IBM Encryption Key Manager component for the Java platform

Introduction, Planning, and User's Guide

GA76-0418-08

IBM Encryption Key Manager component for the Java platform

Introduction, Planning, and User's Guide

GA76-0418-08

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 221.

Ninth edition

This edition applies to the IBM Encryption Key Manager component for the Java platform and to all subsequent releases and modifications until otherwise indicated in new editions. This edition replaces GA76-0418-07.

Do You Have Comments or Suggestions?

Send us your comments by e-mail to starpubs@us.ibm.com or use the Readers' Comments form at the back of this publication. Be sure to include the following:

- Exact publication title
- Form number (for example, GA76-1234-05) , or part number and EC level (located on the back cover)
- Page numbers to which you are referring

When you send information to IBM®, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this Publication

This book contains information to help you install, configure, and use IBM Encryption Key Manager component for the Java™ platform. It includes concepts and procedures pertaining to:

- Encryption on the IBM System Storage TS1120 and TS1130 tape drives, and on IBM LTO Ultrium 4 Tape Drives.
- Cryptographic keys
- Digital certificates

Who Should Read this Publication

This book is intended for storage and security administrators responsible for security and backup of vital data, and anyone assisting in the setup and maintenance of Encryption Key Manager servers in the operating environment. It assumes the reader has a working knowledge of storage devices and networks.

Conventions and Terminology Used in this Publication

This book uses the following typographic conventions:

Table 1. Conventions and Terminology Used in this Publication

Convention	Usage
bold	**Bold** words or characters represent system elements that you must use literally, such as command names, file names, flag names, path names, and selected menu options.
`constant width`	Examples, text specified by the user, and information that the system displays appear in `constant width` typeface.
italic	*italicized* words or characters represent variable values that you must supply.
[item]	Indicates optional items.
{item}	Encloses a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices.
<Key>	Indicates keys you press.

Attention Notice

An attention notice indicates the possibility of damage to a program, device, system, or to data. An exclamation point symbol may accompany an attention notice, but is not required. Sample attention notices follow:

Attention: If you use a power screwdriver to perform this procedure it could destroy the tape.

Attention: Do not operate the 3592 in a poor air-quality environment.

Related Publications

The following publications provide information related to encryption on tape drives:

IBM Encryption Key Manager Publications

- *IBM Encryption Key Manager component for the Java platform Quick Start Guide for LTO Ultrium 4*, GA76-0420
- *IBM Encryption Key Manager component for the Java platform Quick Start Guide for TS1120 and TS1130 Tape Drives*, GA76-0421

IBM System Storage TS1120 Tape Drive and Controller Publications

- *IBM System Storage TS1120 Tape Drive and Controller Operator Guide*, GA32-0556
- *IBM System Storage TS1120 Tape Drive SCSI Reference*, GA32-0562
- *IBM TotalStorage Enterprise Silo Compatible Tape Frame 3592 Introduction, Planning, and User's Guide*, GA32-0463

IBM LTO Ultrium 4 Tape Drive Publications

- *IBM LTO Ultrium 4 Tape Drive Setup, Operator, and Service Guide*, GA27-2102

IBM System Storage TS3500 Tape Library Publications

- *IBM System Storage TS3500 Tape Library Operator Guide*, GA32-0560
- *IBM System Storage TS3500 Tape Library Introduction and Planning Guide*, GA32-0559

IBM Virtualization Engine TS7700 Publications

- *IBM Virtualization Engine TS7700 Series Introduction and Planning Guide*, GA32-0567

IBM 3953 Tape System Publications

- *IBM 3953 Library Manager Model L05 Operator Guide*, GA32-0558
- *IBM 3953 Tape System Introduction and Planning Guide*, GA32-0557

IBM TotalStorage Enterprise Automated Tape Library (3494) Publications

- *IBM TotalStorage Automated Tape Library (3494) Introduction and Planning Guide*, GA32-0448
- *IBM TotalStorage Automated Tape Library (3494) Operator's Guide*, GA32-0449

zSeries—S390 Publications

- *IBM eServer™ zSeries 900 Platform Reference Guide*, G326-3092
- *Introduction to IBM S/390 FICON*, SG24-5176 (IBM Redbook).
- *S/390 System Overview Parallel Enterprise Server — Generation 5*, GA22-7158

- *S/390 System Overview Parallel Enterprise Server — Generation 6*, GA22-1030

IBM Fibre Channel Publications

- *IBM TotalStorage SAN Switch 2109 Model F16 Installation and Service Guide,* SY27-7623
- *IBM Fiber-Optic Channel Link Planning and Installation*, GA32-0367

IBM FICON Publications

- *FICON (FCV Mode) Planning Guide*, SG24-5445-00 (IBM Redbook).
- *Planning for: Fiber Optic Links (ESCON, FICON, Coupling Links, and Open system Adapters)*, GA23-0367
- *Maintenance Information for: Fiber Optic Links (ESCON, FICON, Coupling Links, and Open System Adapters)*, SY27-2597
- *Fiber Channel Connection (FICON) I/O Interface Physical Layer*, SA24-7172
- *Introduction to IBM System/390 FICON*, SG24-5176
- *Planning for the ED-5000 Enterprise Fibre Channel Director*
- *IBM eServer zSeries Connectivity Handbook*, SG24-5444
- *IBM Tape Solutions for Storage Area Networks and FICON*, SG24-5474

Related Software Publications

For information regarding software related to the IBM 3592 Tape System, refer to:

- *IBM Tape Device Drivers Installation and User's Guide*, GC35-0154
- *Basic Tape Library Support User's Guide and Reference*, SC26-7016
- *Environmental Record Editing and Printing (EREP) Program User's Guide and Reference*, GC35-0151
- *IBM Tivoli Storage Manager for AIX Administrator's Guide*, GC32-0768
- *z/OS DFSMS Introduction*, SC26-7397
- *z/OS DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries*, SC35-0427
- | *z/OS DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape*
  | *Drives (3592)*, SC26-7514
- *z/OS Migration*, GA22-7499.
- *z/OS Security Server RACF Security Administrator's Guide*, SA22-7683
- *z/OS Security Server RACF Command Language Reference*, SA22-7687
- *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide*, SA22-7520
- *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide*, SA22-7521
- *z/OS Cryptographic Services Integrated Cryptographic Service Facility Application Programmer's Guide*, SA22-7522
- *z/OS Cryptographic Services System Secure Sockets Layer Programming* , SC24-5901
- *z/VM General Information Version 4 Release 3.0*, GC24-5991
- *z/VM CP Planning and Administration*, SC24-6083
- *z/VM CP Commands and Utilities*, SC24-6081
- *IBM eServer™ zSeries® 900 Platform Reference Guide*, G326-3092
- *Introduction to IBM S/390® FICON®*, SG24-5176 (IBM Redbook) G326-3092
- *S/390 System Overview Parallel Enterprise Server — Generation 5*, GA22-7158

- *S/390 System Overview Parallel Enterprise Server — Generation 6*, GA22-1030

Other Publications

- *American National Standard Institute Small Computer System Interface* X3T9.2/86-109 X3.180, X3B5/91-173C, X3B5/91-305, X3.131-199X Revision 10H, and X3T9.9/91-11 Revision 1

