0xFF.

On each line there must be 8 entries with each entry separated by a space. With 8 entries per line, there must be 32 lines of table entries for each table set, representing the 256 translation values.

The tables have imbedded comments to help in their documentation.

In the table example below, to translate an ASCII character **A** (hexadecimal x'41' or decimal value of '65'), go to entry 65 in the table (Line 8, entry 2) and find a hexadecimal x'C1' which is the EBCDIC **A**.

See Example of PKZTABLES (USASCII) Translation Table.

Note: Do not alter any other members found in the PKZTABLES file or ***PKZIP for iSeries*** may not function correctly.

Creating new Translation Table Members

The following steps should be taken if you wish to define your own translation table:

1. Copy one of the distributed members in PKZTABLES to a member name of your choice.
2. Edit the new table using the OS/400 Source Entry Utility (SEU).

[1] IBM-850 = IBM-4946

3. Change the values with respect to the layout describe above, making sure not to alter the overall table layout. If the overall layout is altered, ***PKZIP for iSeries*** may not work correctly.
4. Save the member and test your changes.

Example of PKZTABLES (USASCII) Translation Table

```
/* PKZIP/400 Translate Table USASCII to EBCDIC */
/*00-07*/ 0x00 0x01 0x02 0x03 0x37 0x2D 0x2E 0x2F  /*00-07*/
/*08-0f*/ 0x16 0x05 0x25 0x0B 0x0C 0x0D 0x0E 0x9F  /*08-0f*/
/*10-17*/ 0x10 0x11 0x12 0x13 0xB6 0xB5 0x32 0x26  /*10-17*/
/*18-1f*/ 0x18 0x19 0x3F 0x27 0x1C 0x1D 0x1E 0x1F  /*18-1f*/
/*20-27*/ 0x40 0x5A 0x7F 0x7B 0x5B 0x6C 0x50 0x7D  /*20-27*/
/*28-2f*/ 0x4D 0x5D 0x5C 0x4E 0x6B 0x60 0x4B 0x61  /*28-2f*/
/*30-37*/ 0xF0 0xF1 0xF2 0xF3 0xF4 0xF5 0xF6 0xF7  /*30-37*/
/*38-3f*/ 0xF8 0xF9 0x7A 0x5E 0x4C 0x7E 0x6E 0x6F  /*38-3f*/
/*40-47*/ 0x7C 0xC1 0xC2 0xC3 0xC4 0xC5 0xC6 0xC7  /*40-47*/
/*48-4f*/ 0xC8 0xC9 0xD1 0xD2 0xD3 0xD4 0xD5 0xD6  /*48-4f*/
/*50-57*/ 0xD7 0xD8 0xD9 0xE2 0xE3 0xE4 0xE5 0xE6  /*50-57*/
/*58-5f*/ 0xE7 0xE8 0xE9 0xBA 0xE0 0xBB 0xB0 0x6D  /*58-5f*/
/*60-67*/ 0x79 0x81 0x82 0x83 0x84 0x85 0x86 0x87  /*60-67*/
/*68-6f*/ 0x88 0x89 0x91 0x92 0x93 0x94 0x95 0x96  /*68-6f*/
/*70-77*/ 0x97 0x98 0x99 0xA2 0xA3 0xA4 0xA5 0xA6  /*70-77*/
/*78-7f*/ 0xA7 0xA8 0xA9 0xC0 0x6A 0xD0 0xA1 0x07  /*78-7f*/
/*80-87*/ 0x68 0xDC 0x51 0x42 0x43 0x44 0x47 0x48  /*80-87*/
/*88-8f*/ 0x52 0x53 0x54 0x57 0x56 0x58 0x63 0x67  /*88-8f*/
/*90-97*/ 0x71 0x9C 0x9E 0xCB 0xCC 0xCD 0xDB 0xDD  /*90-97*/
/*98-9f*/ 0xDF 0xEC 0xFC 0x4A 0xB1 0xB2 0x3E 0xB4  /*98-9f*/
/*a0-a7*/ 0x45 0x55 0xCE 0xDE 0x49 0x69 0x9A 0x9B  /*a0-a7*/
/*a8-af*/ 0xAB 0x0F 0x5F 0xB8 0xB7 0xAA 0x8A 0x8B  /*a8-af*/
/*b0-b7*/ 0x3C 0x3D 0x62 0x4F 0x64 0x65 0x66 0x20  /*b0-b7*/
/*b8-bf*/ 0x21 0x22 0x70 0x23 0x72 0x73 0x74 0xBE  /*b8-bf*/
/*c0-c7*/ 0x76 0x77 0x78 0x80 0x24 0x15 0x8C 0x8D  /*c0-c7*/
/*c8-cf*/ 0x8E 0x41 0x06 0x17 0x28 0x29 0x9D 0x2A  /*c8-cf*/
/*d0-d7*/ 0x2B 0x2C 0x09 0x0A 0xAC 0xAD 0xAE 0xAF  /*d0-d7*/
/*d8-df*/ 0x1B 0x30 0x31 0xFA 0x1A 0x33 0x34 0x35  /*d8-df*/
/*e0-e7*/ 0x36 0x59 0x08 0x38 0xBC 0x39 0xA0 0xBF  /*e0-e7*/
/*e8-ef*/ 0xCA 0x3A 0xFE 0x3B 0x04 0xCF 0xDA 0x14  /*e8-ef*/
/*f0-f7*/ 0xE1 0x8F 0x46 0x75 0xFD 0xEB 0xEE 0xED  /*f0-f7*/
/*f8-ff*/ 0x90 0xEF 0xB3 0xFB 0xB9 0xEA 0xBD 0xFF  /*f8-ff*/

/* PKZIP/400 Translate Table EBCDIC to USASCII */
/*00-07*/ 0x00 0x01 0x02 0x03 0xEC 0x09 0xCA 0x7F  /*00-07*/
/*08-0f*/ 0xE2 0xD2 0xD3 0x0B 0x0C 0x0D 0x0E 0xA9  /*08-0f*/
/*10-17*/ 0x10 0x11 0x12 0x13 0xEF 0xC5 0x08 0xCB  /*10-17*/
/*18-1f*/ 0x18 0x19 0xDC 0xD8 0x1C 0x1D 0x1E 0x1F  /*18-1f*/
/*20-27*/ 0xB7 0xB8 0xB9 0xBB 0xC4 0x0A 0x17 0x1B  /*20-27*/
/*28-2f*/ 0xCC 0xCD 0xCF 0xD0 0xD1 0x05 0x06 0x07  /*28-2f*/
/*30-37*/ 0xD9 0xDA 0x16 0xDD 0xDE 0xDF 0xE0 0x04  /*30-37*/
/*38-3f*/ 0xE3 0xE5 0xE9 0xEB 0xB0 0xB1 0x9E 0x1A  /*38-3f*/
/*40-47*/ 0x20 0xC9 0x83 0x84 0x85 0xA0 0xF2 0x86  /*40-47*/
/*48-4f*/ 0x87 0xA4 0x9B 0x2E 0x3C 0x28 0x2B 0xB3  /*48-4f*/
/*50-57*/ 0x26 0x82 0x88 0x89 0x8A 0xA1 0x8C 0x8B  /*50-57*/
/*58-5f*/ 0x8D 0xE1 0x21 0x24 0x2A 0x29 0x3B 0xAA  /*58-5f*/
/*60-67*/ 0x2D 0x2F 0xB2 0x8E 0xB4 0xB5 0xB6 0x8F  /*60-67*/
/*68-6f*/ 0x80 0xA5 0x7C 0x2C 0x25 0x5F 0x3E 0x3F  /*68-6f*/
/*70-77*/ 0xBA 0x90 0xBC 0xBD 0xBE 0xF3 0xC0 0xC1  /*70-77*/
/*78-7f*/ 0xC2 0x60 0x3A 0x23 0x40 0x27 0x3D 0x22  /*78-7f*/
/*80-87*/ 0xC3 0x61 0x62 0x63 0x64 0x65 0x66 0x67  /*80-87*/
/*88-8f*/ 0x68 0x69 0xAE 0xAF 0xC6 0xC7 0xC8 0xF1  /*88-8f*/
/*90-97*/ 0xF8 0x6A 0x6B 0x6C 0x6D 0x6E 0x6F 0x70  /*90-97*/
/*98-9f*/ 0x71 0x72 0xA6 0xA7 0x91 0xCE 0x92 0x0F  /*98-9f*/
/*a0-a7*/ 0xE6 0x7E 0x73 0x74 0x75 0x76 0x77 0x78  /*a0-a7*/
/*a8-af*/ 0x79 0x7A 0xAD 0xA8 0xD4 0xD5 0xD6 0xD7  /*a8-af*/
/*b0-b7*/ 0x5E 0x9C 0x9D 0xFA 0x9F 0x15 0x14 0xAC  /*b0-b7*/
/*b8-bf*/ 0xAB 0xFC 0x5B 0x5D 0xE4 0xFE 0xBF 0xE7  /*b8-bf*/
/*c0-c7*/ 0x7B 0x41 0x42 0x43 0x44 0x45 0x46 0x47  /*c0-c7*/
/*c8-cf*/ 0x48 0x49 0xE8 0x93 0x94 0x95 0xA2 0xED  /*c8-cf*/
/*d0-d7*/ 0x7D 0x4A 0x4B 0x4C 0x4D 0x4E 0x4F 0x50  /*d0-d7*/
/*d8-df*/ 0x51 0x52 0xEE 0x96 0x81 0x97 0xA3 0x98  /*d8-df*/
/*e0-e7*/ 0x5C 0xF0 0x53 0x54 0x55 0x56 0x57 0x58  /*e0-e7*/
/*e8-ef*/ 0x59 0x5A 0xFD 0xF5 0x99 0xF7 0xF6 0xF9  /*e8-ef*/
/*f0-f7*/ 0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37  /*f0-f7*/
/*f8-ff*/ 0x38 0x39 0xDB 0xFB 0x9A 0xF4 0xEA 0xFF  /*f8-ff*/
/* PKZIP/400 Translate Tables end */
```

