FAQS - z/OS

Generated: 2016-04-27 16:56:06.076000

Versions

Smartcrypt(r) for z / OS version 16.1 L1

SecureZIP for z / OS version 15.0 L16

Note: SecureZIP customers looking for version 16 should migrate to the new Smartcrypt for z/OS.

PKZIP for z / OS version 16.1 L1

The upgrade to SecureZIP for z/OS version 15 is Smartcrypt for z/OS version 16. If you are a SecureZIP for z/OS customer who has a current maintenance contract, you can upgrade to Smartcrypt for z/OS version 16 at no cost.

Supported versions of Smartcrypt, PKZIP and SecureZIP for z / OS are available for all releases of z / OS still supported by IBM. Not all supported versions of PKZIP and SecureZIP run on all recent releases of z / OS. PKZIP and SecureZIP versions 10 and older are no longer supported on any versions of z / OS.

Simply run a ZIP or UNZIP job and the version and level (LVL) will appear in a ZPLI001I message. Such as:

ZPLI001I SecureZIP(R) for z / OS, Version 10.0.0 - 09 / 27 / 07 11.21 LVL(0)
ZPLI001I PKZIP(R) for z / OS, Version 9.0.0b - 05 / 04 / 07 14.31 LVL(7)

Compatibility

A ZIP archive can be transferred from one platform to another, and can be decompressed or modified by a copy of PKZIP or PKUNZIP which is running on that platform. The internal format of a ZIP archive is identical no matter which platform compressed the files that it contains. To ensure compatibility, ZIP archives must be created by PKZIP 2.50 running on MS-DOS or PKZIP version 5 or higher on other platforms.
If you wish to transfer data across platforms using any other version of PKZIP or SecureZIP you should check with your supplier first to confirm that your versions of PKZIP / SecureZIP are compatible. Please also note that the PKZIP DCL products are not compatible with the PKZIP 2.50 for MS-DOS or PKZIP / SecureZIP version 8 products.

Both Smartcrypt and SecureZIP for z / OS offers all of the same features and functionality found in our PKZIP product lines. Smartcrypt / SecureZIP simply adds the ability to secure the data being stored or compressed within the ZIP file. PKZIP Enterprise Edition users can decrypt and decompress any passphrase-encrypted archive created by SecureZIP, even if a strong encryption algorithm (AES128, AES256, etc.) was used to encrypt the data.

Yes, this new system parameter, released in July, 2018 by IBM, has been tested with all PKWARE products and no issues have been identified when using PKWARE software with OSPROTECT=1 for the following releases: V15 and V16 of PKZIP for z/OS, SecureZIP for z/OS, SecureZIP Partner for z/OS, Smartcrypt for z/OS, and Smartcrypt Manager for z/OS. No updates to PKWARE software are necessary to support use of this parameter.

Yes, Smartcrypt for z/OS includes support for direct file transfer protocols. Refer to Chapter 15 in the User Guide for additional information.

Install / Licensing

\The first PKZIP / PKUNZIP job that is run on an unlicensed system (typically due to a CPU upgrade / downgrade or expired license) must be submitted by a user with write authority to the PKWARE license dataset to invoke the grace period.
This authority is only required the first time PKZIP / PKUNZIP is run on the unlicensed system; it is not required after the grace period has been successfully invoked (this is one time per CPU, not one time per IPL).
During the grace period, error messages will be displayed on the console (and the printout) for each execution of PKZIP / SecureZIP for z / OS. At the end of the period, if the license is not updated, the product will no longer function except to VIEW an archive. The five-day grace period is designed so that the program will not cease to function on a weekend or the Monday following the five-day grace period. You must contact PKWARE at pkcustomerservice@pkware.com during the grace period to obtain licensing to allow extended use.

The best place to obtain patches for PKWARE products for z / OS is in our updates section.

Yes, you will need a new license when upgrading from one version to another (i.e. version 14.0.0H to version 15.0.0) due to the differences in codes. Please keep in mind that you will not be able to apply your 15.x license to older product versions.
Additionally, if you are going from PKZIP to SecureZIP, or vice versa, a new key is required, even if the version number remains the same.
To request a key please contact pkcustomerservice@pkware.com .

PKWARE requires a license report be sent to our Customer Service Department when requesting a new or updated license key.
To run a license report you must edit and submit the LICSHSYS member located under the *.INSTLIB dataset.
Once the job has completed successfully please copy and paste the resulting output into an email addressed to Customer Service at pkcustomerservice@pkware.com

PKWARE has a 24 hour turn-around policy for returning authorization codes. There are exceptions to this policy (i.e. production is down, no jobs can run, etc.).
Contact Customer Service for appropriate turn around times at 937.847.2374, option 3, or via email at pkcustomerservice@pkware.com

PKZIP and SecureZIP for z / OS v14.0 license keys use a different logic than the previous PKZIP / SecureZIP version license keys. You will want to verify that the version of the key matches the version of the product the key it is being applied to.
If you do see that the versions do not match, then please contact Customer Service to receive a new license key for the appropriate version of the product you are using. You can contact Customer Service via Email, or by phone at 937-847-2374 and selecting option 3.
If the version does not seem to be the problem, another thing to verify is to begin all lines of the Zap in column two. Be sure all letters, spaces, commas, periods, etc are aligned exactly how the email shows them to be. If you continue to experience problems, please call PKWARE Technical Support at 937.847.2687 or submit your request online.

