Encryption Key Serving

Fred Lates
Sue Marcotte

z/OS® Platform Evaluation Test / Integration Test

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Agenda

- Why encrypt ?
- Data at rest encryption
- Requirements
- Solution
- IBM® key serving products
- I/T environment overview
- Installation of ISKLM
- Keys
- IOS and DFSMS setup
- Running ISKLM
- Auditing
- Process for establishing backup key server
- Documentation

Why encrypt?

- Protect data from unauthorized access
  - Proprietary/business assets
  - Customer personal information
  - Business/Customer financial data
  - Secure disposal of old/unused media

Data at rest encryption

- Application based
  - Performance concerns
  - Loss of data
  - Recovery of data (DR)
  - Key management
  - Difficulty in transport of data to business partner/client

Requirements

- Ability to move “data in the clear” on disk to “data encrypted” on tape , with ease
- Little or negligible performance impacts
- Use hardware based keys
- Minimize risk for loss of data
  - Ability to serve keys from an alternative location
  - Minimize risk when disposing of media

Solution

- Encrypt at the drive
  - Ease of use
  - Minimal performance impact
  - Reliable
- Secure Key
  - Hardware encryption
- Multiple Key Servers
  - Minimize risk for loss of data

© 2011 IBM Corporation

Key Serving Product History

- Enterprise Key Manager
  - Tape media encryption only
  - Manages JCE, JCECCA, JCERACFKS and JCERACFCCAKS keystores
- Tivoli Key Lifecycle Manager
  - Tape and Disk encryption
  - Uses WAS(SSRE) GUI
  - Required DB2
- IBM Security Key Lifecycle Manager
  - Tape and disk encryption
  - Similar to EKM

Integration Test Environment

- 2 Parallel Sysplex Environments
  - 4 image sysplex – Runs ISKLM
  - 9 image sysplex – Runs EKM
- Each Sysplex has its own RACF® database
- Utilize Shared HFS for Unix System Services
- 3592 tape drives
- ICSF runs on all images

Installation of ISKLM

- Created new directory /ISKLM and mounted the product filesystem
  - IBMSKLM.jar   (ISKLM product jar file)
  - samples/ISKLMConfig.properties.zos
  - samples/PROCLIB.ISKLM
  - samples/PROCLIB.ISKLMENV
  - samples/SMF2LOG.JCL

Installation of ISKLM

- Verify Java™ SDK 5.0 or 6.0 is installed

```
export PATH=/java/java631UK58682/J6.0/bin:$PATH
java -version
java version "1.6.0"
Java(TM) SE Runtime Environment (build pmz3160sr8fp1-
    20100624_01(SR8 FP1))
IBM J9 VM (build 2.4, JRE 1.6.0 IBM J9 2.4 z/OS s390-31
    jvmmz3160sr8ifx-20100609_59383 (JIT enabled, AOT enabled)
J9VM - 20100609_059383
JIT  - r9_20100401_15339ifx2
GC   - 20100308_AA)
JCL  - 20100624_01
```

Filesystems for config parm and logs

- Created 2 new filesystems and mountpoints for config parm & logs. Also updated BPXPRM00 in sys1.parmlib

```
MOUNT FILESYSTEM('OMVSSPT.ISKLM.ETC.ZFS') TYPE(ZFS)
   MODE(RDWR) MOUNTPOINT('/KLMETC')

MOUNT FILESYSTEM('OMVSSPT.ISKLM.LOGS.ZFS') TYPE(ZFS)
   MODE(RDWR) MOUNTPOINT('/KLMLOGS')
```

  – Created directories for each system in both of the filesystems
    - Z1, Z2, Z3, Z4

Migration from EKM

- Files needed to be saved from prior EKM instance
  - Configuration file
  - Device table
  - Metadata file
- Review the ISKLM sample config file and customize accordingly with Keystore properties from EKM i.e. /KLMETC/Z2
  - ISKLMConfig.properties.zos
- Copy the EKM device table and metadata file to the new ISKLM directory.
  - filedrive.table
  - SKLMData.xml