Appendix G - Implementation Considerations

PKZIP for iSeries components have been structured and written for the iSeries platforms and iSeries Servers. The code base includes support for the OS/400 operating system.

With any product placed into production, familiarity with (and verification of) features is always highly recommended. Below is a list of key features that users of PKZIP should thoroughly understand.

Key Features

- The commands PKZIP and PKUNZIP contain all parameters and options required to use ***PKZIP for iSeries***. The parameters and options are in a format that provide a migration path toward the open systems environment. All parameters in the two commands are supported with the OS/400 help panels. The parameters and options should be reviewed to understand the features supported.
- With the IFS, long file names are supported for Files, Archive files, and list files.
- Lists files for inputting file selections and outputting selections is supported for both the QSYS Library File system and the IFS.
- Archive files are supported in both the QSYS Library File system and the IFS. By using the archive files within the IFS, performance is enhanced. CPU operation is less efficient and archive files are smaller. As a result, elapsed run time is reduced.
- Provides a quick and easy method for installing on the iSeries without running under the QSECOFR user or security authority.
- Using the parameter MSGTYPE, messages can be directed to a printer file, the message queue, or both.
- When archive files are created in the QSYS Library File System, the record length can be defined to fit the customer environment.
- When a file is extracted to the QSYS Library File System, and neither the file nor the extended attribute for the record length exists, the default file's created record length can be defined externally.
- When compressing a file in text mode, the you can define the record as a fixed-length record with trailing blanks that are based on the file's record length description in the QSYS Library file system.
- When selecting files in the IFS for compression, you can specify whether to search recursively through the directories for file selection, or to only search the currently specified directory.
- When extracting files with the PKUNZIP command, you can drop the path of files in the archive and use only the file name. This allows you to build the extracted file in new directories, libraries, and/or files. This applies to IFS and QSYS Library file systems.
- When using files with the product, the attributes are stored in the archive to indicate how the file was stored (binary, text, text in EBCDIC, SAVF, or Database Services) and can be used when extracting files with FILETYPE(*DETECT).
- The PKUNZIP command allows for defining default paths and libraries for both IFS and QSYS Library file systems.
- File selection has been written to provide additional file filtering controls. Individual file selection can be made using the FILE parameter and/or the "List File" (INCLFILE) which allows a list of files to be selected.

- File exclusion has been written to provide additional file filtering controls. Individual file exclusion can be made using the EXCLUDE parameter and/or the “List File” (EXCLFILE), which allows a list of files to be excluded.
- ***PKZIP for iSeries*** is a licensed product, and without proper licensing, it can only be used to view archives. Licensing allows flexibility with the product features and hardware platforms. The license dataset must be defined and customized prior to use of the product.
- Extended attributes are stored within the archive directory.

  Note: These attributes are not published as part of a program control interface and may change between releases. Using the parameter TYPE(*VIEW) with VIEWOPT(*ALL) will report extended file information in the same report format order, regardless of the order that extended attributes are stored in the archive.
- ***PKZIP for iSeries*** library can be renamed from the standard distribution name of PKZ560510 to fit the operational environment.
- PKZIP and PKUNZIP commands can be executed without the products library being part of the library list.

Appendix H - SPOOL Files Considerations

This appendix contains information on how ***PKZIP for iSeries*** handles spool files in different scenarios that might be helpful consideration in planning of compressing spool files.

Spool File Selections

Care should be taken when selecting spool files to not set all of the spool file selection parameters to *ALL, as this will select all spool files on your iSeries. This is why the default for the user id is SFUSER(*CURRENT) to at least limit it to the current user in case a selection is not filled in correctly.

If a spool file is deleted after the selection but before the actual compression takes place, the PKZIP job will fail.

SPLF Attributes

When a spool is selected and the parameter EXTRAFL is coded *YES (the default) , then extended attributes listed below are stored in archive and can be viewed with PKUNZIP TYPE(*VIEW) VIEWOPT(*ALL). Also when the spool files are stored in the archive, the date and time for the file is the spool files creation date and time and can be viewed with PKUNZIP.