Enterprise consumption pricing is a new alternative pricing model that was introduced in June 2019. PKWARE offers the following license types which offer MIPS-based pricing options:

Machine License

Hardcap LPAR

Softcap LPAR

The PKWARE softcap LPAR license type can be used by customers choosing to migrate to IBM enterprise consumption pricing. This option uses routine SCRT reporting to facilitate licensing. With the introduction of enterprise consumption pricing, a new SCRT reporting tool is available from IBM. This change will introduce a revised reporting procedure for using PKWARE Softcap LPAR pricing. Details on this change will be available near the end of June, 2019.

Encryption / Decryption

PKWARE solutions utilize strong encryption so there is nothing that can be done if you lose or forget your password. It is important to remember your password as PKWARE has no special means for "getting around" the encryption and may not be able to assist in the recovery of an encrypted file.

Encryption is activated through the use of the -PASSWORD and / or -RECIPIENT commands. If a value is present for either setting, whether through commands or default settings, then encryption will be attempted in accordance with other settings (such as -ENCRYPTION_METHOD). However, if ENCRYPTION_METHOD=NONE is specified, then encryption will be bypassed.
Please note that certificate-based encryption with RECIPIENTs requires that one of the Strong ENCRYPTION_METHODs (minimum 128-bit) be selected.
BSAFE_AES128 - A SecureZIP implementation of the AES 128-bit key algorithm, including cipher-block-chaining and block padding. When recipient-based encryption is requested, this is the default encryption method unless the installation defaults module has been tailored differently.

Smartcrypt and SecureZIP for z / OS can encrypt data for security control with digital certificates and / or passwords. In addition to standard PKZIP encryption (96-bit), PKWARE has implemented Advanced Encryption Standard (AES), DES, 3DES and RC4 algorithms for added security and performance. Additional security features included are:

Support for Suite B is also provided. Selectable encryption methods
Filename Encryption Password masking Increased maximum password length (up to 200 characters) ISPF dialog password field blanking Cipher Block Chaining when any of the AES algorithms are used An enhanced key protection option

When creating ZIP file you will need to include the following in the JCL:

PASSWORD (password)
ENCRYPTION_METHOD (Standard | AES128 | AES192 | AES256 | DES | 3DES | RC4).

STANDARD supports standard encryption and is the default value.
AES128 uses the Crypto-C AES 128-bit key algorithm.
AES192 uses the Crypto-C AES 192-bit key algorithm.
AES256 uses the Crypto-C AES 256-bit key algorithm.
DES uses the Crypto-C DES key algorithm.
3DES uses the Crypto-C 3DES key algorithm.
RC4 uses the Crypto-C RC4 key algorithm.

Password-based encryption depends on both the sender and receiver knowing, and providing intellectual input (the password) in clear text. The password is used to derive a binary Master Session Key for each decryption run. No key information is kept within the ZIP Archive, therefore both parties must retain the password in an external location.
Recipient-based encryption provides a means by which the Master Session Key (MSK) information can be hidden, protected, and carried within the ZIP Archive. This is done by using technique known as Digital Enveloping with Public Key Encryption. The technique requires that the creating process have a copy of the recipient's Public Key Digital Certificate, which is used to protect and store the MSK. In addition, the receiving side must have a copy of the recipient's Private Key Digital Certificate. With these two pieces of information in place, there is no need for users to retain or recall a password for decryption.

Recipient-based encryption / decryption requires Public and Private key certificates kept in file streams encoded according to the x.509 standard.

Recipient-based encryption requires that public and private key certificates be used by Smartcrypt. These are kept in file streams encoded according to the X.509 standard. A certificate store is the location of where various types of certificates are kept and accessed. The primary store components used by SecureZIP include:

CSPUB: Certificate store for individual public-key X.509 certificates on the local system. CSPRVT: Certificate store for individual private-key X.509 certificates on the local system. CSPUB_DBX: VSAM index structure for the public-key X.509 certificates on the local system CSPUB_DBX_PATH_CN: VSAM index structure allowing the common name search and selection of public / private-key X.509 certificates on the local system. CSPUB_DBX_PATH_EM: VSAM index structure allowing the email search and selection of public / private-key X.509 certificates on the local system. CSPUB_DBX_PATH_PUBKEY: VSAM index structure allowing the identification of public / private-key X.509 certificates on the local system. CSCA: Certificate store for Certificate Authority public-key X.509 certificates on the local system. CSROOT: Certificate store for the Trusted Root public-key X.509 certificates on the local system. CSCRL: Certificate store for the certificate revocation lists maintained on the local system. LDAP: Certificate store for individual public-key X.509 certificates accessible via a TCPIP network.