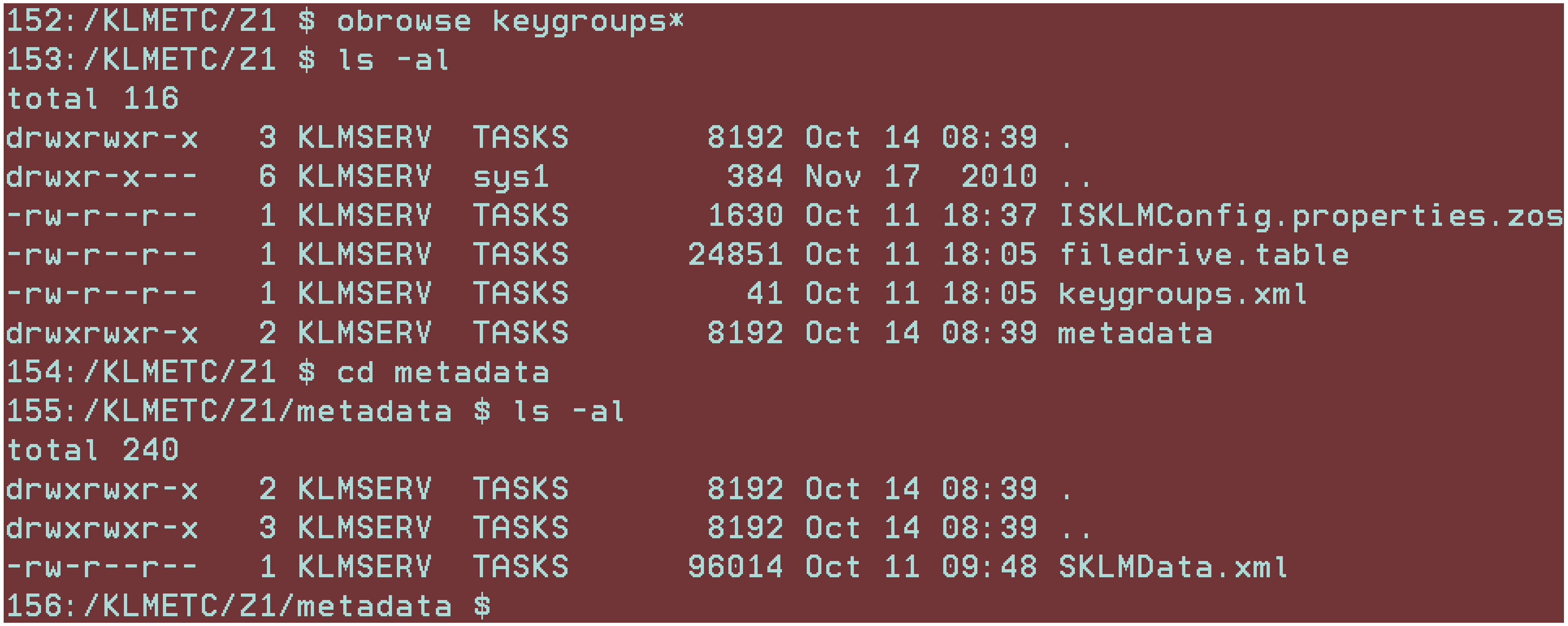
Metadata

As encryption processing is performed, the Security Key Lifecycle Manager for z/OS collects the following data:

- Drive Serial Number
- Drive WorldWideName
- Creation Date
- Key Alias 1
- Key Alias 2

```
</KeyUsageEvent><KeyUsageEvent>
<driveSSN>000007888422</driveSSN>
<volSer>WXY662</volSer>
<driveWWN>500507630240FBCC</driveWWN>
<keyAlias2>Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21</keyAlias2>
<keyAlias1>Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20</keyAlias1>
<dateTime>Tue Oct 18 19:35:08 GMT 2011</dateTime>
</KeyUsageEvent></KeyUsageEvents>
```

View of directories & files



```
152:/KLMETC/Z1 $ obrowse keygroups*
153:/KLMETC/Z1 $ ls -al
total 116
drwxrwxr-x   3 KLMSERV  TASKS       8192 Oct 14 08:39 .
drwxr-x---   6 KLMSERV  sys1         384 Nov 17  2010 ..
-rw-r--r--   1 KLMSERV  TASKS       1630 Oct 11 18:37 ISKLMConfig.properties.zos
-rw-r--r--   1 KLMSERV  TASKS      24851 Oct 11 18:05 filedrive.table
-rw-r--r--   1 KLMSERV  TASKS         41 Oct 11 18:05 keygroups.xml
drwxrwxr-x   2 KLMSERV  TASKS       8192 Oct 14 08:39 metadata
154:/KLMETC/Z1 $ cd metadata
155:/KLMETC/Z1/metadata $ ls -al
total 240
drwxrwxr-x   2 KLMSERV  TASKS       8192 Oct 14 08:39 .
drwxrwxr-x   3 KLMSERV  TASKS       8192 Oct 14 08:39 ..
-rw-r--r--   1 KLMSERV  TASKS      96014 Oct 11 09:48 SKLMData.xml
156:/KLMETC/Z1/metadata $
```