Extended Attributes:

1. Description: The spool file description is built as follows:

"Job-Name/User-Name/#Job-Number/Spool-File-Name/Fspool-File-Number.Suffix"
For Example: "MYJOB/BILLS#152681/INVOICE/F0021.SPLF"

2. Spool File Type: *SCS: SNA Character Stream, *IPDS: An intelligent printer data stream, *AFPDS: Advanced Function Print Data Stream, *USERASCII: An ASCII data stream user
defined, *LINE: Line data that is very printer specific, and *AFPDSLINE: Mixed data (line data and AFPDS data).
3. Target File created: Describes the target type file created during compression. SPLF: Spool Files, TXT: ASCII Text Conversion, and PDF: Portable Document Format.
4. Number of Pages contained in the spool file.

An example of the attributes view seen with –VIEWOPT(*ALL) for a spool file converted to a PDF for the might look like:

```
Filename: CRTCM60.PDF
Detected File type:     Binary
Created by:             PKZIP for iSeries  5.6    PKZIP 2.x compatible
Minimum to Extract:     PKZIP 2.0  Or Greater
Compression method:     Deflated   [Fast]
Date and Time           2003 Oct 17 07:22:00
Compressed size:        2316 bytes
Uncompressed size:      8146 bytes
32-bit CRC value (hex): 40950039
Extended attributes:    yes, [Length = 112]
Spool File Type:*SCS, Target File:PDF, Nbr Of pages(3).
SPLF Desc:EVWSS/EVWSS/#007892/CRTCM/F0060.PDF.
File Comment:"none"
```

The above view comes from the below spool file :

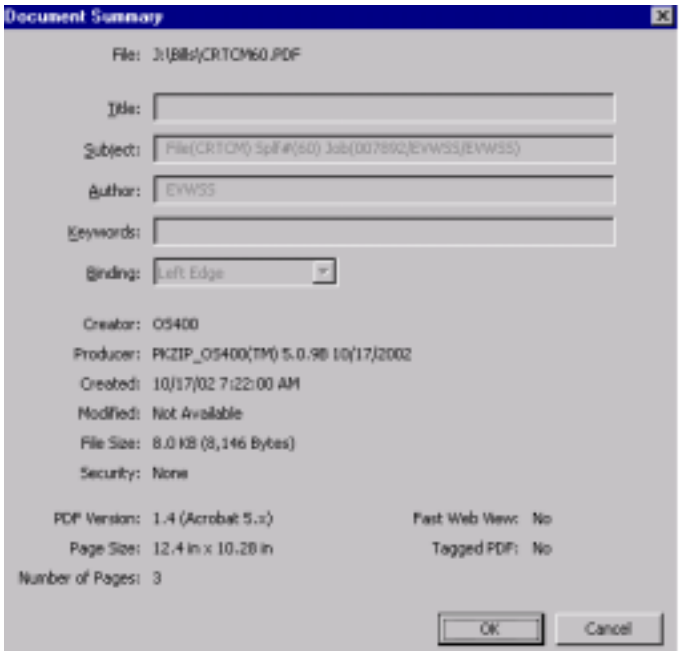
```
5722SS1  V5R1M0      Work With Output Queue    QPRINT2  in  QGPL  11/19/02 14:08:53 Page    1
File   User  User Data Status Pages  Cpy Form Tp  Pty File Number  Job    Number Date     Time
CRTCM  EVWSS           RDY      3    1   *STD     5         60    EVWSS  007892 10/17/02 07:22:00
```

PDF Creation Attributes

When creating a PDF, the attributes are also stored in the PDF document to help trace back what spool file that it originated from.

1. The date and time of the spool file creation will be the PDF date and time of creation.
2. The Author will be the USER ID that created the Spool File.
3. The subject will be made up of the Spool File Name, Number, and the JOB (Number/User ID/Job Name).

An example of a PDF summary is:



Appendix I – History of Changes

This appendix contains information on how ***PKZIP for iSeries*** handles spool files in different scenarios that might be helpful consideration in planning of compressing spool files.

RELEASE 5.5 January 2003

- New parameter ADVCRYPT to specify encryption level.
- New Parameter VPASSWORD which is required for Advanced Encryption and is used to verify with PASSWORD to assure accuracy.
- Password now accepts up to 200 characters.
- Spool Files can be archived or converted to a TEXT or PDF format.
- PKUNZIP now supports using a member *FIRST or *LAST for a member when using the QSYS file system.
- PKZIP now recognizes the older physical files with attributes of PF38 (System/38) physical files.
- Parameter CVTDATA is added to PKZIP and PKUNZIP commands and CVTNAME program changed to accept 4 variables (Old File Name, New File name, CVTFLAG, and CVTDATA).
- PKZIP and PKUNZIP will issues *STATUS message for completion messages so they can be monitored. The following message can now be monitored:

  AQZ0012 - PKZIP ending with Nothing to do for <&1>.

  AQZ0020 - PKZIP Completed Successfully.

  AQZ0022 - PKZIP Completed with Errors.

  AQZ0037 - PKUNZIP Completed Successfully.

  AQZ0038 - PKUNZIP Completed with Errors.
- Fix security authorization when creating a new directory or folder in the IFS so that authority is inherited from directory above.
- When only the file name is used in FILES parameter for selection, PKZIP was defaulting to *CURLIB. This has been revised per documentation to use *LIBL as the default.
- When creating a file in the Integrated File System, the file will be created with attributes of directory that is created in plus owner of the PKUNZIP run.
- PKZIP revised to include files with attribute DDMF and file type DDM while searching for files in the selection parameters.
- Added translation code tables 850 ASCII and 037 EBCDIC to PKZIPTABLES file (PKZ850037).
- PKZCHGOWN CLP program "source" is provided in the source file QCLSRC file to assist customer who may want to change the owner of the object distributed.
- When creating files, PKUNZIP uses SIZE(*NOMAX) with CRTPF to avoid constant message replies.
- Added the informational program WHATOSV to distribution. WHATOSV shows OS Version, serial number, and process group required for licensing. To use CALL WHATOSV.
- EXDIR was expanded from 30 character to 224 characters to allow for more flexibility with the IFS.

- When getting the message AQZ0148 "Archive file empty", PKZIP will also issue the message AQZ0022 "PKZIP Completed with Errors" at the end of the run. The AQZ0022 is a message that can be monitored. The message AQZ0022 is now being issued as a diagnostic type message instead of a completion message.
- PKUNZIP revised to not use IFS Stat when checking the archives in the QSYS file system. This was preventing authority adoption for PKUNZIP archive files. PKUNZIP is now performing object and file checking.
- PKZIP revised random number for creation of the Temporary archive name was improved to avoid duplication. The random routines now use the job number for initial seed.
- When extracting with PKUNZIP TYPE(*NEWER) and the files extracting does not exist, Extract the files as if they the same as newer when files exist.
- Improved performance extracting to a file that has a large number of members. Improved Performance selecting files for compression when files has large number members.
- Expanded the field sizes of the ARCHIVE and FILES parameters to 140 characters in PKZIP and PKUNZIP commands to allow more folders in the path of IFS files.