IBM Online Access

Encryption Key Manager Support

For the latest version of Encryption Key Manager, visit the following URL:

- **http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504**

IBM Java Security Components and Keystores

- Overview technical article: **http://www.ibm.com/developerworks/library/j-ibmsecurity.html**
- SDK 1.4.2: **http://www.com/developerworks/java/jdk/security/142/**
- SDK 5.0 : **http://www.ibm.com/developerworks/java/jdk/security/50/**
- hwkeytool: **http://www.ibm.com/servers/eserver/zseries/software/java/hwkeytool.html**
- ikeyman: For ikeyman and the keytool user guide: see SDK 1.4.2 or SDK 5.0 links above
- For additional information specific to z/OS: **http://www.ibm.com/servers/eserver/zseries/software/java/**

IBM pSeries—RS/6000, AIX Information

For additional information about IBM eServer pSeries® servers, visit the info center at:

- **http://publib16.boulder.ibm.com/pseries/en_US/infocenter/base/**

IBM Storage Media Support

The following URL provides access to current regional and country-specific IBM addresses and telephone numbers.

- **http://www.storage.ibm.com/media/distributors**

IBM TotalStorage Enterprise Tape System 3592 Support

For general information about the 3592 Tape System, visit the following URL:

- **http://www.ibm.com/servers/storage/tape/3592/index.html**

For general information about the TS1120 Tape Drive, visit the following URL:

- **http://www.ibm.com/servers/storage/tape/ts1120/index.html**

For information about supported servers for the 3592 Tape System and TS1120 tape Drive, visit the following URL:

- **http://www.ibm.com/servers/storage/tape/compatibility/pdf/3592_interop.pdf**

The following URLs provide access to additional current information related to 3592 Tape System.

Device Driver Support

To access the 3592 Firmware and Device Driver Matrix, visit the following URL:

- **http://www.ibm.com/servers/storage/support/tape/3592/downloading.html**

To access the TS1120 Firmware and Device Driver Matrix, visit the following URL:

- **http://www.ibm.com/servers/storage/support/tape/ts1120/downloading.html**

You can download device driver software and read documentation about various device drivers at the following URL:

- **ftp://ftp.software.ibm.com/storage/devdrvr/**

IBM Virtualization Engine TS7700 Encryption Support

White paper: *IBM Virtualization Engine TS7700 Series Encryption Overview* available at

- **http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504**

IBM Network Integration and Deployment Services

The following URL provides information about connectivity and the integration of cabling systems.

- **http://www.ibm.com/services/networking/integration**

IBM Tape Storage Publications

Use this URL for IBM Hardware product documents in a PDF format for viewing and printing.

- **http://www.ibm.com/servers/storage/tape/resource-library.html#publications**

SAN Fabric

This link provides information on high-performance switches and gateways.

- **http://www.storage.ibm.com/ibmsan/products/sanfabric.html**

I/O Connectivity

This link provides updated information regarding FICON® and fibre channel connectivity.

- **http://www.ibm.com/servers/eserver/zseries/connectivity**

Redbooks

Use this URL to access the IBM Redbooks™:

- **http://www.redbooks.ibm.com/**

Vendor Support

These URLs provide compatibility information in PDF format for implementing software, servers, and operating systems with IBM tape drives and libraries.

- For TS1120: **http://www.ibm.com/servers/storage/tape/compatibility/pdf/ts1120_isv_matrix.pdf**
- For 3592: **http://www.ibm.com/servers/storage/tape/compatibility/pdf/3592_isv_matrix.pdf**

Non-IBM Support

HP Information

The following publications and URL relate to HP-UX systems:

- *HP-UX Reference for HP-UX 10.20, 11.00, and 11i*, Hewlett-Packard Company

- *System Administration Tasks, HP-UX Release 10.20, 11.00, and 11i*, Hewlett-Packard Company
- Additional HP information can be found at this URL: **http://docs.hp.com**

Linux Information

Red Hat Information

The following URL relates to Red Hat Linux® systems:

- **http://www.redhat.com**

SuSE Information

The following URL relates to SuSE Linux systems:

- **http://www.suse.com**

Microsoft Windows Information

The following URL provides access to information about Microsoft® Windows® systems:

- **http://www.microsoft.com**

SGI Information

The following URL provides access to information about SGI systems:

- **http://techpubs.sgi.com/library**

SUN Information

The following URL provides access to information about Solaris (Sun) systems:

- **http://docs.sun.com/**

Summary of Changes

Ninth Edition

In the ninth edition, information has been enhanced to address the following:

- The introduction of the new IBM System Storage TS1130 Tape Drive (3592 Models E06 and EU6). See Chapter 1, "Tape Encryption Overview" and Chapter 2, "Hardware and Software Requirements."
- New information was added in chapter 4, "Upgrading to the latest Encryption Key Manager or IBM Java SDK" to provide assistance when downloading newer versions.
- Keystore passwords are now restricted to a maximum of 127 characters. See "Managing Keystores" in Chapter 2.
- Miscellaneous editorial changes.

Eighth Edition

In the eighth edition, information has been enhanced to address the following:

- The recommended use of Error Correction Code (ECC) memory in the server running the Encryption Key Manager. See "Managing Encryption" in Chapter 1.
- The replacement of IBM TotalStorage Productivity Center/Limited Edition (TPC/LE), a prerequisite for the Encryption Key Manager running on HP, Sun, and Windows platforms, by IBM TotalStorage Productivity Center/Basic Edition (TPC/BE). See "Hardware and Software Requirements" in Chapter 2.
- The clarification of IBM Support contacts based on the platform on which the Encryption Key Manager runs, and the nature of the problem. See "Whom Do I Contact for IBM Support" in Chapter 6.

Seventh Edition

In the seventh edition, information has been enhanced to address the following:

- Explanation why changing keystore passwords should be avoided unless necessary, and description of what to do if it is necessary. See "Changing Keystore Passwords" in Chapter 3.
- New procedure for keygroup creation. See "Using CLI Commands to Define Key Groups" in Chapter 3.
- New procedure: "Configuring the Encryption Key Manager to Use Two Keystores." See Chapter 4.
- New problem determination aids. See Chapter 6.
- Miscellaneous editorial changes.

Sixth Edition

In the sixth edition, information has been enhanced to address the following:

- Support for the organization of symmetric encryption keys into key groups. See "Creating key groups" in Chapter 3.

- Support for the Encryption Key Manager to run in the background as a daemon or Windows Service, controlled and monitored from a remote command line interface client. See "Starting and stopping the key manager server" in Chapter 5.
- The ability to specify a default symmetric key for each encrypting tape drive. See the **adddrive** and **moddrive** command descriptions in Chapter 5.
- The obfuscation of keystore passwords when stored in the configuration properties file. See the password property descriptions in Appendix B.
- Miscellaneous editorial changes.

