Integrate SecureZIP with DFSMSdss and FDR Backup Processing

PKZIP has long been a trusted utility for backing up datasets on all platforms, including the zSeries platform. PKZIP's compressed archives provide an ideal container for efficiently storing copies of critical data onsite or offsite.

SecureZIP protects the confidentiality of your data by adding enterprise-grade encryption to PKZIP's existing capabilities. Using X.509 V3 digital certificates and the NIST-approved, FIPS 197-certified, AES 256-bit algorithm, SecureZIP provides strong, public/private key encryption that integrates easily with your backup processing. SecureZIP also provides other security-related features, including these:

- **Contingency-key processing** to ensure that the organization can always decrypt encrypted data
- **Filename encryption** to ensure that file metadata is protected from malicious interception
- **Digital signing/authentication** to validate identity and guard against data tampering

SecureZIP's ability to back up critical business datasets—including files created by traditional mainframe backup utilities like DFSMSdss and FDR—and to store these datasets in strongly encrypted archives using the industry-accepted, cross-platform ZIP format is gaining SecureZIP an increasingly important place in the backup strategies of the zSeries data center.

Use SecureZIP Archiving with DFSMSdss Processing

SecureZIP can compress and strongly encrypt an entire DFSMSdss backup inside a SecureZIP archive. DFSMSdss backup datasets are normally cataloged in the MVS catalog, and the volumes are managed using DFSMSrmm or CA-1.

Note: For SecureZIP to read output DFSMSdss files, the DFSMSdss JCL should specify on the output DD card a DCB blocksize of 32760

Archive a DFSMSdss Backup

Here is sample JCL that can be used with SecureZIP to encrypt and archive a DFSMSdss backup.

```
//DSSZIP PROC DUNIT=DISK,VOLSER='REQUIRED',OUTVOL='REQUIRED',
// DSSDSN='REQUIRED',TUNIT=CART
//*****************************************************************
//*** THIS PROC HAS TWO STEPS
//*** THE FIRST STEP INVOKES DFSMSdss TO BACKUP A SET OF DATASETS
//*** THE SECOND STEP INVOKES SECUREZIP TO ZIP THE OUPUT OF THE dss
//*** THE SYMBOLIC PARAMETERS FOR THIS STEP ARE:
//*** TUNIT - USED TO SPECIFY THE TAPE UNIT TYPE - DEFAULT=CART
//*** DUNIT - USED TO SPECIFY UNIT PARAMETER ON THE DASD DD CARD
//*** VOLSER - USED TO SPECIFY VOLUME SERIAL OF DASD DD CARD
//*** OUTVOL - USED TO SPECIFY VOLUME SERIAL OF THE dss OUTPUT TAPE
//*** DSSDSN - USED TO SPECIFY THE DSN OF THE OUTPUT FILE OF dss
//*** TO RUN - ENTER SYMBOLIC PARMS AND ENTER THE PASSWORD TO USE
//*** AND THE DATASETNAME MASK TO DUMP WITH dss
//*****************************************************************
//DSS EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD UNIT=&DUNIT.,VOL=(PRIVATE,SER=&VOLSER.),DISP=OLD
//TAPE     DD UNIT=&TUNIT.,
//            DCB=(BLKSIZE=32760),
//            LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=&DSSDSN.
//ZIPIT    EXEC PGM=SECZIP,COND=(4,LT,DSS)
//SYSPRINT DD SYSOUT=*
//INFILE1  DD DSN=&DSSDSN.,
//            DISP=(OLD,KEEP),
//            UNIT=&TUNIT.
//ARCHOUT  DD DSN=&ZIPDSN.,UNIT=&TUNIT.,DISP=(NEW,CATLG),
//            DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760),LABEL=(1,SL)
// PEND
//DOIT EXEC DSSZIP,DSSDSN='dss.output.tape',
// VOLSER=disk01,OUTVOL=dsstap,ZIPDSN=ZIPPED.ARCHIVE
//DSS.SYSIN DD *
DUMP DATASET(INCLUDE(my.datasetname.mask)) -
INDDNAME(DASD) OUTDDNAME(TAPE)
/*
//ZIPIT.SYSIN    DD *
-ENCRYPTION_METHOD(BSAFE_AES256)
-PWD(|
 Put your password here)
-ARCHIVE_OUTFILE(ARCHOUT)
-ADD
-INFILE(INFILE1)
-DATA_TYPE(BINARY)
-SAVE_LRECL=Y
//
```