  Added keyword ?MBR to parameter EXDIR in command PKUNZIP. If EXDIR is coded with keyword ?MBR and the file system is the QSYS Library system PKUNZIP will use the member name for the file name. For example:
- EXDIR('newlib/?MBR') and DROPPATH(*ALL) parameters are coded and the file name in archive is "mylib/myfile/mymbr", the file will be extract to the file "newlib/mymbr(mymbr)". This only valid for TYPFL2ZP(*DB) files.
- Support Of Spool Files along with text document conversion.
- New parameter DELIM was added to PKZIP to provide the ability to define the end-of-record characters at the end of a Text record (such as, CRLF, etc.).
- New Install process from CD using LODRUN command.

RELEASE 5.5.1 March 2003

- Resolved under rare conditions when extracting a text file from an archive created in a UNIX environment, there may be 1- 2 invalid extra last text records..
- Added new error check for Spool Files: AQZ0301 Parameter SFTARGET(*SPLF) requires EXTRAFLD(*YES). Correct and rerun. AQZ0302 FILETYPE parameter must be (*DETECT) when compressing Spool Files.
- Improved page fitting during spool file conversion when using *PDFLETTER and PDFLEGAL as the option in the parameter SFTARGET.

RELEASE 5.5.2 May 2003

- Under certain conditions when converting a spool file to a text or PDF file, there are extraneous characters in some lines due to buffers not being cleared proper.
- Added message AQZ0304 to indicate the job in parameter SFJOBNAM cannot be found or is invalid when selecting a spool file.
- Added ability to have a GZIP archive file greater than 2 Gig.

Appendix J – Creating Commands with new defaults

There are times when some clients would like to change the defaults of the commands for their environment. This is why the source for the commands PKZIP, PKZSPOOL, and PKUNZIP are included in the distribution library in the QCMDSRC file. It is important that the sequences, values, and properties of the parameters do not change. To protect you ability to recover, it is suggested that you rename the previous command and command source and copy them before make any changes.

If you want to restrict where the commands can be performed then change the parameter ALLOW(*ALL) to the value you require. See IBM CL manuals or prompt CRTCM.

Below are sample commands to create the commands where in this example the PKZIP library is PKZ560510.

Create PKZIP Command:

```
➔CRTCMD  CMD(PKZ560510/PKZIP) PGM(PKZ560510/PKZIP)
      SRCFILE(PKZ560510/QCMDSRC)
      HLPPNLGRP(PKZ560510/PKZIP) HLPID(*CMD)
```

Create PKZSPOOL Command

```
➔CRTCMD  CMD(PKZ560510/PKZSPOOL) PGM(PKZ560510/PKZIP)
      SRCFILE(PKZ560510/QCMDSRC)
      HLPPNLGRP(PKZ560510/PKZIP) HLPID(*CMD)
```

Create PKUNZIP Command:

```
➔CRTCMD  CMD(PKZ560510/PKUNZIP) PGM(PKZ560510/PKUNZIP)
               SRCFILE(PKZ560510/QCMDSRC)
               HLPPNLGRP(PKZ560510/PKUNZIP) HLPID(*CMD)
```

Appendix K – Large File Support Licensing

Large File Support is a licensed feature in release 5.6 of PKZIP for iSeries™ and as such will follow all other licensed features and consideration. In most cases care is taken at the beginning of each run to warn the customer of the ZIP64 requirements. One major difference is the ZIP64 feature could be activated during a large ZIP run where the archive crosses the boundary for ZIP64 support. In this situation the ZIP process will continue until the build has completed. A message will be issued and a temporary 5 day grace period will go into effect. Below is a summary list on what can activate the ZIP64 feature license.

- Any indication of ZIP64 in an existing Archive will require "Large Archive Support" to update the archive with PKZIP or extract with PKUNZIP.
- A 5-day grace period will be allowed on the first trigger point of ZIP64 usage. If the grace period expires then all ZIP64 processing will halt and require the obtaining of the license feature for the Large File System.
- Large Archive Support includes:
    - Encountering a ZIP64 End-Central configuration in an existing Archive.
    - Central Directory Offset is beyond the 4 GB range.
    - The number files in Archive exceeds 65,534.
    - Encountering a Central Directory entry in ZIP64 format: (Uncompressed size, Compressed Size, or Local Offset exceeds 4 GB).
    - When selecting files in PKZIP, the number of files in the old archive plus the new selected files exceeds 65,534 number of files.
    - When selecting one or mores files to compress and the file sizes exceed 4 GB.
    - Trying to extract a file from an archive where the file size exceeds 4 GB.

Note: PKUNZIP will *VIEW and *TEST archives with ZIP64 condition without trying to check for Large System Support feature.

Appendix L - Extended Attributes

This chapter reviews the extended attributes that ***PKZIP for iSeries*** builds when using the parameters EXTRAFLD(*YES) or DBSERVICE(*YES). These extended attributes can be viewed for a file by using PKUNZIP with parameters TYPE(*VIEW) and VIEWOPT(*DETAIL) for the archive.

Within the ZIP Archive are two directories that provide information about the files that are held within it. A Local Directory (included at the beginning of each file) contains information such as the file size and the date the file was zipped. The Central Directory (located at the end of the ZIP Archive) lists the complete contents of the ZIP Archive and is the primary source of information for UNZIP processing. In each directory there can exist extended data or attributes for the compressed files.

When EXTRAFLD(*YES) is specified, a set of basic attributes are created for each file in the archive. The type of attribute created depends on the type of file system (QSYS Library File System or IFS) in which the file being compressed exists. The standard attributes for both systems are described below.

If the parameter DBSERVICE(*YES) is specified for the PKZIP command, and if the file being compressed is a Physical File (PF-DTA or PF_SRC), then the basic extended attributes are created followed by the detailed Database attributes. With the detailed Database attributes, PKUNZIP can build and compile the DDS source file to recreate the database file (see DATABASE Attributes). The database file will be compressed in the Binary mode.

Standard QSYS Library File System Attributes

When the files being compressed are from the “QSYS Library File System”, the standard attributes created are described in the following tables.

For additional information, see the DSPFD command in the IBM manuals.

Physical Files (both Source and Data)

	Attribute	Description
1	Maximum Record Length	Specifies the length (in bytes) of the records stored in the physical file.
2	Library Text	The text description of the library that the file does exist.
3	File Text	The text description of the File.
4	Member Text	The text description of the member.
5	File Type	Specifies whether each member of the physical file being created contains data records or contains source records (statements. PF_SRC or PF_DTA.).
6	Source Type Code	Specifies the source type of the member, such as, CLP, PF, LF, RPGLE, etc.

SAVF

	Attribute	Description
1	Library Text	The text description of the library that the SAVF exists.
2	File Text	The text description of the SAVF.