Sample of our configuration file

```
Admin.ssl.keystore.name=safkeyring\://TAPEKMS/KeyStore4758
Audit.handler.file.size=10000
debug.output.file=/KLMLOGS/Z1/debug
config.keystore.provider=IBMJCECCA
Admin.ssl.keystore.password=password
config.keystore.type=JCECCARACFKS
TransportListener.ssl.truststore.name=safkeyring\://TAPEKMS/KeyStore4758
Audit.handler.file.name=isklm_audit.log
TransportListener.ssl.port=1443
zOSCompatibility=true
TransportListener.tcp.port=3801
config.keygroup.xml.file=FILE\:/KLMETC/Z1/keygroup.xml
config.drivetable.file.url=FILE\:/KLMETC/Z1/filedrive.table
```

Sample of configuration file (cont'd)

```
TransportListener.ssl.keystore.name=safkeyring\://TAPEKMS/KeyStore4758
Audit.handler.file.directory=/KLMLOGS/Z1/logs
Audit.metadata.file.cachecount=0
Audit.eventQueue.max=0
Audit.event.outcome=success,failure
ds8k.acceptUnknownDrives=true
debug=none
TransportListener.ssl.ciphersuites=JSSE_ALL
config.keystore.file=safkeyring\://TAPEKMS/KeyStore4758
Audit.event.types=all
TransportListener.ssl.clientauthentication=0
TransportListener.ssl.host=localhost
TransportListener.ssl.protocols=SSL_TLS
```

Sample of configuration file (cont'd)

```
config.keystore.password=password
requireHardwareProtectionForSymmetricKeys=true
drive.default.alias2=Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21
drive.default.alias1=Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20
Audit.metadata.file.name=/KLMETC/Z1/metadata/SKLMData.xml
symmetricKeySet=des01-1f
debug.output=simple_file
drive.acceptUnknownDrives=true
Admin.ssl.truststore.name=safkeyring\://TAPEKMS/KeyStore4758
fips=Off
TransportListener.ssl.keystore.password=password
```

Copy the unrestricted policy files

These unrestricted policy files are required by the Security Key Lifecycle Manager for z/OS in order to serve AES keys.

```
142:/java/java16uk58682isklm/J6.0/demo/jce/policy-files $ cp unrestricted/* /java
/java16uk58682isklm/J6.0/lib/security
143:/java/java16uk58682isklm/J6.0/demo/jce/policy-files $
```

Added Hardware provider to Java

- Note: only needed if using ICSF

- /java/java16uk58682isklm/J6.0/lib/security/java
  - security.provider.1=com.ibm.jsse2.IBMJSSEProvider2
  - security.provider.2=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
  - security.provider.3=com.ibm.crypto.provider.IBMJCE
  - security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
  - security.provider.5=com.ibm.security.cert.IBMCertPath
  - security.provider.6=com.ibm.security.sasl.IBMSASL
  - security.provider.7=com.ibm.xml.crypto.IBMXMLCryptoProvider
  - security.provider.8=com.ibm.xml.enc.IBMXMLEncProvider

RACF updates

- Created started task entry
  - RDEFINE STARTED ISKLM*.* STDATA(USER(KLMSERV) GROUP(TASKS) TRACE(YES))

- Issued additional permit accesses for KLMSERV
  - TAPEKMS.KEYSTORE4758.LST in class RDATALIB
  - ICSF resources in class CSFSERV

© 2011 IBM Corporation

Created a PDS for the java config parms

- This PDS contains a member for each system
- This allows the use of a single started task for the sysplex yet allowing unique Java and config settings for each system

KLMSERV.ENCRYPT.CONFIG FB 80/6160 CYL(1,1)

Created member in java config dataset for JVM environment script

Example of contents of dataset “KLMSERV.ENCRYPT.CONFIG(Z1)

```
 # This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.
#Set these variables to the installation unique values
# ISKLM_HOME = directory where ISKLM runs from
# JAVA_HOME = directory where Java is mounted
export ISKLM_HOME="/ISKLM"
export JAVA_HOME=/java/java16uk58682isklm/J6.0
export PATH=/bin:"${JAVA_HOME}"/bin:
#LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin:"${JAVA_HOME}"/bin/j9vm
LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390
LIBPATH="$LIBPATH":"${JAVA_HOME}"/lib/s390/j9vm
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic
export LIBPATH="$LIBPATH":
```