You must first create and prime a local certificate store on the mainframe. You do so by properly configuring the PKISPF and PKZSTART members of the INSTLIB dataset, then following these steps ...

\Execute PKZSTART
Type A at the option line for the Administration Panel,
Type CS at the option line for the Certificate Store Panel,
Select option 1 for "Local Certificate Store Administration."
Type CREATE at the option line and complete the required files for HLQ, the New Database Profile and page down to add any specific SMS / non-SMS allocation parameters that may be needed.
When prompted, select to enable / disable the various security policy settings
Review and submit the JCL that is created.
Check for a Return Code of zero
A new database profile should have been created and set as your "Active DB Profile"
To verify proper setup choose option 8, then option 2 to run the Certificate Store IVP.
Distribute the DB Profile location to all appropriate users with the proper syntax (e.g. INCLUDE_CMD(YOUR.ACTIVE.DB(PROFILE))

Please note that Smartcrypt / SecureZIP for z / OS ships with four x.509 test key pairs (PKWARE Test1, PKWARE Test2, PKWARE Test3 and PKWARE Test4).
Only PKWARE Test3 and PKWARE Test4 have the trusted chain included (ROOT and INTERMEDIATE CA).

Upon a successful export of the digital certificate from the Certificate Authority Tool (e.g. Windows Internet Explorer | Tools | Options | Content | Certificates) you must perform a BINARY transfer of the certificate to the mainframe, storing the cert in a sequential file or as a member of a PDS. From the SecureZIP Local Certificate Store Administration Menu perform the following steps...

Select option 3 "Add new Certificates to the Local Store"
Enter the certificate PDS / file location (e.g. 'PKZIP.PUBLIC.CERT')
Assign a custom member name if desired (e.g. PKT2005)
Confirmation will appear in the top right corner of the screen saying the key was either added of failed (e.g. PKT2005 ADDED)

Please note when entering a PRIVATE key you must enter the password in order to add the PRIVATE key to the Local Store.

When using a digital certificate to encrypt a ZIP file you will need to include the following three parameters in the command stream:

INCLUDE_CMD(YOUR.SECZIP.DATABASE(PROFILE)
ENCRYPTION_METHOD (AES128 | AES192 | AES256 | DES | 3DES | RC4)
RECIPIENT (DB:CN=Common Name,R)

Please note that additional RECIPIENT options include searching for digital certificates stored on a LDAP server or in specific datasets on z System.
For additional details on implementing digital certificate encryption into the job stream please refer to Chapter 6 of the Smartcrypt for zOS User's Guide.

Smartcrypt / SecureZIP will automatically detect which encryption method was specified during the ZIP process and operate accordingly. A -VIEWDETAIL run can be performed to see which algorithms were used for various files.

Yes, when both RECIPIENT and PASSWORD settings are used, to encrypt a file, the Master Session Key is derived from the Password and also protected by using Public Key Encryption.
This means that a ZIP file may be decrypted either by a user who knows the password, or who has their private key certificate available.

The ZIP File format specification allows for a maximum recipient list size of 3,275. This can be restricted further by other file attributes associated with the data, and by run-time capacity limitations (such as virtual storage). (Note: Approximately 20 bytes is required for each recipient within the ZIP archive central directory record for each file. This area is limited to 64K in size).

When using recipient-based encryption, plan on an initial increase of 4MB of 31-bit storage for up to 15 recipients. LDAP will require an additional 1MB for every 27 recipients above 15. File-based and local certificate store will require an additional 1MB for every 41 recipients above 15.
Some SecureZIP operations require additional virtual storage to operate successfully. In particular, activation of certificate-based operations (for example, recipient-based encryption and digital signature operations) may necessitate increasing the 31-bit region size. It is recommended that the user evaluate the virtual storage used by a SecureZIP step in relation to the number of digital certificates used for a particular process and establish an appropriate REGION size.

Smartcrypt / SecureZIP for z / OS can locate a recipients' public-key certificates stored on your company's LDAP-compliant directories through the Directory Integration Module. This approach allows for recipient selection based on common name, email address, or other installation-configured LDAP fields. One or more LDAP-compliant servers may be configured for searching.
Your company's technical support staff responsible for configuring the LDAP compliant directory that stores certificates can provide you with the necessary information for the LDAP profile. For proper LDAP configuration please refer to Chapter 4 of the "SecureZIP for zOS System Administrators Guide."
To inquire about the Directory Integration Module, please contact your PKWARE account representative, or email sales at pkcustomerservice@pkware.com

To meet corporate security policies, SecureZIP provides the ability to include a "contingency-key" or master recipient certificate in a job when strong encryption is activated through the MASTER_RECIPIENT setting.
The MASTER_RECIPIENT may be set directly in the defaults module, or indirectly by specifying MASTER_RECIPIENT in a command stream referenced by SECUREZIP_CONFIG. This default-module-only setting specifies a PDSE member that contains SecureZIP certificate store configuration commands to be automatically included in the processing stream. The configuration command values from this member will be included at the start of command input processing prior to / / SYSIN statements being read. The data set(member) will be converted into an "INCLUDE_CMD=(pdse(member)" command internally and will be echoed to the message log in accordance with the ECHO setting.
SecureZIP certificate store configuration commands entered from other sources such as / / SYSIN will override the values read in from this source. This ensures the organization will always be able to extract and / or decrypt the secured archive / data using the "global" private key certificate.