Standard "IFS" Attributes

When the files being compressed are from the "Integrated File System," the standard attributes for stream files are as described in the following table.

	Attribute	Description
1	Code Page	
2	File creations date/time	The date and time when the file was created.
3	File last access date/time	The date and time when the file was last accessed.
4	File last modification date/time	The data and time when the file was last modified.

Advanced Encryption Attributes

When the files being compressed with Advanced Encryption, encryption attributes are added to the archive.

	Attribute	Description
1	Encryption Method/ Algorithm	Indicates the Encryption Method.
2	Encryption Key Type	The key type used with the encryption algorithm.
3	Security Method	Indicates whether a password, certificate, or both password and certificate are in use.

DATABASE Attributes

With the parameter DBSERVICE(*YES) in PKZIP, a special set of attributes are created for files that are Physical Files (both Source and Data types) in the "QSYS Library File System". These attributes can be used by PKUNZIP to create a database by building the DDS source and issuing the CRTPF command for the source.

These Database attributes are described in the following tables.

File Physical Attributes

For more information, see the CRTPF and CHGPF commands or IBM Publications.

	Attribute	Description
1	File Record Format	The name of the file's Record Format.

	Attribute	Description
2	File Record Text	The Text description for the File's Format.
3	Maximum Number of Members	Specifies the maximum number of members that the physical file being created can have at any time.
4	Number Fields	The number of fields in the database.
5	Number of keys	The number of key fields in the database.
6	SIZE - nbr Records	SIZE parameter option - The number of records that can be inserted before an automatic extension occurs.
7	SIZE - increment value	SIZE parameter option - Value for the number of additional records.
8	SIZE - number of increments	SIZE parameter option - Maximum number of parts to be added automatically to the member.
9	Public Authority	AUT - Specifies the authority given to users who do not have specific authority to the physical file. **Note:** If an authorization list was specified for the file then AUT is set to *CHANGED. See "Database Attributes Considerations."
10	Record Description CCSID	The Coded character set identifier for the Record description.
11	Delete Percent	DLTPCT - Specifies the percentage of deleted records a file can contain before you want the system to send a message to the system history log.
12	Reuse deleted records	REUSEDLT - Specifies whether deleted record space should be reused on subsequent write operations.
13	Access path size	ACCPTHSIZ - Specifies the maximum size of auxiliary storage that can be occupied by access paths that are associated with keyed source physical files.
14	Access path maintenance	MAINT - Specifies the type of access path maintenance used for all members of the physical file.
15	Access path recovery	RECOVER - For files having immediate or delayed maintenance on their access paths, specifies when recovery processing of the file is done if a system failure occurs while the access path is being changed.
16	Allocate storage	ALLOCATE - Specifies storage space is allocated for the initial number of records (SIZE parameter) for each physical file member when it is added.
17	Force keyed access path	FRCACCPTH - For files with keyed access paths only, specifies access path changes are forced to auxiliary storage along with the associated records in the file whenever the access path is changed.
18	Record format level check	LVLCHK - Specifies the record format level identifiers in the program are checked against those in the logical file when the file is

	Attribute	Description
		opened.
19	Program describe File	Defines whether File is external file type.
20	Triggers Available	File had Triggers defined. See “Database Attributes Considerations.”
21	File SQL Table	File is an SQL Table.
22	Constraints Available	File had Constraints defined. See “Database Attributes Considerations.”
23	File has Null Fields	File contains fields which allows NULL contents. See “Database Attributes Considerations.”

File Field Attributes

See “iSeries e DDS Reference Publication No. RBAF-P000-00” for a description of the keywords.

	Attribute	Description
1	Field Name	Field Name.
2	Field Data Type	Specifies the data type associated with the field (such as, A-character, P-Packed, Decimal, etc.).
3	Field Length	Specifies the field length for named field.
4	Number of Decimals	Specifies the decimal placement within a zoned decimal field and the data type of the field as it appears in your program.
5	Field Usage	Specifies that a named field is an output-only or program-to-system field.
6	Field CCSID	(CCSID) - Specifies a coded character set identifier for character fields.
7	Field Text	(TEXT) - A text description (or comment) for the record format or field that is used for program documentation.
8	Column Headings	(COLHDG) - Specifies column headings used as a label for this field by various iSeries file tools.
9	Keyboard Shift Character	(REFSHIFT) - Specifies a keyboard shift for a field when the field is referred to in a display file or DFU operation.
10	Allow NULLS	(ALWNULL) - To define this field to allow the null value.
11	Variable Length	(VARLEN) - To define this field as a variable length field.
12	Alias	(ALIAS) - Specifies an alternative name for the field.
13	Default Values	(DFT) - Specifies a default value for a field.
14	Edit Formatting values	(EDTCDE, EDTWRD, DATFMT, DATSEP, TIMFMT, TIMSEP) - Specifies editing for the field you are defining when the field is referenced later during display or printer file creation.
15	Validity Checking	(CHECK, COMP, RANGE, etc.) - Specifies validity checking in display files.
16	Indicator for Double Precision	An indicator for float type fields specifying single or double precision.
17	Allocated Length	The allocated length in bytes for data types that use variable lengths.

Key Field Attributes

See the IBM Manual for the DDS description of keywords for KEYS.

	Attribute	Description
1	Key Field Name	The field name of the key.
2	Type of Sequence	Defines the type of key and the sequence of the key.

Database Attribute Considerations

Special Database functionality and information such as Triggers, File Constraints, Authorization Lists, and Alternate Collating Sequences are not stored in an archive.

1. If a file has fields that have the option ALWNULL, warning message AQZ0521 will be issued.
2. If the file contains Triggers, warning message AQZ0518 will be issued. Any information about the triggers and associated programs are not stored in the archive.
3. If a file has File Constraints, warning message AQZ0519 will be issued.
4. If an authorization list was specified for the file, then the AUT parameter is set to "*CHANGED." The authorization list is not stored in the archive. It is up to the user to set the authorization list when extracted.
5. If a file has an Alternate Collating Sequence, warning message AQZ0520 will be issued. The information about the alternate collating sequence is not stored the archive.

If the database's Triggers, File Constraints, Alternate Collating Sequence, and Logical files are to be preserved, then save the database file, trigger programs (if they exist), and logical files to a SAVF. Then compress the SAVF.

Note: Using DBSERVICE(*YES) can increase the size of the archive because some databases may have hundreds of fields and attributes. If this archive will be used on a platform other than iSeries with ***PKZIP for iSeries***, these attributes are ignored and therefore should not be used.

Source File Considerations

When compressing source files, the use of the archive file should be considered. By using the DBSERVICE(*NO), only the data is extracted in text mode, such as, the sequences numbers and change dates are not included. This would be the desired method if the source members are being transferred to another platform, or if the sequence number and changes dates are not important. This method would also give the best results for the total size of the archive.

If the sequence numbers and change dates are to be preserved, then you would use DBSERVICE(*YES) to store this source file as a database file. This would not work very well on non-EBCDIC platforms because the data is stored in binary mode with no EBCDIC to ASCII translation. The problem is that each member will have all available database attributes. This is still minimal because there are only three fields for a source database.