Create member in config dataset for JVM environment script (cont'd)

```
 # Customize your CLASSPATH here

CLASSPATH="${JAVA_HOME}"/lib:"${JAVA_HOME}"/lib/ext
CLASSPATH=$CLASSPATH:/ISKLM/IBMSKLM.jar
export CLASSPATH="$CLASSPATH":
export ZZZZ="/KLMETC/Z1/ISKLMConfig.properties.zos"
export XXXX="com.ibm.ltklm.ISKLMServer"
export JZOS_MAIN_ARGS="$XXXX $ZZZZ"
# Configure JVM options (if any)
IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider"
export IBM_JAVA_OPTIONS="$IJO "
```

Update proclib from sample

```
//ISKLM PROC JAVACLS='com.ibm.jzosekm.ISKLMConsoleWrapper',
//   ARGS=,                          < ARGS TO JAVA CLASS
//   LIBRARY='JZOS.JAVA6031.LOADLIB',    < STEPLIB FOR JVMLDM MODULE
//   VERSION='60',                    < JVMLDM VERSION: 14, 50, 56
//   LOGLVL='+T',                     < DEBUG LVL: +I(INFO) +T(TRC)
//   REGSIZE='0M',                    < EXECUTION REGION SIZE
//   LEPARM=''
//*********************************************************************
//* STORED PROCEDURE FOR EXECUTING THE JZOS JAVA BATCH LAUNCHER
//*  SPECIFICALLY, TO EXECUTE THE ENTERPRISE KEY MANAGER UNDER JZOS
//*********************************************************************
//ISKLM  EXEC PGM=JVMLDM&VERSION,REGION=&REGSIZE,
//   PARM='&LEPARM/&LOGLVL &JAVACLS &ARGS'
//STEPLIB  DD DSN=&LIBRARY,DISP=SHR
//SYSPRINT DD SYSOUT=*          < SYSTEM STDOUT
//SYSOUT   DD SYSOUT=*          < SYSTEM STDERR
//STDOUT   DD SYSOUT=*          < JAVA SYSTEM.OUT
//STDERR   DD SYSOUT=*          < JAVA SYSTEM.ERR
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY
//*************************************************************
//*  THE FOLLOWING MEMBER CONTAINS THE JVM ENVIRONMENT SCRIPT
//*************************************************************
//STDENV DD DSN=KLMSERV.ENCRYPT.CONFIG(&SYSNAME.),DISP=SHR
//*
```

Creating the Keystore and Keys

Documentation to create Keystore and Keys is in infocenter Example 3: Using the JCERACFKS or JCECCARACFKS Keystore on z/OS

- Define Keyring:
```
RACDCERT ID(TAPEKMS) ADDRING(KeyStore4758)
```
- Issue appropriate access to the keyring function RACF profiles, if does not exist
```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(TAPEKMS) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(TAPEKMS) ACC(READ)
```
- Generate self-signed certificate authority certificate
```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('MyLocalCA')O('MyCo')C('US'))
WITHLABEL('MyLocalRACFCA') PCICC(LOCAL.RACF.CERTAUTH) SIZE(2048)
```
- Generated 2 Certificate/RSA Key Pairs for the Security Key Lifecycle Manager for z/OS server instance on z/OS
```
RACDCERT ID(TAPEKMS) GENCERT SUBJECTSDN(CN('ITOperations')
O('MyCo') C('US')) WITHLABEL('Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20') PCICC(ITOPS.KLM20.CERT)
SIZE(1024) SIGNWITH(CERTAUTH LABEL('MyLocalRACFCA'))

RACDCERT ID(TAPEKMS) GENCERT SUBJECTSDN(CN('ITOperations2')
O('MyCo') C('US')) WITHLABEL('Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21') PCICC(ITOPS.KLM21.CERT)
SIZE(1024) SIGNWITH(CERTAUTH LABEL('MyLocalRACFCA'))
```
- To refresh the RACF profiles, a SETROPTS RACLIST(DIGTCERT) REFRESH was issued.