When SecureZIP is being used to encrypt data, either with RECIPIENT or PASSWORD, a recipient specified by MASTER_RECIPIENT is automatically included. However, -MASTER_RECIPIENT alone does not trigger encryption to take place.

Someone who cannot decrypt the contents of an archive may still be able to infer sensitive information just from the unencrypted names of files. To prevent this, you can encrypt the names of files in addition to their contents.
Encrypted file names can be viewed in the clear-that is, unencrypted-only when the archive is opened by a recipient holding the authorizing keys or password, if the archive was encrypted using a recipient list, or by someone who has the password, if the archive was encrypted using a password. SecureZIP for z / OS encrypts filenames using your current settings for (strong) encryption method and algorithm.

Yes, Smartcrypt can be configured to operate using FIPS 140-2 Level 4 cryptographic modules.

Yes, Smartcrypt can be configured to operate using Suite-B algorithms.

Yes, while Smartcrypt will not directly export a smartkey from Smartcrypt Enterprise Manager, it can can use the passphrase retrieved from a Smartkey user to open a "smartkey-encrypted" file.

General

```
/ /
/ / FTPSTEP EXEC PGM=FTP,PARM='bigiron.pkware.com (EXIT'
/ / SYSPRINT DD SYSOUT=*
/ / INPUT DD *
FTP_SUPPORT
PKW!PKW
CWD USUPPORT
BINARY
PUT YOUR.FULLY.QUALIFID.DSN CLOSE QUIT
/ /
```

With the incorporation of ZIP64 technology, Smartcrypt / SecureZIP for z / OS offers an increased ZIP Archive size, raised from 4 gigabytes (32 bit binary counter) to a theoretical limit of over 17 billion gigabytes (64 bit binary counter). Large file support functionality is also available in PKZIP for z / OS v9.0 Enterprise Edition.

Smartcrypt / SecureZIP for z / OS offers an increased ZIP Archive capacity, raised from 65,535 to a theoretical limit of over 4 billion files. This increased capacity is also available in PKZIP for z / OS.

Smartcrypt / SecureZIP password-based encryption uses a symmetric implementation of keys, meaning that the same key is required for both encryption and decryption.
SecureZIP requires that the password be specified at both ends so that the internal encryption / decryption key can be derived algorithmically. No form of the password or key is carried in the ZIP Archive.
Two levels of key are derived from the password for processing:

Master Session Key
File Session Key

The process for deriving the keys are as follows:

If running on an EBCDIC-based platform, the password is converted to ASCII (so as to match the same password being entered on an ASCII-based system such as Windows).
The ASCII password is passed to a hashing function to generate a master session key. This key is used to encrypt / decrypt random data that is in turn used to generate a key for the file itself. (The password is not used to directly derive the File Session key for decrypting the data). In addition, a random initial vector (IV) is created for each file for cipher-block chaining, thereby ensuring that two encryption runs for the same data / password combination will result in different cipher streams.

A pseudo-code representation of the encryption process is as follows:
Password = GetUserPassword()
RD = Random()
ERD = Encrypt(RD,DeriveKey(SHA1(Password)))
For Each File
IV = Random()
VData = Random()
FileSessionKey = DeriveKey(SHA1(RD, IV))
Encrypt(VData + VCRC32 + FileData,FileSessionKey)

First, we recommend consulting the Service Support Schedule on the Announcements page to ensure your version of product is supported on the new version of z / OS. If possible, it is recommended going to the most recent versions of Smartcrypt, PKZIP or SecureZIP.

These are actually informational messages designed to assist SecureZip users with identifying if their Hardware Cryptography utility / hardware is active or installed.
For a PKZIP user, this is displayed for you so if you wish to contact Customer Service to upgrade to Smartcrypt / SecureZip from PKZIP, we have all of the information needed to better assist you.
If these messages are showing up as ending in an E (Error) and not an I (Informational), a code change is in place. Version 9 levelset 3 has this correction and is available at our website in our product updates section.

PKZIP for z / OS Enterprise Edition and all editions of Smartcrypt / SecureZIP offer the ability to create self-extracting archives on z System for the Windows and UNIX platforms only. For licensing and distribution requirements please contact your PKWARE account representative at 937.847.2374 or via email at pkcustomerservice@pkware.com .