Spool File Attributes

When the files being compressed are from a spool file, the standard attributes are :

	Attribute	Description
1	Spool File Type	Device Type SCS, AFP, etc.
2	Target Output Type	The type of file that was archived or converted (Spool File, Text, or PDF).
3	Internal Job and spool ID	Internal Job and Spool codes controlling the spool file.
4	Spool File description	The description of the file using the file name, file number, job ,user and job number.
5	Pages	Number of pages in the spool file.
6	Code Page	The code page the was used to convert the spool file to TEXT or PDF..
7	OUTQ	The OUTQ and the OUTQ library the spool file resided.
8	User ID	The User who created the Spool File

Appendix M – FIPS-197 Certification of PKZIP for AES

Advanced Encryption Standard FIPS Validation

PKZIP for OS/400 Version 5.5 and higher (which includes ***PKZIP for iSeries***) use AES, which is the official US Government standard for encryption. The AES algorithm was approved as the Federal Information Processing Standard by the Commerce Department on May 26th, 2002.

The National Institute of Standards and Technology (NIST) has announced approval of the Federal Information Processing Standard (FIPS) for the AES algorithm (see NIST AES FIPS Information at http://csrc.nist.gov/CryptoToolkit/aes/).

The ***PKZIP for OS/400 Version 5.5*** implementation was validated in accordance with FIPS-197 for the Advanced Encryption Standard. Information regarding the validation specification and certifications of PKWARE products can be found at http://csrc.ncsl.nist.gov/cryptval/aes/aesval.html ***(certificate #63)***.

The AES Algorithm

The Rijndael algorithm was chosen uses a combination of advanced security, performance, efficiency, ease of implementation, and flexibility to make it the appropriate standard of advanced encryption for the AES.

Rijndael performs consistently in both hardware and software and in cross platform environments regardless of its use in feedback or non-feedback modes. Rijndael's key setup time is very good, and its key agility is excellent. Memory requirements are very low, making it the first choice for restricted-space environments, in which it also demonstrates high performance. Power and timing attacks are easily defended against due to Rijndael's operations.

Note that the AES was intentionally developed to replace DES.

AES Key Sizes

Currently, AES has three key sizes. They are: 128, 192, and 256 bits. Key sizes are depicted in the following decimal terms:

3.4×10^{38} possible 128-bit keys;
6.2×10^{57} possible 192-bit keys; and
1.1×10^{77} possible 256-bit keys.

In comparison, DES keys are only 56 bits, which means there are approximately 7.2×10^{16} possible DES keys. Therefore, there they are on the order of 10^{21} times more AES 128-bit keys than DES 56-bit keys.

Advanced Encryption Standard Algorithm Validation Certificate

The National Institute of Standards and Technology of the United States of America

Certificate No. 63

The Communications Security Establishment of the Government of Canada

The National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) hereby validate the Advanced Encryption Standard (AES) algorithm testing results of the implementation identified as:

PKZIP for OS/400, Version 5.5

and supplied by:

PKWARE, Inc.

in accordance with the specifications of the *Advanced Encryption Standard (AES)* (FIPS 197) and *Recommendations for Block Cipher Modes of Operation* (SP800-38A 2001 ED) as indicated on the reverse of this certificate. Implementations bearing the same identification and manufactured to the same specifications as the validated implementation may be labeled as complying with FIPS 197 for the modes, states, and key sizes identified in this certificate. No reliability test has been performed and no warranty of the implementation is either expressed or implied.

The validated implementation was tested using the following operating environment (for software implementations, operating environment includes processor and operating system; for firmware implementations, operating environment includes processor only; for hardware implementations, operating environment is not applicable):

OS/400 5.2; Proc: 2434

The vendor should be contacted to obtain a list of operating environments that support the validated implementation. Likewise, the vendor should be contacted to obtain a list of products/applications that use the validated implementation.

This certificate must include the following page that details the scope of conformance and includes the validation authorities' signatures.

The NIST document, *"The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)"* describes a series of known answer, multi-block message and Monte Carlo tests for implementations of FIPS 197:*Advanced Encryption Standard* , using modes of operation specified in NIST Special Publication 800-38A 2001 ED,*Recommendation for Block Cipher Modes of Operation* . This implementation has been tested using the Cryptographic Algorithm Validation System (CAVS) Version 1.3. The scope of conformance achieved by the algorithm implementation identified as:

PKZIP for OS/400, Version 5.5

and tested by the accredited Cryptographic Module Testing laboratory: **CEAL: A CygnaCom Solutions Laboratory**
NVLAP Lab Code 200002-0

is as follows. The validated implementation performs AES in the following modes of operation, states, and key sizes:

Mode(s) of Operation	State(s)	Key Size(s)
Cipher Block Chaining (CBC)	**Encrypt/Decrypt**	**128,192,256**

Signed on behalf of the Government of the United States

Signature: [signature]

Date: 10 March 2003

Chief, Computer Security Division
National Institute of Standards and Technology

Rev. 07/2002

Signed on behalf of the Government of Canada

Signature: [signature]

Date: 4 Apr 02

Director, Information Protection Group
Communications Security Establishment

GLossary

This glossary provides definitions for items that may have been referenced in the PKZIP® documentation. It is not meant to be exhaustive. There are excellent source of documentation for computing terms on the Internet, three of which are shown below:

IBM's Terminology Web Site	**http://www.networking.ibm.com/nsg/nsgmain.htm**

Absolute Path Name

A string of characters that is used to refer to an object, starting at the highest level (or root) of the directory hierarchy. The absolute path name must begin with a slash (/), which indicates that the path begins at the root. This is in contrast to a Relative Path Name. See also Path Name.

AES

The Advanced Encryption Standard is the official US Government encryption stand for customer data.

American Standard Code for Information Interchange

The ASCII code (American Standard Code for Information Interchange) was developed by the American National Standards Institute for information exchange among data processing systems, data communications systems, and associated equipment and is the standard character set used on MS-DOS and UNIX-based operating systems. In a ZIP archive, ASCII is used as the normal character set for compressed text files. The ASCII character set consists of 7-bit control characters and symbolic characters, plus a single parity bit. Since ASCII is used by most microcomputers and printers, text-only files can be transferred easily between different kinds of computers and operating systems. While ASCII code does include characters to indicate backspace, carriage return, etc., it does not include accents and special letters that are not used in English. To accommodate those special characters, Extended ASCII has additional characters (128-255). Only the first 128 characters in the ASCII character set are standard on all systems. Others may be different for a given language set. It may be necessary to
create a different translation tables (see Translation Table) to create standard translation between ASCII and other character sets.

American National Standards Institute (ANSI)

An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

ANSI

See American National Standards Institute.

API

See Application Programming Interface shown below.

Application Programming Interface

An interface between the operating system (or systems-related program) that allows an application program written in a high-level language to use specific data or services of the operating system or the program. The API also allows the user to develop an application program written in a high level language to access PKZIP data and/or functions of the PKZIP system.