Creating Keystore and Keys

- Ensure the Security Lifecycle Manager for z/OS Server certificates and its designated certificate authority are connected to the key ring of the Security Key Lifecycle Manager for z/OS

```
RACDCERT ID(TAPEKMS) CONNECT(CERTAUTH LABEL('MyLocalRACFCA') RING(KeyStore4758))

RACDCERT ID(TAPEKMS) CONNECT(LABEL(’Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20’)
    RING(KeyStore4758))

RACDCERT ID(TAPEKMS) CONNECT(LABEL(’Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21’)
    RING(KeyStore4758))
```

- Gave the Security Key Lifecycle Manager for z/OS Server instance RACF authority to the key label of the private keys stored in the ICSF PKDS

```
RDEFINE CSFKEYS ITOPS. KLM1.CERT UACC(NONE)
PERMIT ITOPS.KLM1.CERT CLASS(CSFKEYS) ID(KLMSERV) ACCESS(READ)

RDEFINE CSFKEYS ITOPS. KLM2.CERT UACC(NONE)
PERMIT ITOPS.KLM2.CERT CLASS(CSFKEYS) ID(KLMSERV) ACCESS(READ)
```

- To refresh the RACF profiles, a SETROPTS RACLIST(CSFKEYS) REFRESH was issued.

Setting up the IOS member

- Specify EKM primary and Secondary servers in SYS1.PARMLIB(IECIOSXX)
  - EKM PRIMARY=x.yy.zz.243:3801   (VIPA IP)
  - EKM SECONDARY=x.yy.zz.150:3801

- Dynamically activate the IP addresses with the following commands

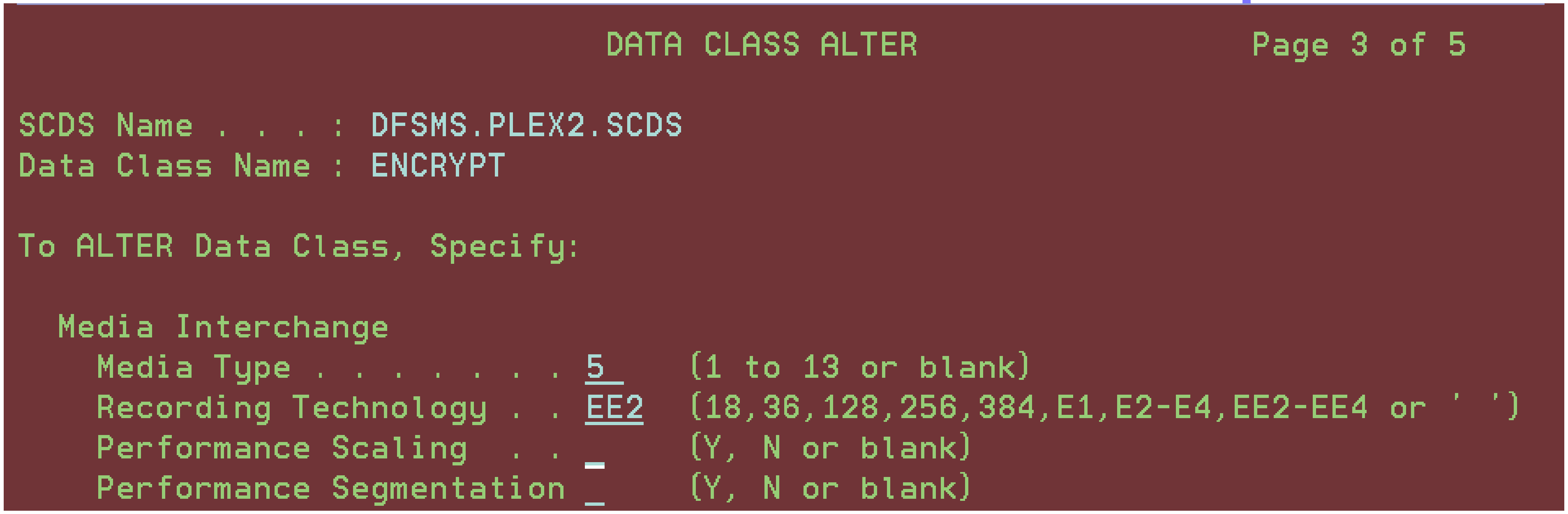
```
- SETIOS EKM[,PRIMARY= {ip_address[:port]}
                          OR
              [,SECONDARY= {ip_address[:port] }
```

© 2011 IBM Corporation

TCPIP updates

- Reserve the 2 ports for the ISKLM task
- Consider using VIPA (sysplex distributor) for IP/Port serving the keys
- For Key access from network outside of firewall, need to establish a ‘tunnel’ so the incoming IP is allowed access