Fifth Edition

In the fifth edition, information has been enhanced to address the following:

- Support for encryption on IBM LTO Ultrium 4 Tape Drives and LTO 4 tape cartridges. See Chapter 1, "Tape Encryption Overview."
- Support for the T10 encryption command set defined by the InterNational Committee for Information Technology Standards (INCITS). See "Encryption key processing by other applications" in Chapter 1.
- A new procedure for upgrading the Software Developer Kit on z/OS when using Crypto. See "Migrating from SDK 1.4.2 to SDK 5.0 on z/OS with the IBM Hardware Crypto Provider" in Chapter 4.
- New **list** and **refreshks** commands. See Chapter 5.
- A new metadata XML file and query function. See Chapter 8, "Using Metadata".

Fourth Edition

In the fourth edition, information has been enhanced to address the following:

- The definition of aliases to allow access to encrypted data. See "The importance of keys and certificates" in Chapter 2.
- Miscellaneous editorial changes.

Third Edition

In the third edition, information has been enhanced to address the following:

- The procedures for installing the Encryption Key Manager and keystores on z/OS® and on Unix-based operating systems have been reorganized and updated. See Chapter 2.
- Miscellaneous additions and corrections.

Second Edition

In the second edition, information has been enhanced to address the following:

- Encryption key flow in system-managed encryption on System z™ platforms. Refer to "System-managed Tape Encryption" in Chapter 1.
- Service tasks required for Encryption setup. Refer to "TS1120 Tape Drive Installation Process for Encryption" in Chapter 2.
- Encryption keystore creation on z/OS. Refer to "Installing on z/OS" in Chapter 3.
- Synchronizing Encryption Key Manager servers. Refer to "Synchronizing data between two Encryption Key Manager servers" in Chapter 4.
- Miscellaneous additions and corrections.

Chapter 1. Tape Encryption Overview

Data is one of the most highly valued resources in a competitive business environment. Protecting that data, controlling access to it, and verifying its authenticity while maintaining its availability are priorities in our security-conscious world. Data encryption is a tool that answers many of these needs. The IBM Encryption Key Manager component for the Java platform (referred to as the Encryption Key Manager from this point forward) simplifies encryption tasks.

The IBM System Storage™ TS1130 Tape Drive (3592 Model E06 and Model EU6[1]) and TS1120 Tape Drive (3592 Model E05) are capable of encrypting data as it is written to any size IBM TotalStorage® Enterprise Tape Cartridge 3592, including WORM cartridges. The IBM LTO Ultrium 4 Tape Drive is also capable of encrypting data as it is written to any LTO Ultrium 4 Data Cartridge. Encryption is performed at full line speed in the tape drive after compression. (Compression is more efficiently done before encryption.) This new capability adds a strong measure of security to stored data without the processing overhead and performance degradation associated with encryption performed on the server or the expense of a dedicated appliance.

The tape drive encryption solution comprises three major elements:

The Encryption-Enabled Tape Drive

All TS1130 Tape Drives are encryption-capable. All TS1120 Tape Drives with Feature Code 5592 or 9592 are *encryption-capable*. All IBM LTO Ultrium 4 Tape Drives are *encryption-capable*[2]. This means that they are functionally capable of performing hardware encryption, but this capability has not yet been activated. In order to perform hardware encryption, the tape drives must be *encryption-enabled*.

In an IBM System Storage TS3500 Tape Library, the TS1120 and TS1130 tape drives can be encryption-enabled through the IBM System Storage Tape Specialist.

Note: When a TS1130 Tape Drive is attached to a 3592 J70 or C06 tape controller, the tape drive must be enabled for system-managed encryption. This applies even when encryption is not being used by the host.

For all other TS1120 and TS1130 tape drives this process consists of having an IBM representative set up the drive for encryption. Only encryption-enabled TS1120 and TS1130 tape drives can be used to read and write encrypted 3592 tape cartridges.

All IBM LTO Ultrium 4 Tape Drives can be encryption-enabled through the IBM System Storage Tape Specialist. However, encryption must be licensed on LTO Ultrium 4 Tape Drives in tape libraries. This is acquired with Feature Code 1604 on the TS3500 Library or Feature Code 5900 on other libraries. Refer to your library documentation for more information.

1. The 3592 EU6 Tape Drive is a 3592 E05 Tape Drive canister upgraded to contain a Model E06 drive through the MES (Miscellaneous Equipment Specification) process.
2. Fibre-Channel (FC) and Serial Attached SCSI (SAS) IBM LTO-4 tape drives are encryption-capable. SCSI IBM LTO-4 tape drives are encryption aware, can load and handle encrypted LTO-4 cartridges, but cannot process encryption operations.

See "Hardware and Software Requirements" on page 18 for more information on tape drives.

Encryption Key Management
Encryption involves the use of several kinds of keys, in successive layers. The generation, maintenance, control, and transmission of these keys depends upon the operating environment where the encrypting tape drive is installed. Some applications such as Tivoli Storage Manager, are capable of performing key management. For environments without such applications or those where application agnostic encryption is desired, IBM provides the Encryption Key Manager to perform all necessary key management tasks. "Managing Encryption" on page 3 describes these tasks in more detail.

Encryption Policy
This is the method used to implement encryption. It includes the rules that govern which volumes are encrypted and the mechanism for key selection. How and where these rules are set up depends on the operating environment. See "Managing Encryption" on page 3 for more information.

Note: In the Tape Storage environment, the Encryption function on tape drives (desktop, stand-alone and within libraries) is configured and managed by the customer and not the IBM System Services Representative (SSR). In some instances SSRs will be required to enable encryption at a hardware level when service access or service password controlled access is required. Customer setup support is by Field Technical Sales Specialist (FTSS), customer documentation, and software support for encryption software problems. Customer "how to" support is also provided via support line contract.

Encryption Key Manager Components

The Encryption Key Manager is part of the IBM Java environment and uses the IBM Java Security components for its cryptographic capabilities. (For more information on the IBM Java Security components please see the related publications section.) The Encryption Key Manager has three main components that are used to control its behavior. These components are:

Java security keystore
The keystore is defined as part of the Java Cryptography Extension (JCE) and an element of the Java Security components, which are, in turn, part of the Java runtime environment. A keystore holds the certificates and keys (or pointers to the certificates and keys) used by the Encryption Key Manager to perform cryptographic operations. Several types of Java keystores are supported offering different operational characteristics to meet your needs. These characteristics are discussed in detail in "Keystore Considerations" on page 29.

⚠ It is impossible to overstate the importance of preserving your keystore data. Without access to your keystore you will be unable to decrypt your encrypted tapes. Please carefully read the topics below to understand the methods available for protecting your keystore data.

Configuration files
The configuration files allow you to customize the behavior of the Encryption Key Manager to meet the needs of your organization. These behavioral choices are described extensively in this document, first in Chapter 2, "Planning Your Encryption Key Manager Environment," on page 13

page 13, then in the Chapter 5, "Configuring the Encryption Key Manager," on page 135, and later in Appendix B where the full set of configuration options is described.

Tape drive table
The tape drive table is used by the Encryption Key Manager to keep track of the tape devices it supports. The tape drive table is a non-editable, binary file whose location is specified in the configuration file. You can change its location to meet your needs.