PKZIP for z / OS includes the Assembler User API which provides a powerful tool for the user to dynamically take control of the format of data to be archived and the target data set names of files to be extracted.
The Two User APIs currently available are Data Record Transformation API for ZIP processing and File Name Manipulation API for UNZIP processing.
Data Record Transformation API for ZIP processing provides a means to restructure a data record before compression takes place. A common use is to transform binary and packed decimal fields into display-format numerics. This is useful when the system intended to receive the ZIP Archive does not easily handle these field formats.
File Name Manipulation API for UNZIP processing provides a means to transform filenames into manageable IBM MVS-compatible data set names. This is useful when specialized re-direction of files is required that surpasses the capabilities of the PKZIP for MVS command set.

Smartcrypt / PKZIP / SecureZIP for z / OS has several uses for temporary files. They are engaged at various times depending on the processing involved and the virtual storage resources available at execution time. Key uses of work files are:

A staged copy of a targeted ZIP archive (this is a permanent file with a temporary name)
Spill data sets to hold compressed data until it is ready for writing to the ZIP Archive.
Temporary files to manage ZIP file selection requests and Archive directory information.
ISPF user-files to hold generated JCL, utility message output and Browse extraction data. The allocation controls for these files are located on the PKZIP for z / OS Configuration screen. Note that large amounts of disk space may be required to Browse or Extract a file under ISPF.

! Note that installations running DFSMS / MVS or an equivalent product may find that file allocation requests made through PKZIP for z / OS result in different file characteristics than those requested. This is because Systems Managed Storage routines active in the system intercept allocation requests and have the ability to alter the location and / or attributes of a data set. Use of PKZIP for z / OS file allocation request parameters must be coordinated with the operating environment's Automatic Class Selection (ACS) routines that are in control at the installation. Check with your Storage Administration or Systems Programming staff for appropriate values (e.g. UNIT, DCB attributes) to be used for temporary files.

According to RFC1952 "GZIP file format specification version 4.3", data within a GZIP file structure may be held in either TEXT or BINARY format: "If FTEXT is set, the file is probably ASCII text. This is an optional indication, which the compressor may set by checking a small amount of the input data to see whether any non-ASCII characters are present. In case of doubt, FTEXT is cleared, indicating binary data. For systems which have different file formats for ASCII text and binary data, the decompressor can use FTEXT to choose the appropriate format."
PKZIP for z / OS uses the DATA_TYPE command (with options of BINARY, TEXT or DETECT) to govern how input data is to be handled during the compression process. The FTEXT flag is then set accordingly.

Once the GZIP file has been created on z System, FTP transfer the file in BINARY mode to the PC. Once the file resides on the PC, add the .GZ extension so the UNZIP utility will recognize the GZIP format.

The biggest issue to overcome when moving PKZIP images from Test to Production is the high level qualifiers you installed PKZIP with to the Test system. With this in mind, please make sure you perform the necessary steps below to ensure a smooth transition from Test to Prod.

Rename all DSN's to Prod naming scheme. i.e. (DEV.PKZIP.ZOS.HELP to PKZIP.ZOS.HELP)
Re-Assemble the ACZDFLT via the ASMDFLT to reflect the new PROD HLQ. You may need to make some changes to ACZDFLT to reflect the appropriate HLQ's.
Modify the PKZSTART of the INSTLIB to reflect the new Prod. HLQ.

That is it.

You may either provide the command -ARCHIVE_COMMENT( ) in the PKZIP job, or change the ACZDFLT default value for that command (See INSTLIB(ASMDFLT)).

How do I get rid of the need for the INSTLIB? Some installations desire to eliminate any allocation of a PARMLIB or CONFIG dataset through PARMLIB_DSNAME_ZIP and PARMLIB_DSNAME_UNZIP. If no installation-supplied dataset commands are desired, then ACZDFLT parameters may be set to bypass the allocation attempt. Only / / SYSIN DD and EXEC PARM='...' parameters will be processed. Both parameters should be set to avoid dataset allocations in the generation of ACZDFLT.
MCZDFLTS TYPE=CSECT,
LICENSE_HLQ=pkzip.mvs,
PARMLIB_DSNAME_ZIP=NULLFILE,
PARMLIB_DSNAME_UNZIP=NULLFILE

One of the features of PKZIP for z / OS is to allow a common input file for commands. This is controlled either by a / / PARMLIB DD statement in the job step, or via the ACZDFLT module. As distributed, the ACZDFLT module contains default parameters of PARMLIB_DSNAME_ZIP='PKKZIP.MVS.INSTLIB(CMDZIP)' and PARMLIB_DSNAME_UNZIP='PKZIP.MVS.INSTLIB(CMDUNZIP)'. These members are dynamically allocated for PKZIP and PKUNZIP processing respectively, and contain the commented commands shown above.
You may either edit the members shown above to remove the commented commands, or create a modified ACZDFLT module that points to another dataset member. (See INSTLIB(ASMDFLT))