Application System/400 (iSeries)

One of a family of general purpose systems with a single operating system (Operating System/400) that provides application portability across all models.

Archive

1. The act of transferring files from the computer into a long-term storage medium. Archived files are often compressed to save space.
2. An individual file or group of files which must be extracted and decompressed in order to be used.
3. A file stored on a computer network, which can be retrieved by a file transfer program (FTP) or other means.
4. The PKZIP file that holds the compressed/zipped data file.

ASCII

See American Standard Code for Information Interchange.

iSeries

See Application System/400 shown above.

iSeries Object

An object that exists in a library on the iSeries system and is represented by an object on the PC. For example, a user profile is an iSeries object represented on the PC by the user profile object.

Authorized Program Analysis Report (APAR)

A request for correction of a defect in a current release of an IBM supplied program.

Bandwidth

The capacity of a communications line, normally expressed in bits per second (bps). A lack of bandwidth is one of the prime motivations for compression software.

Batch Job

A predefined group of processing actions submitted to the system to be performed with little or no interaction between the user and the system. This is in contrast to an Interactive Job.

Binary File

A file that contains codes that are not part of the ASCII character set. Binary files can utilize all 256 possible values for each byte in the file.

Bit

A contraction of binary digit. Either of the binary digits, 0 or 1. Compare with byte.

Block

(1) A group of records that are recorded or processed as a unit.
(2) A set of adjacent records stored as a unit on a disk, diskette, or magnetic tape.

Byte

(1) The smallest unit of storage that can be addressed directly.
(2) A group of 8 adjacent bits. In the EBCDIC/ASCII coding system, 1 byte can represent a character. In the double-byte coding system, 2 bytes represent a character.

Code Page

A specification of code points for each graphic character set or for a collection of graphic character sets. Within a given code page, a code point can have only one specific meaning. A code page is also sometimes known as a code set.

Command Line

The blank line on a display console where commands, option numbers, or selections can be entered.

Control Language (CL) Program

A program that is created from source statements consisting entirely of control language commands.

CRC

See Cyclic Redundancy Check.

Cryptography

(1) A method of protecting data. Cryptographic services include data encryption and message authentication.
(2) In cryptographic software, the transformation of data to conceal its meaning; secret code.
(3) The transformation of data to conceal its information content, to prevent its undetected modification, or to prevent its unauthorized use.

Current Library

The library that is specified to be the first user library searched for objects requested by a user. The name for the current library can be specified on the Sign-On display or in a user profile. When you specify an object name (such as the name of a file or program) on a command, but do not specify a library name, the system searches the libraries in the system part of the library list, then searches the current library before searching the user part of the library list. The current library is also the library that the system uses when you create a new object, if you do not specify a library name.

Cyclic Redundancy Check (CRC)

A Cyclic Redundancy Check is a number derived from a block of data, and stored or transmitted with the data in order to detect any errors in transmission. This can also be used to check the contents of a ZIP archive. It's similar in nature to a checksum. A CRC may be calculated by adding words or bytes of the data. Once the data arrives at the receiving computer, a calculation and comparison is made to the value originally transmitted. If the calculated values are different, a transmission error is indicated. The CRC information is called redundant because it adds no significant information to the transmission or archive itself. It's only used to check that the contents of a ZIP archive are correct. When a file is compressed, the CRC is calculated and a value is calculated based upon the contents and using a standard algorithm. The resulting value (32 bits in length) is the CRC that is stored with that compressed file. When the file is decompressed, the CRC is recalculated (again, based upon the extracted contents), and compared to the original CRC. Error results will be generated showing any file corruption that may have occurred.

Data Compression

The reduction in size (or space taken) of data volume on the media when performing a save or store operations.

Data Integrity

(1) The condition that exists as long as accidental or intentional destruction, alteration, or loss of data does not occur.
(2) Within the scope of a unit of work, either all changes to the database management systems are completed or none of them are. The set of change operations are considered an integral set.

DBCS

See Double-byte Character Set.

Double-byte Character Set (DBCS)

A set of characters in which each character is represented by 2 bytes. Languages such as Japanese, Chinese, and Korean, which contain more symbols than can be represented by 256 code points, require double-byte character sets. Because each character requires 2 bytes, the typing, displaying, and printing of DBCS characters requires hardware and programs that support DBCS. Four double-byte character sets are supported by the system: Japanese, Korean, Simplified Chinese, and Traditional Chinese. See also the Single-Byte Character Set (SBCS).

EBCDIC

See the Extended Binary Coded Decimal Interchange Code shown below.

Encryption

The transformation of data into an unintelligible form so that the original data either cannot be obtained or can be obtained only by decryption.

Extended Attribute

Information attached to an object that provides a detailed description about the object to an application system or user.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

The Extended Binary Coded Decimal Interchange Code is an 8-bit binary code for larger IBM mainframes in which each byte represents one alphanumeric character or two decimal digits. The single-byte structure has a range of X'00' to X'FF'. Control commands are subset with a range of X'00' to X'3F' while graphic characters have a range from X'41' to X'FE'. The space character is represented by a X'40'. EBCDIC is similar in nature to ASCII code, which is used on many other computers. When ZIP programs compress a text file, they translate data from EBCDIC to ASCII characters within a ZIP archive using a translation table.

File Transfer Protocol (FTP)

In TCP/IP, an application protocol used for transferring files to and from host computers. FTP requires a user ID and possibly a password to allow access to files on a remote host system. FTP assumes that the Transmission Control Protocol is the underlying protocol.

FTP

See File Transfer Protocol shown above.

GZIP

GZIP (also known as GNU zip) is a compression utility designed to utilize a different standard for handling compressed file data in an Archive. Its main advantages over other compression utilities are much better compression and freedom from patented algorithms. It has been adopted by the GNU project and is now relatively popular on the Internet. Additional information can be found at http://www.gzip.org.

Host

The controlling or highest-level system in a data communications configuration; for example, an iSeries system is the host system for the work stations connected to it.

Integrated File System

A function of the operating system that provides storage support similar to personal computer operating systems (such as DOS and OS/2) and UNIX systems.

Interactive Job

A job started for a person who signs on to a work station and communicates (or "converses") with another computing entity such as a mainframe or iSeries system. This is in contrast to a Batch Job.

Keyed Sequence

An order in which records are retrieved based on the contents of key fields in records. For example, a bank name and address file might be in order and keyed by the account number.

Keyword

(1) A mnemonic (abbreviation) that identifies a parameter in a command.
(2) A user-defined word used as one of the search values to identify a document during a search operation.
(3) In COBOL, a reserved word that is required by the syntax of a COBOL statement or entry.

(4) In DDS, a name that identifies a function.
(5) In REXX, a symbol reserved for use by the language processor in a certain context. Keywords include the names of the instructions and ELSE, END, OTHERWISE, THEN, and WHEN.
(6) In query management, one of the predefined words associated with a query command.
(7) A name that identifies a parameter used in an SQL statement. See also parameter.

LAC

See License Authorization Code shown below.