KeyGroups.xml file
This password-protected file contains the names of all encryption key groups and the aliases of the encryption keys associated with each key group.

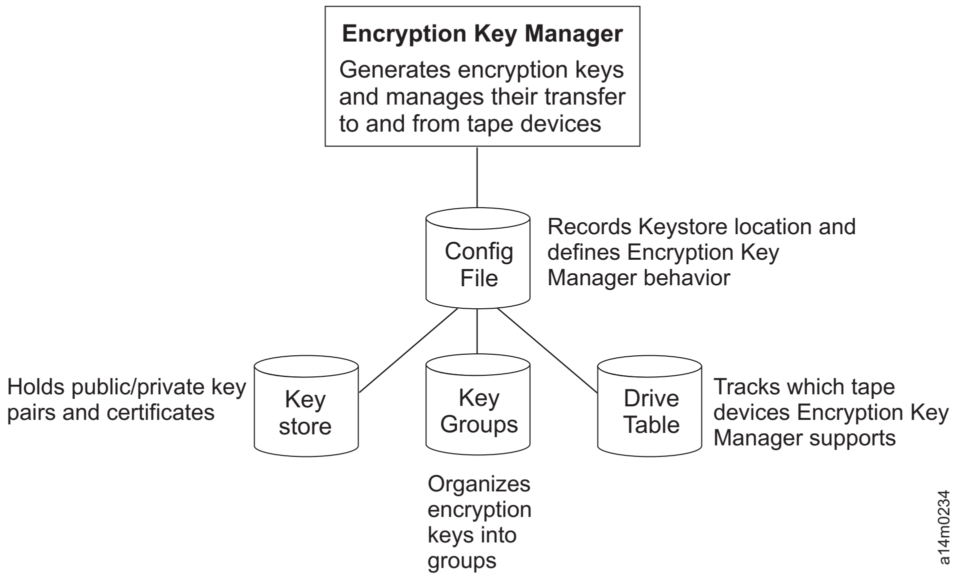

Encryption Key Manager
Generates encryption keys and manages their transfer to and from tape devices
Config File
Records Keystore location and defines Encryption Key Manager behavior
Holds public/private key pairs and certificates
Key store
Key Groups
Drive Table
Tracks which tape devices Encryption Key Manager supports
Organizes encryption keys into groups
a14m0234

Figure 1. The Encryption Key Manager's four main components

Managing Encryption

The IBM Encryption Key Manager is a Java software program that assists IBM encryption-enabled tape drives in generating, protecting, storing, and maintaining encryption keys that are used to encrypt information being written to, and decrypt information being read from, tape media (tape and cartridge formats). The Encryption Key Manager operates on z/OS, i5/OS® (IBM i Operating System), AIX®, Linux, HP-UX, Sun Solaris, and Windows, and is designed to run in the background as a shared resource deployed in several locations within an enterprise. The Encryption Key Manager is capable of serving numerous IBM encrypting tape drives, regardless of where those drives reside (for example, in tape library subsystems, connected to mainframe systems through various types of channel connections, or installed in other computing systems.) A command line interface client provides a robust set of commands to customize the Encryption Key Manager for your environment and monitor its operation. The Encryption Key Manager uses one or more keystores to hold the certificates and keys (or pointers to the certificates and keys) required for all encryption tasks. The Encryption Key Manager supports the following IBM keystores: JCE4758KS/JCECCAKS, JCE4785RACFKS/JCECCARACFKS, JCERACFKS, IBMi5OSKeyStore, and PKCS11IMPLKS. See "Keystore Considerations" on page 29 for detailed information.

The Encryption Key Manager performs the function of requesting the generation of encryption keys and passing those keys to the TS1120, TS1130, or LTO-4 tape drives. The key material, in wrapped (encrypted) form does reside in system memory during processing by the Encryption Key Manager. Note that the key material (in either wrapped or clear form) must be transferred without error to the appropriate tape drive so that data written on a cartridge may be recovered (decrypted). If for some reason the key material (in either wrapped or clear form) is corrupted due to a bit error in system memory, and that key material is used to write data to a cartridge, then the data written to that cartridge will be unrecoverable (not able to be decrypted). There are safeguards in place to make sure that such data errors do not occur. However, if the machine hosting the Encryption Key Manager is not using Error Correction Code (ECC) memory there remains a possibility that the key material may become corrupted while in system memory and the corruption could then cause data loss. Although the risk is slight, it is always recommended that machines hosting the Encryption Key Manager use ECC memory.

The Encryption Key Manager acts as a background process awaiting key generation or key retrieval requests sent to it through a TCP/IP communication path between itself and the tape library, tape controller, tape subsystem, device driver, or tape drive. When a tape drive writes encrypted data, it first requests an encryption key from the Encryption Key Manager. Upon receipt of the request, the Encryption Key Manager performs the following tasks.

For TS1120 and TS1130 tape drives: The Encryption Key Manager generates an Advanced Encryption Standard (AES) key and serves it to the tape drives in two protected forms:

- Encrypted or *wrapped*, using Rivest-Shamir-Adleman (RSA) key pairs. TS1120 and TS1130 tape drives write this copy of the key to the cartridge memory and three additional places on the tape media in the cartridge for redundancy.
- Separately wrapped for secure transfer to the tape drive where it is unwrapped upon arrival and the key inside is used to encrypt the data being written to tape.

When an encrypted tape cartridge is read by a TS1120 or TS1130 Tape Drive, the protected AES key on the tape is sent to the Encryption Key Manager where the wrapped AES key is unwrapped. The AES key is then wrapped with a different key for secure transfer back to the tape drive, where it is unwrapped and used to decrypt the data stored on the tape. The Encryption Key Manager also allows protected AES keys to be rewrapped, or *rekeyed*, using different RSA keys from the originals used when the tape was written. Rekeying is useful when an unexpected need arises to export volumes to business partners whose public keys were not included; it eliminates the need to rewrite the entire tape and enables a tape cartridge’s data key to be reencrypted with a business partner’s public key.

For LTO Ultrium 4 Tape Drives: The Encryption Key Manager fetches an existing AES key from a keystore and wraps it for secure transfer to the tape drive where it is unwrapped upon arrival and used to encrypt the data being written to tape.

When an encrypted tape is read by an LTO Ultrium 4 Tape Drive, the Encryption Key Manager fetches the required key from the keystore, based on the information in the Key ID on the tape, and serves it to the tape drive wrapped for secure transfer.

For TS1120, TS1130, and LTO Ultrium 4 tape drives: There are three methods of encryption management to choose from. These methods differ in where the encryption policy engine resides, where key management is performed for your solution, as well as how the Encryption Key Manager is connected to the drive. Your operating environment determines which is the best for you. Key management and the encryption policy engine may be located in any one of the following three environmental layers.