During EXTRACT processing, PKUNZIP uses MVS Dynamic Allocation Services (SVC99) to look for the target dataset as "old" initially. If MVS dynamic allocation detects that the data set does not exist, then this informational message is issued by the system. PKZIP for z / OS handles this condition and continues by creating the target dataset before continuing.
The command -SUPPRESS_DYNALLOC_MSGS may be used to instruct MVS Dynamic Allocation Services to suppress messages such as these. However, be aware that other dynamic allocation messages may be suppressed as well.
Use the following to suppress these messages.
-SUPPRESS_DYNALLOC_MSGS will suppress the warning message

An Abend S80A is usually caused by insufficient 24-bit ("below the 16-megabyte line") storage in the Region for the job. System I / O functions such as QSAM use this storage area for buffering. If too many buffers are requested (e.g. DCB=BUFNO=nnnn) for a file, and the REGION parameter for the Job Step is too small, then system services may fail. Some adjustments that can be made in an attempt to alleviate the problem are: Try increasing the REGION for the JOB STEP.
If DCB=BUFNO was specified for one or more of the files for performance reasons, consider reducing the BUFNO value. If multi-tasking was requested for the job, consider reducing the number of tasks with the MULTI_THREAD_LIMIT command.

Try modifying the -DATA_TYPE command on the UNZIP side of the equation. If you are using BINARY, try changing this to TEXT.
The -DATA_TYPE(BINARY) command is used to direct PKZIP for z / OS to bypass EBCDIC to ASCII character translation. This feature is useful when the file contains non-text data (such as graphics or internal numeric representations), or if text-based data is to be extracted only to other EBCDIC-based platforms.
All data within a file is treated the same during ZIP processing in accordance with the -DATA_TYPE(TEXT) and -DATA_TYPE(BINARY) commands. Care should be taken when zipping files that may contain both text and binary data. Use of the -DATA_TYPE(TEXT) command when binary data exists within the file will produce unpredictable results for fields containing binary data. -DATA_TYPE(BINARY) should be used to preserve data integrity; however, this means that text data will not be translated to the ASCII format by UNZIP processing in a cross-platform environment.
In a case where neither -DATA_TYPE(TEXT) nor -DATA_TYPE(BINARY) is specified in either the command or installation default module, PKZIP for z / OS will read a portion of data from the input file (in accordance with the -DATATYPE_DETECT_DEPTH value) and scan it for non-translatable text characters using the active text translation table. If the number of translatable text characters (as specified by the -DATATYPE_DETECT_TABLE) meets or exceeds the percentage specified by -DATATYPE_TEXT_PERCENT, the file will be treated as -DATA_TYPE(TEXT). Otherwise, it will be treated as if DATA_TYPE(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 it is unknown whether a file in the ZIP Archive is text or binary, the user may use the -ACTION(VIEWDETAIL) command to examine the file attributes.
Note: It is possible for members of the same PDS or PDSE to be treated differently when -DATA_TYPE(DETECT) is used because of a varying mix of data. Each member is treated as an independent file during ZIP processing.

SAVE_LRECL(Y) must be specified when compressing BINARY data of variable length. This includes NONVSAM RECFM=U / V files and VSAM ESDS, KSDS and Variable-RRDS files. The record length information of each record is required for PKUNZIP to properly restore each record. Unpredictable results may occur during EXTRACT processing if SAVE_LRECL(N) is specified (e.g. data records streamed across output record boundaries, VSAM PUT failures, scrambled data with unexpected keys in VSAM KSDS files).
VSAM note: The RECORDSIZE parameter is commonly misunderstood. RECORDSIZE(80 80) in a Cluster definition does not accurately reflect whether all records in a Cluster are fixed in length.
The first RECORDSIZE parameter is used during an Access Method Services Define Cluster or Import to compute the amount of space to be used (when RECORDS is used in the DEFINE). The second RECORDSIZE parameter is used during Record I / O and limits the size of an output record. Any record size between 1 and 80 may be put to the file (with the exception of a fixed RRDS).

This is not a failure of PKZIP for z / OS. In this scenario, PKZIP for DOS, WinZip, PKZIP for i5 / OS, and PKZIP for Unix (just as an example) will decompress the same file without a problem. There are only two possible resolutions to this issue:

Create the archive with a RECFM of Variable or Undefined file type.
Ask the makers of the Java ZIP utility to add some tolerance to their ZIP utility like other ZIP developers have done.

With both resolutions, the resulting output ZIP file will not have trailing characters. We are at the mercy of other file types within the 390 environment, where padding the last record out will always occur.