DFSMS Data Class Setup

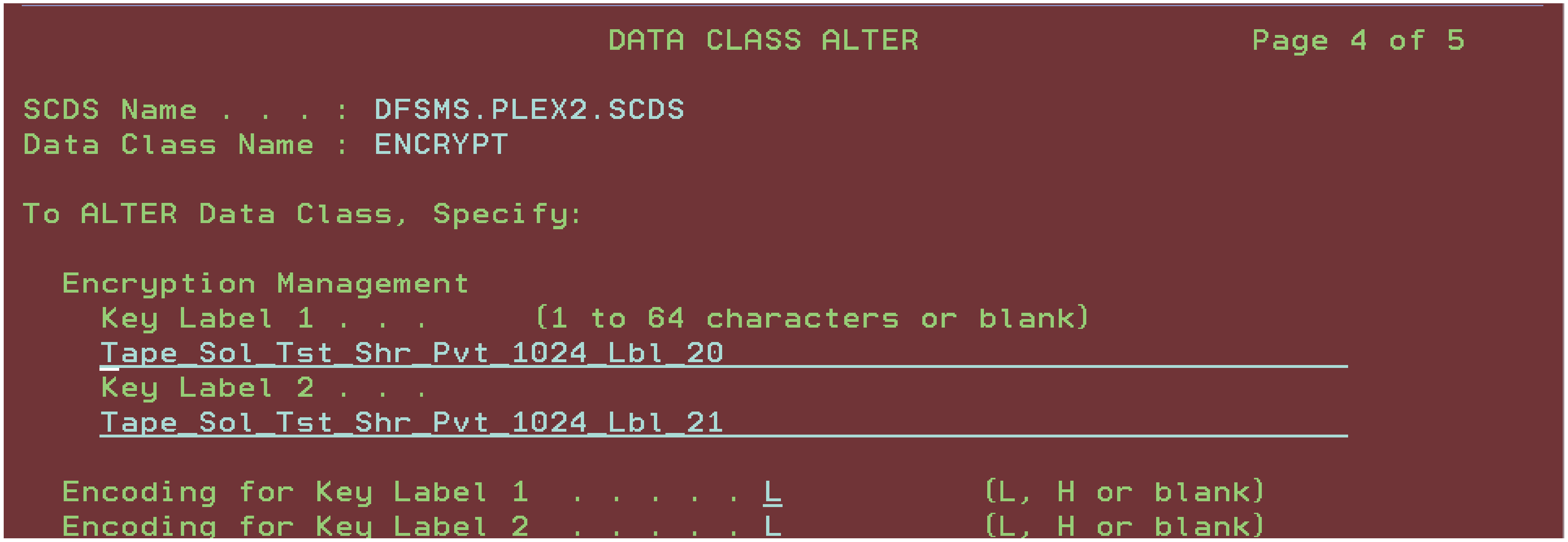


```
                         DATA CLASS ALTER                 Page 3 of 5

SCDS Name . . . : DFSMS.PLEX2.SCDS
Data Class Name : ENCRYPT

To ALTER Data Class, Specify:

  Media Interchange
    Media Type . . . . . . . 5     (1 to 13 or blank)
    Recording Technology . . EE2   (18,36,128,256,384,E1,E2-E4,EE2-EE4 or ' ')
    Performance Scaling  . . _     (Y, N or blank)
    Performance Segmentation _     (Y, N or blank)
```



```
                         DATA CLASS ALTER                 Page 4 of 5

SCDS Name . . . : DFSMS.PLEX2.SCDS
Data Class Name : ENCRYPT

To ALTER Data Class, Specify:

  Encryption Management
    Key Label 1 . . .      (1 to 64 characters or blank)
    Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20
    Key Label 2 . . .
    Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21

  Encoding for Key Label 1  . . . . . L          (L, H or blank)
  Encoding for Key Label 2  . . . . . L          (L, H or blank)
```

Setting up DFSMS Routines

- Data Class:

```
 FILTLIST TAPE_UNITS INCLUDE('3590','3490','3590-1','3590-H','3592-2E',
                            '3592-3E')
.
.
WHEN(&UNIT = '3592-2E')    SET &DATACLAS = 'ENCRYPT`
```

- Storage Class:

```
  FILTLIST TAPE_LIST INCLUDE('3590','3490','3590-1','3592-2E','3592-3E')
 .
 .
 WHEN(&UNIT = '3592-2E')    SET &STORCLAS = 'SCLSAT3'
```

Storage Group:

WHEN ('SCLSAT3')   SET &STORGRP = 'SGRPAT3'