Lempel-Ziv (LZ)

A technique for compressing data. This technique replaces some character strings, which occur repeatedly within the data, with codes. The encoded character strings are then kept in a common dictionary, which is created as the data is being sent.

Library List

A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL.

License Authorization Code (LAC)

The inserted code that is needed to unlock an iSeries licensed program.

Logical Partition

A subset of a single iSeries system that contains resources (such as processors, memory, and input/output devices). A logical partition operates as an independent system. If hardware requirements are sufficient, multiple logical partitions can exist within a system.

LZ

See Lempel-Ziv (LZ) shown above.

New ZIP Archive

A New ZIP archive is the archive created by a compression program when either an old ZIP archive is updated or when files are compressed and no ZIP archive currently exists. It may be thought of as the “receiving” archive. Also see Old ZIP Archive.

Null Value

A parameter position within a record for which no value is specified.

n-way Processor Architecture

A processor architecture that provides expandability for future system growth by allowing for additional processors. To the user, the additional processors are transparent because they separately manage the work load by sharing the work evenly among the n-way processors.

Old ZIP Archive

An Old ZIP archive is an existing archive which is opened by a compression program to be updated or for its contents to be extracted. It may be thought of as the “sending” archive. Also see New ZIP Archive.

Operating System/400 (OS/400)

An IBM-licensed program that is as the primary operating system for an iSeries system.

OS/400

See Operating System/400 shown above.

Output Queue

An AS/400 object that contains entries for spooled output files to be written to an output device.

Packed Decimal Format

A decimal value in which each byte within a field represents two numeric digits except the far right byte, which contains one digit in bits 0 through 3 and the sign in bits 4 through 7. For all other bytes, bits 0 through 3 represent one digit; bits 4 through 7 represent one digit. For example, the decimal value +123 is represented as 0001 0010 0011 1111 (or 123F in hexadecimal).

Path Name

(1) A string of characters used to refer to an object. The string can consist of one or more elements, each separated by a slash (/), and may begin with a slash. Each element is typically a directory or equivalent, except for the last element, which can be a directory or another object (such as a file).
(2) A sequence of directory names followed by a file name, each separated by a slash.
(3) In a hierarchical file system (HFS), the name used to refer to a file or directory. The path name must start with a slash (/) and consist of elements separated by a slash. The first element must be the name of a registered file system. All remaining elements must be the name of a directory, except the last element, which can be the name of a directory or file. See also Absolute Path Name and Relative Path Name.
(4) The name of an object in the integrated file system. Protected objects have one or more path names.

Physical File

Describes how data is to be presented to (or received from) a program and how data is stored in the database. A physical file contains a single record format and at least one member.

Physical File Member

A named subset of the data records in a physical file. See also member.

Production Library

A library which contains objects needed for normal processing. This contrasts with a Test Library.

Program Temporary Fix (PTF)

A temporary solution to (or a bypass of) a problem that is necessary to provide a complete solution to correct a defect in a current unaltered release of a program. May also be used to provide an enhancement to a product before a new release of the product is available. Generally, PTFs are incorporated in a future release of the product.

PTF

See Program Temporary Fix shown above.

QSYS

The library shipped with the system that contains objects, such as authorization lists and device descriptions created by a user, and the system commands and other system objects required to run the system. The system identifier is QSYS.

Qualified Name

The full name of the library that contains the object and the name of the object.

Record

A group of related data, words, or fields treated as a single unit, such as a name, address, and social security number.

Reduced Instruction Set Computer (RISC)

A RISC computer uses a small, highly efficient subset of the instructions available on a standard computer. This allows for rapid processing.

Relative Path Name

A string of characters that is used to refer to an object, starting at some point in the directory hierarchy other than the root. A relative path name does not begin with a slash (/). The starting point is frequently a user's current directory. This is in contrast to an Absolute Path Name. See also Path Name.

Return Code

A value generated by operating system software to a program to indicate the results of an operation by that program. The value may also be generated by the program and passed back to the operator.

RISC

See Reduced Instruction Set Computer shown on page 206.

SBCS

See Single-Byte Character Set.

Service Pack

The iSeries product has available a collection of code fixes that contain PC code. These fixes are contained in a single program temporary fix (PTF) which makes installation easier.

Single-Byte Character Set (SBCS)

A coded character set in which each character is represented by a one-byte code point. A one-byte code point allows representation of up to 256 characters. Languages that are based on an alphabet, such as the Latin alphabet (as contrasted with languages that are based on ideographic characters) are usually represented by a single-byte coded character set. For example, the Spanish language can be represented by a single-byte coded character set. See also the Double-Byte Character Set (DBCS).

Source File

A file of programming code that has not yet been compiled into machine language. A source file can be created by the specification of FILETYPE(*SRC) on the Create command. A source file can contain source statements for such items as high-level language programs and data description specifications. Source files maintained on a PC typically use a .TXT as the extension. On a mainframe, source files are typically found in a partitioned data set or are maintained within a library management tool.

Spool File

Files that exist in an "Output Queue" which contain reports to printed on the AS/400 system. Theses files along with attributes can then be directed and transformed to a printer attached to your system.

Stream File

A data file that contains continuous streams of bits such as PC files, documents, and other data stored in iSeries folders. Stream files are well suited for storing strings of data such as the text of a document, images, audio, and video. The content and format of stream files are managed by the application rather than by the system.

System Library

The library shipped with the operating system that contains objects, such as authorization lists and device descriptions created by a user. Also included are system commands and other system objects required to run the system. The system identifier is QSYS.

System Processor

The operating system logic that contains the processor function to translate and process OS/400 control language commands and programming language statements.

Time Stamp

A software mechanism for recording the current date and/or current time of day.

Translation Table

Translation tables are used by the PKZIP and PKUNZIP programs for translating characters in compressed text files between the ASCII character sets used within a ZIP archive and the EBCDIC character set used on IBM-based systems. These tables may be created and modified by the user as documented in the User's Guide.

Trigger

A set of predefined actions that run automatically whenever a specified action or change occurs, for example, a change to a specified table or file. Triggers are often used to automate environments, such as running a backup when a certain number of transactions are processed.

Truncate

To cut off or delete the data that will not fit within a specified line width or display. This may also be attributed to data that does not fit within the specified length of a field definition.

User Interface

The actions or items that allow a user to interact with (and/or perform operations on) a computer.

Variable-Length

A characteristic of a file in which the individual records (and/or the file itself) can be of varying length. See also Fixed-Length.

ZIP64

ZIP64 is reference to the archive format that supports more than 65,534 files per archive, uncompressed files greater than 4 Gig and archives greater than 4 Gig.

ZIP Archive

A ZIP archive is used to refer to a single file that contains a number of files compressed into a much smaller physical space by the ZIP software.

3270 Display Emulation

A personal computer-based program that allows a PC to perform like a 5250 display station or printer. The program will allow use of the various iSeries system functions.

5250 Emulation

A personal computer-based program that allows a PC to perform like a 5250 display station or printer. The program will allow use of the various iSeries system functions.

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T