When using ARCHIVE_RECFM=FB for Archives intended for off-system transport (via FTP or other electronic means), ARCHIVE_LRECL should also be tuned. The default for ARCHIVE_RECFM=U, which writes exactly the number of bytes required for the Archive. By changing to ARCHIVE_RECFM=FB, the defaults for ARCHIVE_LRECL ARCHIVE_BLKSIZE will still be half-track blocking (27998 on 3390 DASD). By leaving the defaults for these parameters alone, the operating system will write a complete record / block combination (rounded to 27998), resulting in dead space at the end of the archive; increasing transmission time and storage space on the target system. It is recommended that ARCHIVE_LRECL be set to a smaller size (ARCHIVE_BLKSIZE will automatically be set by PKZIP for MVS to match half-track blocking for a given LRECL). This allows the last block of the Archive to be a short block (as governed by QSAM in the operating system) thereby reducing wasted space.
The shortening of the ARCHIVE_LRECL must be balanced with the overhead associated with QSAM managing multiple records per block. A good start would be as follows:
-ARCHIVE_RECFM=FB -ARCHIVE_LRECL=1024

When using ARCHIVE_RECFM=FB for Archives intended for off-system transport (via FTP or other electronic means), ARCHIVE_LRECL should also be tuned.
The default is ARCHIVE_RECFM=U, which writes exactly the number of bytes required for the Archive. By changing to ARCHIVE_RECFM=FB, the defaults for ARCHIVE_LRECL ARCHIVE_BLKSIZE will still be half-track blocking (27998 on 3390 DASD). By leaving the defaults for these parameters alone, the operating system will write a complete record / block combination (rounded to 27998), resulting in dead space at the end of the archive; increasing transmission time and storage space on the target system.
It is recommended that ARCHIVE_LRECL be set to a smaller size (ARCHIVE_BLKSIZE will automatically be set by PKZIP for z / OS to match half-track blocking for a given LRECL). This allows the last block of the Archive to be a short block (as governed by QSAM in the operating system) thereby reducing wasted space. The shortening of the ARCHIVE_LRECL must be balanced with the overhead associated with QSAM managing multiple records per block. A good start would be as follows:
-ARCHIVE_RECFM=FB
-ARCHIVE_LRECL=1024

The DST change will not affect our z / OS products.

The following file formats may be used to house an OpenPGP Key Ring:

RECFM=FB (Sequential or PDS)
RECFM=VB (Sequential or PDS)
RECFM=U (Sequential only)
Unix File System (Sequential binary stream without FileFmt=NL)

Only binary stream copies of a Key ring are supported. (ASCII ARMOR is not supported).
When exporting and transferring keys to z / OS, ensure that BINARY mode is used for both.

Example: A decryption attempt was made using only a passphrase when the file had been encrypted with both a passphrase and public-keys.
ZPGP017I encrypted with Key
ID=A6E4EF67F22DF11D

ZPGP017C unknown
KeyID

ZPGP014I PGP ERRCode=3390. unwrapKey ID=A6E4EF67F22DF11D No Store Provided for
unwrapPublic-Key Encrypted Session Key
Packet

ZPGP018I File was encrypted using AES128 Algorithm for Packet Type 18
ZPAM031I ZIPFORMAT=OpenPGP UNIT=3390 BLKSIZE= 27998
ZPEX001I tested okay SEG / BASE82 / ALLOC

These are informational messages that you can consider as logging information. As the explanation for ZPGP014I indicates, "User Response: No action is required.
The ZPGP014I explanation says that they are to be used by PKWARE Tech Support to assist in diagnostics for unresolved issues. Since there is not an unresolved issue (the archive opened with the password), the information may be ignored.
Tip! - The OpenPGP file format has multiple key definitions that precede the encrypted file data. These are vetted as they are encountered in the file stream. In this case, the ZPGP014I is issued to let us know that the public-key packet could not be used in the decryption process should it be required later in the work flow.

In general, any media or file format supported for a ZIP archive is also supported for an input OpenPGP file.
Some processing restrictions apply based on the OpenPGP file architecture. For example:

The directory information is not independently accessible from the data portion of the OpenPGP file.

If encrypted, a decryption key is required to perform a View of the filename and related metadata stored in the OpenPGP Literal Tag ARCHIVE_FASTSEEK processing used to locate the ZIP Central Directory does not apply. Decryption and Inflation are required to access the filename and other metadata.
Inasmuch as the OpenPGP file format does not retain file data sizes (compressed or uncompressed), in order for a VIEW request to report on file size information, the entire data file must be decrypted and decompressed for byte counts to be accumulated.

The OpenPGP file format does not support the retention of file attributes as ZIP archives do. So, allocation amounts and DCB attributes cannot be retained for use during extraction.

When creating an OpenPGP file with either Passphrase or Public-key encryption, contingency keys may be configured to be included for decipherment by a set of installation-defined keys that are supplemental to those directly requested by the user.
-

Unknown macro: {PGPKEYRDEF=CONTING;PUB;FILE;keyring-file}

Define a read-accessible OpenPGP Key Ring file (keyring-file) containing the public keys to be used as contingency keys.
Configure PARMLIB_DSNAME_ZIP with a definition that references the key ring:
Configure the defaults module setting MASTER_RECIPIENT to reference the keys to be used as contingency keys. A form of the setting may be chosen such that multiple, or specific keys be included from the key ring.

MASTER_RECIPIENT=PGP:CONTING
MASTER_RECIPIENT=PGP:CONTING,KEYID=xxxxxxx
MASTER_RECIPIENT=PGP:CN=name
MASTER_RECIPIENT=PGP:EM=email_address

Not at the present time.