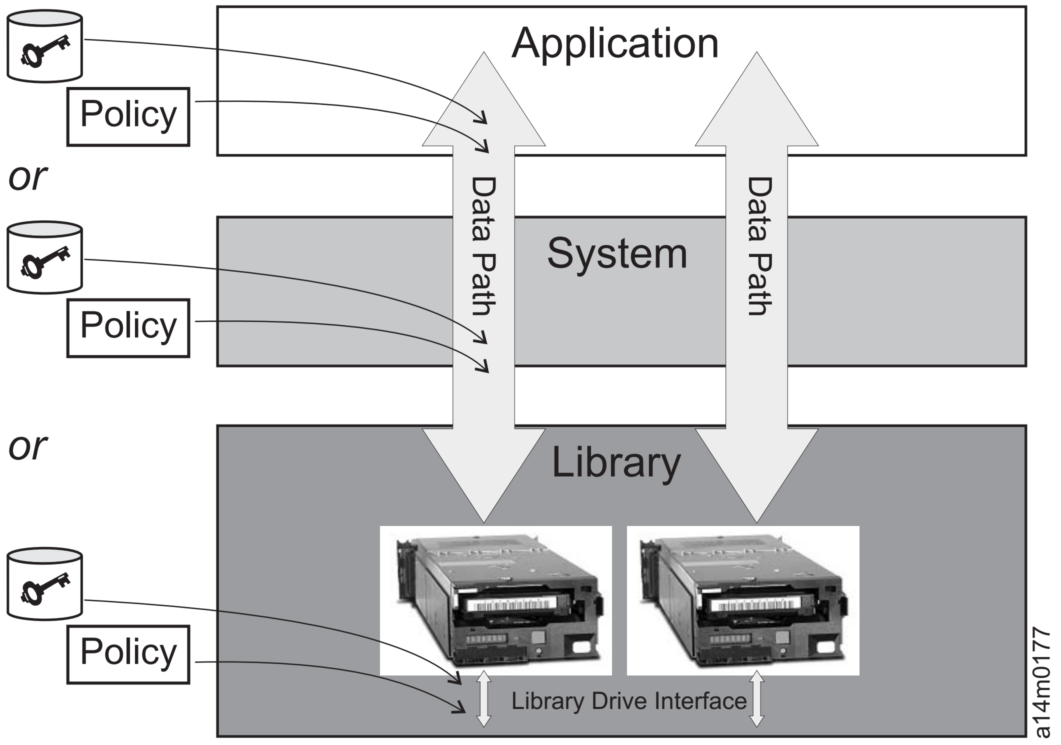


Figure 2. Three possible locations for encryption policy engine and key management.

Application Layer
An application program, separate from the key manager, initiates data transfer for tape storage, for example Tivoli Storage Manager.

System Layer
Everything between the application and the tape drives, for example the operating system, z/OS DFSMS, device drivers, and FICON/ESCON controllers.

Library Layer
The enclosure for tape storage, such as the IBM System Storage TS3500 Tape Library. A modern tape library contains an internal interface to each tape drive within it.

Application-Managed Tape Encryption

This method is best where operating environments run an application already capable of generating and managing encryption policies and keys, such as Tivoli Storage Manager. Policies specifying when encryption is to be used are defined through the application interface. The policies and keys pass through the data path between the application layer and the encrypting tape drives. Encryption is the result of interaction between the application and the encryption-enabled tape drive, and does not require any changes to the system and library layers. Because the application manages the encryption keys, data volumes written and encrypted using the application-managed encryption method can only be read by the same software application that wrote them.

The Encryption Key Manager is not required by, or used by, application-managed tape encryption.

Application-managed tape encryption on IBM TS1120 and TS1130 tape drives and LTO Ultrium 4 tape drives may use either of two encryption command sets:

- The IBM encryption command set developed for the Encryption Key Manager
- The T10 command set defined by the InterNational Committee for Information Technology Standards (INCITS)

Application-managed tape encryption using the TS1120 and TS1130 Tape Drive is supported in the following IBM libraries:

- IBM System Storage TS3400 Tape Library
- IBM System Storage TS3500 Tape Library
- IBM TotalStorage 3494 Tape Library

Application-managed tape encryption using LTO Ultrium 4 Tape Drives is supported in the following IBM tape drives and libraries:

- IBM System Storage TS2340 Tape Drive Express Model S43 and via Xcc/HVEC 3580S4X
- IBM System Storage TS3100 Tape Library
- IBM System Storage TS3200 Tape Library
- IBM System Storage TS3310 Tape Library
- IBM System Storage TS3500 Tape Library

For details about setting up Application-Managed tape encryption, see your Tivoli Storage Manager documentation or visit http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp for more information.

System-Managed Tape Encryption

This method is best for encryption on TS1120, TS1130 and LTO Ultrium 4 tape drives in Open Systems and System z operating environments where the applications that write or read from tape are not capable of performing the key management required for application-managed encryption.

Open Systems

Encryption policies specifying when to use encryption are set up in the IBM tape device driver. System-managed tape encryption and library-managed tape encryption interoperate with one another. In other words, a tape encrypted using system-managed encryption may be decrypted using library-managed encryption, and vice versa, provided they both have access to the same keys and certificates. Otherwise, this may not be feasible.

For details on setting up system-managed encryption on tape drives in an AIX, Linux, Windows, or Solaris environment, see *IBM Tape Device Drivers Installation and User's Guide*, and the *Planning and Operator Guide* for your tape library.

System z

Encryption policies specifying when to use encryption are set up in z/OS DFSMS™ (Data Facility Storage Management Subsystem) or implicitly through each instance of IBM device driver. Additional software products such as IBM Integrated Cryptographic Service Facility (ICSF) and IBM Resource Access Control Facility (RACF®) may also be used. Key generation and management is performed by the Encryption Key Manager, a Java application running on the host or externally on another host. Policy controls and keys pass through the data path between the system layer and the encrypting tape drives. Encryption is transparent to the applications.

For TS1120 and TS1130 tape drives connected to an IBM Virtualization Engine TS7700, encryption key labels are assigned on a per-storage pool basis using the TS7700 Maintenance Interface. DFSMS storage constructs are used by z/OS to control the use of storage pools for logical volumes, resulting in an indirect form of encryption policy management. For more information, see the white paper, *IBM Virtualization Engine TS7700 Series Encryption Overview,* available at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

With system-managed encryption, System z hosts can rekey an encrypted tape on the TS1120 and TS1130 tape drives. Refer to the appropriate operating system documentation for the mechanism that is used to initiate a rekey operation. For example, with z/OS, the existing IEHINITT utility is enhanced to support rekeying. Rekeying is useful for customers who export volumes to multiple business partners because it eliminates the need to rewrite the entire tape and enables a tape cartridge's data key to be reencrypted with a business partner's public key.

For details on setting up system-managed encryption on TS1120 and TS1130 tape drives in a System z platform environment, see *z/OS DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drives (3592)*.

Encryption Key Paths

In system-managed encryption on System z platforms, multiple key paths are supported by the tape controller.

In-Band Key Flow

In-band key flow, shown in Figure 3 on page 8, occurs between the Encryption Key Manager and the tape drive through a FICON proxy (only available in z/OS) on the FICON/ESCON interface. The FICON proxy, supports failover to the secondary key path on failure of first-specified Encryption Key Manager path addresses. Impact on controller service requirements is minimal.