Starting ISKLM

```
IEF403I ISKLM - STARTED - TIME=15.20.45
ISKLM wrapper V2.1 for ISKLM1.1
ISKLM console interaction is now available.
To submit commands to the ISKLM from the console:
   F ISKLM,APPL='ISKLM command'
To stop the ISKLM properly:
   P ISKLM
BPXM023I (KLMSERV) Loaded drive key store successfully
BPXM023I (KLMSERV) Starting the Security Key Lifecycle Manager 1.1-20110126
BPXM023I (KLMSERV) Processing Arguments
BPXM023I (KLMSERV) Contact IBM support at 1-800-IBM-SERV (1-800-426-7378) or through
      your normal business channel.
BPXM023I (KLMSERV) Processing
BPXM023I (KLMSERV) Server is started
BPXM023I (KLMSERV) Server is running. TCP port: 3801, SSL port: 1443
```

© 2011 IBM Corporation

Running encryption job

```
/*JOBPARM SYSAFF=Z1
//*
//STEP2    EXEC PGM=IEBGENER
//SYSUT1   DD DSN=FLATES.ENCRYPT.TEST,DISP=SHR,
//            UNIT=3390
//SYSUT2   DD UNIT=3592-2E,
//        DSN=PET.ENCY.PLX.TEST,LABEL=(1,SL),
//        DATACLAS=ENCRYPT,DISP=OLD,VOL=SER=WXY662
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
```

Output from running encryption job

```
IEF403I ENCRYPT - STARTED - TIME=15.34.35
IEE302I B5E1     ONLINE BY GETTAPE
IEF233A M B5E1,WXY662,,ENCRYPT,STEP2
IEC705I TAPE ON B5E1,WXY662,SL,COMP,ENCRYPT,STEP2,MEDIA5
IEC205I SYSUT2,ENCRYPT,STEP2,FILESEQ=1, COMPLETE VOLUME LIST,  131
VOLS=WXY662,LISTED VOL(S) HAVE BEEN DATA
    ENCRYPTED,KL1CD:L,KL2CD:L,
KL1=Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20,
KL2=Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21,TOTALBLOCKS=2
IEF234E K B5E1,WXY662,PVT,ENCRYPT,STEP2
```

Audit Records

Audit records are stored in SMF (type83). There also is the capability to have file-based recording.

Example of a file-based audit record showing successful encrypt.

```
Runtime event:Ý
  timestamp=Tue Oct 18 19:35:09 GMT 2011
  ComponentId=ÝthreadId=ThreadÝThread-24,5,KeyManagementServerV2-Processors¨¨
  event source=com.ibm.ltklm.logic.fb
  outcome=Ýresult=successful¨
  event type=SECURITY_RUNTIME
  resource=Ýname=Write Request: Drive Serial Number: 000007888422 WWN:500507630240FBCC
      VolSer: WXY662 Key Alias/LabelÝ0¨: Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20 Key
      Alias/LabelÝ1¨: Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21;type=file¨
  action=stop
```

Process for establishing backup key server

- From the primary sysplex, export the certificates from RACF and transmit to the target sysplex.
  - Move the certificate to a file:

```
RACDCERT ID(TAPEKMS) EXPORT (LABEL('Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20'))
    DSN('TAPEKMS.LBL20.PUBKEY') FORMAT(CERTDER)
RACDCERT ID(TAPEKMS) EXPORT (LABEL('Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21'))
    DSN('TAPEKMS.LBL21.PUBKEY') FORMAT(CERTDER)
RACDCERT CERTAUTH EXPORT (LABEL('MyLocalRACFCA')) DSN('TAPEKMS.CA.CERT')
    FORMAT(CERTDER)
```

  - Transmit files to backup sysplex
- Copy private key tokens to a dataset
  - Using the KEYXFER tool from the z/OS UNIX tools Web site, www.ibm.com/servers/eserver/zseries/zos/unix/bpxa1ty2.html , we used the WRITE operation to write the private key token stored in the PKDS to a data set.