Yes, the installation and maintenance procedures have been simplified. View the README in the installation package for information on this change.

V 15.0.1 includes a new command SELECT_LIBRARY_NONPDS to designate whether a requested data set mask should include non-partitioned data sets in the selection process.

INSTLIB(LPACOPY) contains a module identification list for modules which are marked as RENT and are appropriate for LPA use. This list should be referenced for the specific release and maintenance level as this list may change over time. Most applicable modules are 31-bit residency mode and impact Extended LPA rather than 24-bit LPA. Review the residency mode attributes if 24-bit LPA is constrained in your operating environment.

This is a result of the new command in v15 for utilizing zEDC or zIIP hardware. There are three options, suppress the return code, turn off use of the ZEDC / ZIIP option, or authorize the services. The first two options are coded in the ACZDFLT member to cover all jobs, or can be included in specific jobs only.
To suppress the return code (message will still display in the sysprint)
Add the -PKSUPPR=ZPCM123W statement
To turn off the message return code entirely (disables the use of ZEDC / ZIIP hardware)
Add the -FACILITY_COMPRESS=PKZIP statement
Either will address the RC=04 that may cause an issue.
However, if you wish to use either zIIP or zEDC hardware that would entail the following configuration tasks to be completed / verified for zIIP / zEDC workload redirection to be attempted:
Provide APF authorization to PKZIP / SecureZIP in one of two ways:
The System Administrator#rsquo;s guide covers APF Authorization, reference the section on Enabling zIIP Processing or Enabling zEDC Processing.

APF authorize the execution library (or libraries) from which Smartcrypt / PKZIP / SecureZIP will execute.
Install and activate the PKWSVC authorizing the services.

Smartcrypt / SecureZIP Enterprise Edition customers can use the KeyMaker program to convert an OpenPGP key to the X.509 Digital Certificate format before importing into RACF. Some OpenPGP keys may not have expiration dates. When the key is converted to X.509 format, an expiration date is required. KeyMaker will default to setting an expiration date value to one year past the date the OpenPGP key was created. This may result in a converted key having an expiration date in the past. To avoid this issue, use the -expire option to KeyMaker to set an alternate expiration date when converting to X.509 format.

Smartcrypt / SecureZIP Enterprise Edition customers can use the KeyMaker program to convert an OpenPGP key to the X.509 Digital Certificate format before importing into RACF. OpenPGP keys use a different trust model than X.509 keys. When the key is converted to X.509 format, a trusted ROOT is not provided with a trust period of longer than 1 day and the converted key will not be trusted when imported to RACF. To address this issue the key should be explicitly marked as trusted within RACF.

Using the XML_VIEW command available in V15.0.6 produces XML formatted output.

When performing high volume operations such as may occur when using Field Level Encryption, you can remove some overhead that exists due to SAF calls performed for each cryptographic operation. You can disable SAF checks using an available patch from IBM for ICSF version HCR77A1. Information on this patch is available at https: / / www-03.ibm.com / support / techdocs / atsmastr.nsf / WebIndex / FLASH10818. Refer to the section labeled OWH / RNG Authorization Access.

OpenPGP technology includes both a file format and an encryption specification. PKZIP will open "unencrypted" OpenPGP files and OpenPGP files that are signed but not encrypted. PKZIP SE will not allow creating or opening OpenPGP files that use encryption.

FAQ: Container Pricing for IBM Z

Smartcrypt Programming API

Using Smartcrypt Application Integration developers can integrate Smartcrypt encryption and compression functions into applications. Support is available for C, COBOL, ASM, PL/1, CICS, and Java. For additional information refer to the Smartcrypt for z / OS Application Integration Guide and /or the Getting Started with Smartcrypt SDK for Java.

Yes, Smartcrypt supports operations on both structured (often referred to as field-level encryption) and unstructured (files) data.

Yes, using Smartcrypt for z / OS, format-preserving encryption is available through the Visa Format-preserving encryption services. Additionally, for data moving between mainframe and other systems (i.e. Window, Linux, IBM i) the Smartcrypt SDK for Java can provide cross-platform format and length preserving encryption.

Smartcrypt Enterprise Manager for z / OS

The PKWARE Smartcrypt Enterprise Manager for z / OS provides interoperable key management and licensing support for Smartcrypt for z / OS. While not directly integrated today with the Smartcrypt Enterprise Manager for Windows, on z / OS, the Manager provides integration and key management services for z / OS native keystores as well as integration with 3rd party key providers such as Gemalto Safenet KeySecure. This manager on z / OS operates through a separate program using its own set of libraries, managing selected service agents for providing client services. One example of an agent is the PKWARE KeyAgent running under Unix System Services. This agent can deliver keys from KeySecure for registration within ICSF for use by Smartcrypt, or by other application programs. The manager communicates with clients, such as Smartcrypt for z / OS, through local z / OS facilities such as cross-memory services. Refer to the Smartcrypt Manager for z / OS Administrators Guide for more information about this manager.