The controller

- Reports drive status in SMIT displays
- Passes encryption-related errors from the drive to the host
- Reports to the host "encryption failure unit checks"
- Must be reconfigured whenever new encryption drives are introduced for attachment or when an encryption-capable drive is enabled for encryption.

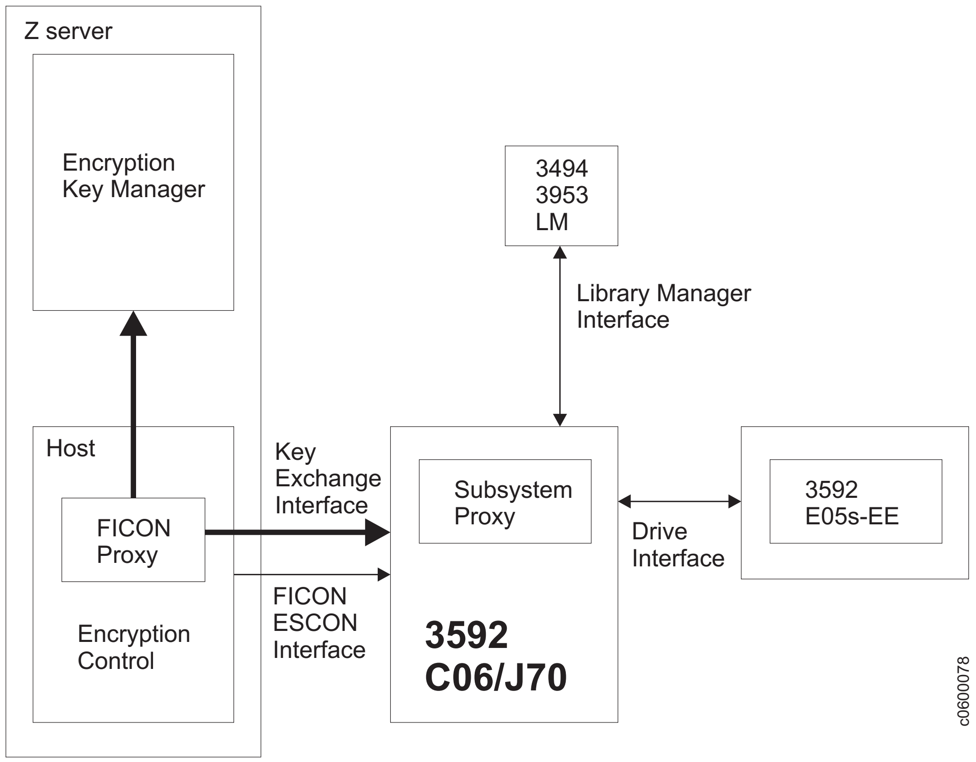


Figure 3. In-band encryption key flow

Out-of-Band Key Flow

Out-of-band key flow, shown in Figure 4 on page 9, occurs between the Encryption Key Manager and the tape drive through a subsystem proxy, located in the 3592 Controller or TS7700 Virtualization Engine, on the Encryption Key Manager interface. Impact on service requirements may be greater than for in-band key flow due to the introduction of two routers on the Encryption Key Manager interface, to and from the controller.

The controller and TS7700

- Supports failover to the secondary key path on failure of first-specified Encryption Key Manager path addresses
- Reports drive status in SMIT displays
- Passes encryption-related errors from the drive to the host
- Reports to the host "encryption failure unit checks"
- Must be reconfigured whenever new encryption drives are introduced for attachment or when an encryption-capable drive is enabled for encryption.

As many as two Encryption Key Manager IP/domain addresses (and as many as two ports) may be entered for each controller, as well as two Domain Name Server IP addresses.

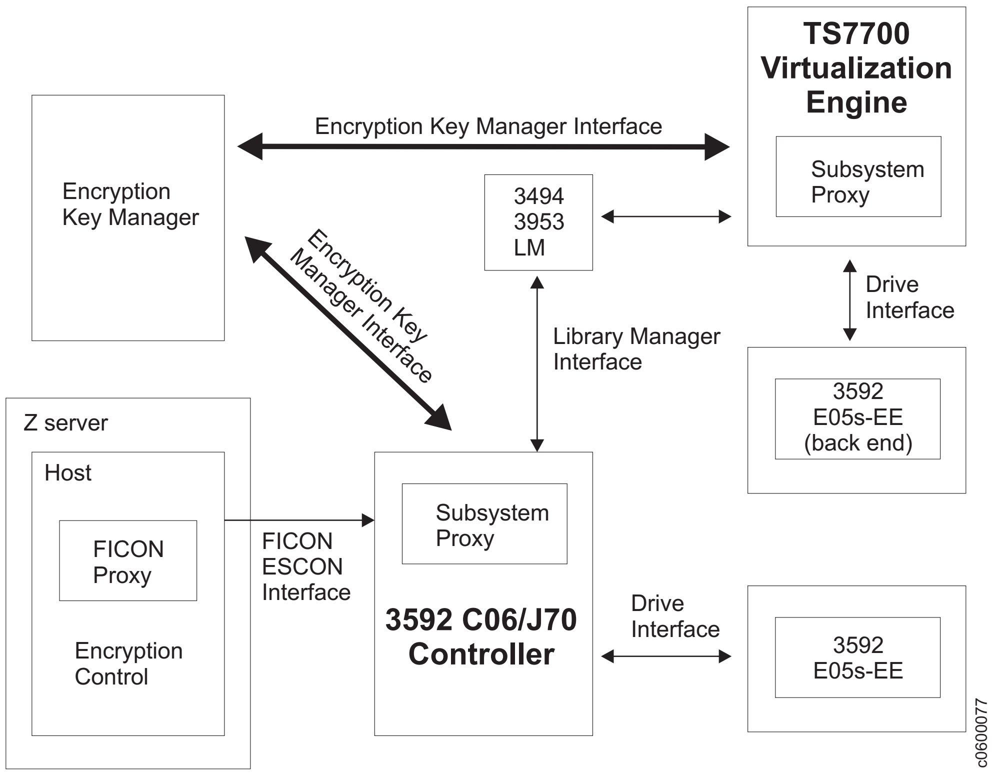


Figure 4. Out-of-band encryption key flow

Library-Managed Tape Encryption

This method is best for TS1120 and TS1130 tape drives and LTO Ultrium 4 Tape Drives in an open-attached IBM System Storage TS3100, TS3200, TS3310, TS3400 or TS3500 tape library, or IBM TotalStorage 3494 Tape Library. For TS3500, barcode encryption policies may be used to specify when to use encryption, and are set up through the IBM System Storage Tape Library Specialist Web interface. In such cases, policies are based on cartridge volume serial numbers. Library-managed encryption also allows other options, such as encryption of all volumes in a library, independent of bar codes. Key generation and management is performed by the Encryption Key Manager, a Java application running on a library network-attached host. Policy control and keys pass through the library-to-drive interface, therefore encryption is transparent to the applications.

Library-managed encryption, when used with certain applications such as Symantec Netbackup, includes support for an *internal label option*. When the internal label option is configured, the TS1120 or TS1130 Tape Drive or LTO Ultrium 4 Tape Drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. Refer to your *Tape Library Operator's Guide* for more information.