NOTE* Assuming the tape drive has LBI support, changing the following on the DFSMSdss / FDR and the PKZIP / Encrypt steps could possibly cut the execution time in half.

```
on the DFSMSdss / FDR step
    //TAPE DD UNIT=TAPE,DCB=(BLKSIZE=32760) to
    //TAPE DD UNIT=TAPE,BLKSZLIM=229376
on the PKZIP step (no DCB=)
    DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760)  to
    RECFM=U,LRECL=0,BLKSZLIM=229376
```

Encrypt and Archive Before Using DFSMSdss

SecureZIP can also be used to encrypt your datasets into a SecureZIP archive before processing with DFSMSdss. The following sample JCL shows how.

```
//ZIPDSS PROC DUNIT=DISK,VOLSER='REQUIRED',OUTVOL='REQUIRED',
// DSSDSN='REQUIRED',TUNIT=CART,ZUNIT=DISK,
// DSPACE='REQUIRED'
//*****************************************************************
//*** THIS PROC HAS TWO STEPS
//*** THE FIRST STEP INVOKES SECUREZIP TO BACKUP A SET OF DATASETS
//*** THE SECOND STEP INVOKES DFSMSdss TO DUMP THE OUPUT OF THE ZIP
//*****************************************************************
//ZIPIT    EXEC PGM=SECZIP
//SYSPRINT DD SYSOUT=*
//ARCHOUT  DD DSN=&ZIPDSN.,UNIT=&ZUNIT.,DISP=(NEW,CATLG),
//            DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760),SPACE=&DSPACE,
//            VOL=(PRIVATE,SER=&VOLSER.),
//DSS EXEC PGM=ADRDSSU,COND=(8,LT,ZIPIT)
//SYSPRINT DD SYSOUT=*
//DASD     DD UNIT=&DUNIT.,VOL=(PRIVATE,SER=&VOLSER.),DISP=OLD
//TAPE     DD UNIT=&TUNIT.,VOL=SER=&OUTVOL.,
//            DCB=(BLKSIZE=32760),
//            LABEL=(1,SL),DISP=(NEW,CATLG),DSNAME=&DSSDSN.
// PEND
//DOIT EXEC ZIPDSS,DSSDSN='dss.output.tape',
// VOLSER=disk01,OUTVOL=dsstap,ZIPDSN=ZIPPED.ARCHIVE
//ZIPIT.SYSIN    DD *
-ENCRYPTION_METHOD(BSAFE_AES256)
-PWD(|
 Put your password here)
-ARCHIVE_OUTFILE(ARCHOUT)
-ADD
-DATA_TYPE(BINARY)
-SAVE_LRECL=Y
my dataset name masks here
//DSS.SYSIN DD *
DUMP DATASET(INCLUDE(my.datasetname from &zipdsn))
- INDDNAME(DASD) OUTDDNAME(TAPE)
/*
//
```

NOTE* Assuming the tape drive has LBI support, changing the following on the DFSMSdss / FDR and the PKZIP / Encrypt steps could possibly cut the execution time in half.

```
on the DFSMSdss / FDR step
    //TAPE DD UNIT=TAPE,DCB=(BLKSIZE=32760) to
    //TAPE DD UNIT=TAPE,BLKSZLIM=229376
on the PKZIP step (no DCB=)
    DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760)  to
    RECFM=U,LRECL=0,BLKSZLIM=229376
And add this to the sysin:
    -ARCHIVE_ZIPFORMAT(FULL_LBI)
```

Use SecureZIP Archiving with FDR Processing

One method of using SecureZIP to encrypt data when backing up with FDR is to encrypt your entire FDR full-volume backup inside a SecureZIP archive.

Note: For SecureZIP to read FDR output files, the FDR backups should be created using the FDR control parameter FORMAT=SPLIT.