  Note: Before using this tool, read the readme file to ensure that you can use this tool in your environment. Especially note that the ICSF Master Keys must be the same on both sysplexes.

```
KEYXFER WRITE,-
ITOPS.KLM20.CERT,-
CRYPTOR2.CCATEST.PARMS(KLM20)
KEYXFER WRITE,-
ITOPS.KLM21.CERT,-
CRYPTOR2.CCATEST.PARMS(KLM21)
```

  - Transmit files to backup sysplex

Process for establishing backup key server

- From the backup sysplex, receive the private key token datasets and the RACF certificate datasets
- Write the private key into the PKDS on the backup sysplex using the KEYXFER job with the READ operation

```
KEYXFER READ,-
ITOPS.KLM20.CERT,-
CRYPTOR2.CCATEST.PARMS(KLM20),-
OVERWRITE
KEYXFER READ,-
ITOPS.KLM21.CERT,-
CRYPTOR2.CCATEST.PARMS(KLM21),-
OVERWRITE
```

- Import the RACF certificate information on the backup sysplex and associate it with the ICSF data where applicable

```
RACDCERT CERTAUTH ADD('TAPEKMS.CA.CERT') WITHLABEL('MyLocalRACFCA')
RACDCERT ID(TAPEKMS) ADD('TAPEKMS.LBL20.PUBKEY') TRUST
WITHLABEL('Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20')
PCICC(ITOPS.KLM20.CERT)
RACDCERT ID(TAPEKMS) ADD('TAPEKMS.LBL21.PUBKEY') TRUST
WITHLABEL('Tape_Sol_Tst_Shr_Pvt_2048_Lbl_21')
PCICC(ITOPS.KLM21.CERT)
```

- To refresh the RACF database, a SETROPTS RACLIST(DIGTCERT) REFRESH was issued.

Process for establishing backup key server

- Ensure the Security Lifecycle Manager for z/OS Server certificates and its designated certificate authority are connected to the key ring of the Security Key Lifecycle Manager for z/OS

```
RACDCERT ID(TAPEKMS) CONNECT(CERTAUTH LABEL('MyLocalRACFCA') RING(KeyStore4758))

RACDCERT ID(TAPEKMS) CONNECT(LABEL(’Tape_Sol_Tst_Shr_Pvt_1024_Lbl_20’)
    RING(KeyStore4758))

RACDCERT ID(TAPEKMS) CONNECT(LABEL(’Tape_Sol_Tst_Shr_Pvt_1024_Lbl_21’)
    RING(KeyStore4758))
```

- Gave the Security Key Lifecycle Manager for z/OS Server instance RACF authority to the key label of the private keys stored in the ICSF PKDS

```
RDEFINE CSFKEYS ITOPS. KLM20.CERT UACC(NONE)
PERMIT ITOPS.KLM1.CERT CLASS(CSFKEYS) ID(KLMSERV) ACCESS(READ)

RDEFINE CSFKEYS ITOPS. KLM21.CERT UACC(NONE)
PERMIT ITOPS.KLM2.CERT CLASS(CSFKEYS) ID(KLMSERV) ACCESS(READ)
```

- To refresh the RACF profiles, a SETROPTS RACLIST(CSFKEYS) REFRESH was issued.

Documentation

- http://www-01.ibm.com/software/tivoli/products/security-key-lifecycle-mgr-z/
- http://www.redbooks.ibm.com/redpapers/abstracts/redp4646.html?Open

Comments?

Backup

© 2011 IBM Corporation

Keys

- The Security Key Lifecycle Manager for z/OS requires the definition of at least two aliases (certificates or key labels) for each encrypting tape drive
- This requirement enables access to the encrypted data at another location, whether within your organization or outside it
- The private key for one of these aliases must be known

Disk Encryption Process

- DS8000 receives read request and sends the EEDK to Security Key Lifecycle Manager for z/OS
- Security Key Lifecycle Manager for z/OS verifies DS8000 is in the Device Table
- Security Key Lifecycle Manager for z/OS fetches certificate required to process the EEDK from keystore
- Security Key Lifecycle Manager for z/OS unwraps EEDK using private key of KEK pair to recover unlock key
- Security Key Lifecycle Manager for z/OS wraps unlock key with a key the drive can decrypt and sends the wrapped unlock key to DS8000
- DS8000 unwraps the unlock key and uses it to decrypt the data or to perform a write-append

Tape Encryption Process

- Tape drive requests key to encrypt tape (via IP/PORT assignment in IOS)
- Security Key Lifecycle Manager for z/OS verifies tape device in Device Table
- Security Key Lifecycle Manager for z/OS fetches keys and certificates for tape device from keystore
- Security Key Lifecycle Manager for z/OS generates a random DK
- Security Key Lifecycle Manager for z/OS wraps DK with public key to create an EEDK
- Security Key Lifecycle Manager for z/OS sends the EEDK and (separately wrapped) DK to the tape drive
- Tape drive unwraps the DK and writes the EEDK on tape leader
- Tape drive encrypts data using DK and writes encrypted data to tape