Note: If you use library-managed encryption and IBM tape and changer drivers running on Open Systems platforms (AIX, HP-UX, Linux, Solaris, Windows), information for bulk rekey is available in the *IBM Tape Device Drivers Installation and User's Guide*, available at ftp://ftp.software.ibm.com/storage/devdrvr/Doc/IBM_Tape_Driver_IUG.pdf.

System-managed tape encryption and library-managed tape encryption interoperate with one another. In other words, a tape encrypted using system-managed encryption may be decrypted using library-managed encryption, and vice versa, provided they both have access to the same keys and certificates. Otherwise, this may not be feasible.

About Encryption Keys

An encryption key is typically a random string of bits generated specifically to scramble and unscramble data. Encryption keys are created using algorithms designed to ensure that each key is unique and unpredictable. The longer the key constructed this way, the harder it is to break the encryption code. Both the IBM and T10 methods of encryption use 256-bit AES algorithm keys to encrypt data. 256-bit AES is the encryption standard currently recognized and recommended by the U.S. government, which allows three different key lengths. 256-bit keys are the longest allowed by AES.

Two types of encryption algorithms may be used by the Encryption Key Manager: symmetric algorithms and asymmetric algorithms. Symmetric, or secret key encryption, uses a single key for both encryption and decryption. Symmetric key encryption is generally used for encrypting large amounts of data in an efficient manner. 256-bit AES keys are symmetric keys. Asymmetric, or public/private encryption, uses a pair of keys. Data encrypted using one key can only be decrypted using the other key in the public/private key pair. When an asymmetric key pair is generated, the public key is typically used to encrypt, and the private key is typically used to decrypt.

The Encryption Key Manager uses both symmetric and asymmetric keys; symmetric encryption for high-speed encryption of user or host data, and asymmetric encryption (which is necessarily slower) for protecting the symmetric key.

Encryption keys may be generated by the Encryption Key Manager, by applications such as Tivoli Storage Manager, or by a utility such as keytool. The responsibility for generating AES keys and the manner in which they are transferred to the tape drive depends on the tape drive type and the method of encryption management. However, it may be helpful to understand the difference between how the Encryption Key Manager uses encryption keys and how other applications use them.

How the Encryption Key Manager Processes Encryption Keys

| **On TS1120 and TS1130 Tape Drives**

In system-managed and library-managed tape encryption, unencrypted data (clear text) is sent to the tape drive and converted to ciphertext using a symmetric 256-bit AES Data Key (DK) generated by the Encryption Key Manager. The cyphertext is then written to tape. The Encryption Key Manager uses a single, unique Data Key for each Enterprise Tape Cartridge. This Data Key is also encrypted, or wrapped, by the Encryption Key Manager using the public key from an asymmetric Key Encrypting Key (KEK) pair. This process creates an Externally Encrypted Data Key (EEDK). The EEDK is written to the cartridge memory and to three additional places on the tape media in the cartridge. The tape cartridge now holds both the encrypted data and the means to decrypt it for anyone holding the private part of the KEK pair. Figure 5 on page 11 illustrates this process.

The DK is also wrapped a second time, possibly using the public key of another party, to create an additional EEDK. Both EEDKs can be stored on the tape cartridge. In this way, the tape cartridge can be shipped to a business partner holding the corresponding private key that would allow the DK to be unwrapped and the tape decrypted by the business partner.

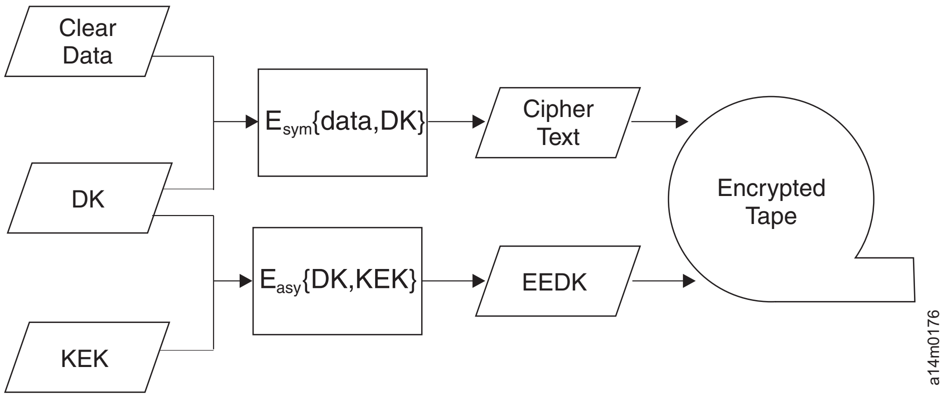


Figure 5. Encryption Using both Symmetric and Asymmetric Encryption Keys. System-Managed and Library-Managed Encryption on TS1120 and TS1130 tape drives

On the LTO Ultrium 4 Tape Drive

In system-managed and library-managed tape encryption, unencrypted data is sent to the LTO Ultrium 4 Tape Drive and converted to ciphertext using a pre-generated symmetric Data Key (DK) from a keystore available to the Encryption Key Manager, and is then written to tape. The Encryption Key Manager selects a pre-generated Data Key in round robin fashion. Data Keys are reused on multiple tape cartridges when an insufficient number of Data Keys have been pre-generated. The Data Key is sent to the LTO 4 tape drive in encrypted, or wrapped, form by the Encryption Key Manager. The LTO 4 tape drive unwraps this Data Key and uses it to perform encryption or decryption. However, no wrapped key is stored anywhere on the LTO 4 tape cartridge. This is a major | difference between the way TS1120 or TS1130 and LTO devices operate with the Encryption Key Manager. Once the encrypted volume is written, the Data Key must be accessible based on the alias or key label, and available to the Encryption Key Manager in order for the volume to be read. Figure 6 on page 12 illustrates this process.

The Encryption Key Manager also gives you the ability to organize your symmetric keys for LTO encryption into key groups. In this way, you can group keys according to the type of data they encrypt, the users who have access to them, or by any other meaningful characteristic. See "Creating and Managing Key Groups" on page 100 for more information.

Encryption Key Processing by Other Applications (Encryption Key Manager not Used)

| **On TS1120, TS1130, and LTO Ultrium 4 Tape Drives**

In application-managed tape encryption, unencrypted data (clear text) is sent to the tape drive and converted to ciphertext using a symmetric Data Key (DK) provided by the application, and is then written to tape. The Data Key is not stored anywhere on the tape cartridge. Once the encrypted volume is written, the Data

Key must be in a location available to the application, a server database, for example, in order for the volume to be read.

TS1120 and TS1130, and LTO Ultrium 4 tape drives can use applications such as Tivoli Storage Manager for application-managed encryption. Tivoli Storage Manager uses a single, unique Data Key for each tape cartridge.