Encrypt and Archive Entire FDR Backup

Here is sample JCL that can be used with SecureZIP and FDR to create an FDR full-volume backup inside a SecureZIP archive:

```
//FDRZIP PROC TAPEDSN='REQUIRED',RETPD=390,
// TDUMMY=' ',  ENTER TDUMMY='DUMMY ' TO DUMMY OUT THIS DD CARD
// DISK='REQUIRED',VOLSER='REQUIRED',DISP=OLD,
// ZIPDSN='REQUIRED',TUNIT=CART
//*****************************************************************
//*** THIS PROC HAS TWO STEPS
//*** THE FIRST STEP INVOKES FDR TO BACKUP A VOLUME
//*** THE SECOND STEP INVOKES SECUREZIP TO ZIP THE OUTPUT OF THE FDR
//*** THE SYMBOLIC PARAMETERS FOR THIS STEP ARE:
//*** TDUMMY - USED TO DUMMY OUT THE DISK1 JCL STATEMENT
//*** TAPEDSN - USED TO SPECIFY THE DATASET NAME OF THE TAPE DATASET
//*** TUNIT - USED TO SPECIFY THE TAPE UNIT TYPE - DEFAULT=CART
//*** RETPD - USED TO SPECIFY THE RETENTION PERIOD OF THE FDR TAPE
//*** DISK - USED TO SPECIFY UNIT PARAMETER ON THE DISK1 DD CARD
//*** VOLSER - USED TO SPECIFY VOLUME SERIAL OF DISK1 DD CARD
//*** DISP - USED TO SPECIFY THE DISPOSITION OF THE DISK1 DD CARD
//*** TO RUN - ENTER THE SYMBOLIC PARMS AND ENTER THE PASSWORD TO USE
//*****************************************************************
//FDR EXEC PGM=FDR,REGION=8M,PARM='DUMP TYPE=FDR,FORMAT=SPLIT'
//SYSPRINT  DD SYSOUT=*
//SYSPRIN1  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//TAPE1     DD %TDUMMY.DSN=&TAPEDSN.,
//             DISP=(NEW,CATLG),
//             UNIT=&TUNIT.,
//             LABEL=RETPD=&RETPD.
//DISK1     DD UNIT=&DISK.,VOL=SER=&VOLSER.,DISP=&DISP.
//SYSIN     DD DUMMY
//ZIPIT    EXEC PGM=SECZIP,COND=(0,NE,FDR)
//SYSPRINT  DD SYSOUT=*
//INFILE1   DD %TDUMMY.DSN=&TAPEDSN.,
//             DISP=(OLD,KEEP),
//             UNIT=&TUNIT.
//ARCHOUT   DD DSN=&ZIPDSN.,UNIT=&TUNIT.,DISP=(NEW,CATLG),
//             DCB=(RECFM,LRECL=32760,BLKSIZE=32760),LABEL=(1,SL)
// PEND
//DOIT EXEC FDRZIP TAPEDSN=FDRZIP,DISK=DISK,VOLSER=MYVOL1,
// ZIPDSN=ZIPPED.ARCHIVE
//ZIPIT.SYSIN    DD *
-ENCRYPTION_METHOD(BSAFE_AES256)
```

```
-PWD(|
 Put your password here)
-ARCHIVE_OUTFILE(ARCHOUT)
-ADD
-INFILE(INFILE1)
-DATA_TYPE(BINARY)
-SAVE_LRECL=Y
//
```

NOTE* Assuming the tape drive has LBI support, changing the following on the DFSMSdss / FDR and the PKZIP / Encrypt steps could possibly cut the execution time in half.

```
on the DFSMSdss / FDR step
    //TAPE DD UNIT=TAPE,DCB=(BLKSIZE=32760) to
    //TAPE DD UNIT=TAPE,BLKSZLIM=229376
on the PKZIP step (no DCB=)
    DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760)  to
    RECFM=U,LRECL=0,BLKSZLIM=229376
And add this to the sysin:
    -ARCHIVE_ZIPFORMAT(FULL_LBI)
```

When using the method above, the FDR backup catalog only references the original FDR tapes used to create the FDR backup. Cataloging the SecureZIP archive causes the MVS catalog to contain the list of volumes in the SecureZIP archive. The central directory in the SecureZIP archive contains the name of the FDR backup dataset stored in the archive. Decrypting this dataset using SecureZIP makes this information available for use during recovery processes using FDR. A removable media manager like DFSMSrmm or CA-1 can be used to manage the locations of volumes.

Encrypt and Archive Before Using FDR

Another approach is to encrypt your datasets into a SecureZIP archive before using FDR. FDR will see the SecureZIP archive as an MVS dataset, and the FDR catalog will store the name of the SecureZIP archive.

Here is sample JCL that can be used with SecureZIP and FDR to create such an archive.

```
//ZIPFDR PROC TAPEDSN='RE..QUIRED',RETPD=390,
// TDUMMY=' ',  ENTER TDUMMY='DUMMY ' TO DUMMY OUT THIS DD CARD
// ZIPDSN='REQUIRED',TUNIT=CART
//*****************************************************************
//*** THIS PROC HAS TWO STEPS
//*** THE FIRST STEP INVOKES SECZIP TO BACKUP SPECIFIED DATASETS
//*** THE SECOND STEP INVOKES FDRDSF TO ZIP OUTPUT OF THE FDR
//*** THE SYMBOLIC PARAMETERS FOR THIS STEP ARE:
//*** TDUMMY - USED TO DUMMY OUT THE DISK1 JCL STATEMENT
//*** TAPEDSN - USED TO SPECIFY DATASET NAME OF THE FDR TAPE
//*** TUNIT - USED TO SPECIFY THE TAPE UNIT TYPE - DEFAULT=CART
//*** RETPD - USED TO SPECIFY THE RETENTION PERIOD OF FDR TAPE
//*** DISK - USED TO SPECIFY UNIT PARAMETER ON THE DISK1 DD CARD
//*** VOLSER - USED TO SPECIFY VOLUME SERIAL OF DISK1 DD CARD
//*** DISP - USED TO SPECIFY THE DISPOSITION OF THE DISK1 DD CARD
//*** TO RUN - ENTER THE SYMBOLIC PARMS AND ENTER THE PASSWORD TO USE
//*****************************************************************
```

```
//ZIPIT    EXEC PGM=SECZIP
//SYSPRINT DD SYSOUT=*
//ARCHOUT DD DSN=&ZIPDSN.,UNIT=&TUNIT.,DISP=(NEW,CATLG),
// DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760),LABEL=(1,SL)
//*
//FDR EXEC PGM=FDRDSF,REGION=8M,
// PARM='DUMP TYPE=DSF/ SELECT DSN=&ZIPDSN.',
// COND=(4,LT,ZIPIT)
//SYSPRINT  DD SYSOUT=*
//SYSPRIN1  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//TAPE1     DD %TDUMMY.DSN=&TAPEDSN.,
//             DISP=(NEW,CATLG),
//             UNIT=&TUNIT.,
//             LABEL=RETPD=&RETPD.
//DISK1     DD UNIT=&TUNIT.,DISP=(OLD,KEEP),DSN=&ZIPDSN.
//SYSIN     DD DUMMY
// PEND
//DOIT EXEC ZIPFDR TAPEDSN=FDRZIP,ZIPDSN=zipped.archive
//ZIPIT.SYSIN    DD *
-ENCRYPTION_METHOD(BSAFE_AES256)
-PWD(|
 Put your password here)
-ARCHIVE_OUTFILE(ARCHOUT)
-DATA_TYPE(BINARY)
-ACTION(ADD)
-SAVE_LRECL=Y
list.of.datasetname.masks.here
//
```

NOTE* Assuming the tape drive has LBI support, changing the following on the DFSMSdss / FDR and the PKZIP / Encrypt steps could possibly cut the execution time in half.

```
on the DFSMSdss / FDR step
    //TAPE DD UNIT=TAPE,DCB=(BLKSIZE=32760) to
    //TAPE DD UNIT=TAPE,BLKSZLIM=229376
on the PKZIP step (no DCB=)
    DCB=(RECFM=FB,LRECL=32760,BLKSIZE=32760)  to
    RECFM=U,LRECL=0,BLKSZLIM=229376
And add this to the sysin:
    Thx
    -ARCHIVE_ZIPFORMAT(FULL_LBI)
```

Depending on the needs of the data center, more than one of these methods may be appropriate. If you have questions about which method to use, please contact PKWARE, Inc. at www.pkware.com or 800-4PKWARE.