Alternatively, the tape drives can be used by applications that use the T10 command set to perform encryption. The T10 command set uses symmetric 256-bit AES keys provided by the application. T10 can use multiple, unique Data Keys per tape cartridge, and even write encrypted data and clear data to the same tape cartridge. When the application encrypts a tape cartridge, it selects or generates a Data Key using a method determined by the application and sends it to the tape drive. The key is **not** wrapped with an asymmetric public key and it is **not** stored on the tape cartridge. Once the encrypted data is written to tape, the Data Key must be in a location available to the application in order for the data to be read.

The process for application-managed tape encryption (as well as system-managed and library-managed encryption on LTO) is shown in Figure 6.

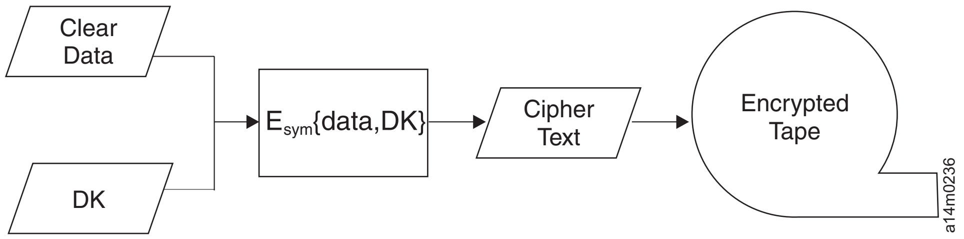


Figure 6. Encryption Using only Symmetric Encryption Keys. Application-Managed Encryption on TS1120 and TS1130 tape drives, and System-Managed, Library-Managed, and Application-Managed Encryption on LTO Ultrium 4 tape drives.

In Summary

The number of encryption keys that may be used for each volume depends on the tape drive, the encryption standard, and method used to manage the encryption. For transparent encryption of LTO 4, (that is, using system-managed or library-managed encryption with the Encryption Key Manager,) the uniqueness of Data Keys depends on the availability of a sufficient number of pre-generated keys to the Encryption Key Manager.

Table 2. Encryption Key Summary

	Keys used by		
Encryption Management Method	**TS1120 and TS1130 (IBM Encryption)**	**LTO 4 (IBM Encryption)**	**TS1120 and TS1130/LTO 4 (T10 Encryption)**
System-Managed Encryption / Library-Managed Encryption (Encryption Key Manager)	1 unique DK / cartridge	1 DK / cartridge	N/A
Application-Managed Encryption (no Encryption Key Manager)	1 unique DK / cartridge	1 DK / cartridge	Multiple DKs / cartridge
DK = Symmetric AES 256-bit Data Key			

Chapter 2. Planning Your Encryption Key Manager Environment

This section is intended to provide information to allow you to determine the best Encryption Key Manager configuration for your needs. Many factors must be considered when you are planning how to set up your encryption strategy. Please review these topics with care.

Encryption Setup Tasks at a Glance

Before you can use the encryption capability of the tape drive, you must be sure that certain software and hardware requirements are met. The following checklists are intended to help you meet these requirements.

Note: Please contact your IBM Representative for additional information about encryption on the IBM TS1120, TS1130, or LTO Ultrium 4 Tape Drive.

Encryption Key Manager Setup Tasks

Before you can encrypt tapes, the Encryption Key Manager must first be configured and running so that it can communicate with the encrypting tape drives. The Encryption Key Manager need not be running while tape drives are being installed, but it must be running in order to perform encryption.

These are the tasks you must perform before using the Encryption Key Manager.

- Decide what system platform(s) to use as Encryption Key Manager server(s).
- Upgrade server operating system if necessary. (See “Hardware and Software Requirements” on page 18.)
- Upgrade the Java Virtual Machine if necessary. (See “Hardware and Software Requirements” on page 18.)
- Install Java Unrestricted Policy Files. (See “Hardware and Software Requirements” on page 18.)
- Upgrade the Encryption Key Manager JAR. See “Downloading the Latest Version Encryption Key Manager” on page 132.
- Decide on keystore type. (See “Which Keystore is Right for You” on page 37.)
- Create keys, certificates, and key groups.

  “Example 1: Using the Java Keytool and JCEKS on z/OS” on page 50

  “Example 2: Using the JCE4758KS or JCECCAKS Keystore with the Java Hwkeytool on z/OS” on page 52 (z/OS only)

  “Example 3: Using the JCERACFKS or JCE4758RACFKS/JCECCARACFKS Keystore on z/OS” on page 55 (z/OS only)

  “Create a Keystore and Certificates (Unix-based and Windows Operating Systems)” on page 87

  “Generating Keys and Aliases for Encryption on LTO 4” on page 94

  “Creating and Managing Key Groups” on page 100

  “Using Digital Certificate Manager to Create and Manage Keystores on i5/OS” on page 105
- If necessary, import keys and certificates (See previous step).

- Define the configuration properties file. (See Chapter 5, "Configuring the Encryption Key Manager," on page 135.)
- Define tape drives to the Encryption Key Manager or set **drive.acceptUnknownDrives** configuration property value on. (See "adddrive" on page 161 to define drives explicitly, or see "Automatically Update Tape Drive Table" on page 135.)
- Start the Encryption Key Manager server. (See "Starting and Stopping the Key Manager Server" on page 157. To start the Encryption Key Manager on z/OS See "Quick Test Running Encryption Key Manager Under USS" on page 71.)
- Start the command line interface client. (See "The Command Line Interface Client" on page 159) .

Planning for Application-Managed Tape Encryption

In order to perform encryption the following is required:

- Encryption-capable TS1120, or TS1130 or LTO Ultrium 4 tape drive(s)

Application-Managed Tape Encryption Setup Tasks

Any task not identified as an IBM service task is the responsibility of the customer.

1. Install and cable the TS1120 or TS1130 tape drive (IBM service task) or LTO Ultrium 4 Tape Drive.
   - Update library firmware (3494, TS3500 where applicable)
   - Update tape drive firmware (all tape drives in same library or environment)
2. Encryption-enable the TS1120 or TS1130 or LTO Ultrium 4 tape drive. Refer to *IBM System Storage TS3500 Operator's Guide* for configuring TS1120 or TS1130 or LTO Ultrium 4 tape drives on TS3500. For TS1120 or TS1130 tape drives on all others, this is an IBM service task.
3. Install appropriate IBM tape device driver level (Atape, for example) where required by application.
4. Set up encryption policies. Refer to *IBM Tivoli Storage Manager for AIX Administrator's Guide* .
5. Perform write/read operation to test encryption.
6. Verify encryption of the test volume by Autonomic Management Engine (AME): issue
   ```
   QUERY VOLUME FORMAT=DETAILED
   ```
   Verify that `Drive Encryption Key Manager` is set to `Tivoli Storage Manager`.

Planning for System-Managed Tape Encryption

In order to perform system-managed encryption the following is required:

- Encryption-capable TS1120 or TS1130 or LTO Ultrium 4 tape drive(s).
- Key(s) and corresponding certificates.
- IBM Encryption Key Manager component for the Java platform.
- Routers and cables for out-of-band Encryption Key Manager-to-TS1120 or TS1130 tape drive path (System z platforms only).

Setup Tasks for System-Managed Tape Encryption on IBM System z Platforms

Any task not identified as an IBM service task is the responsibility of the customer.

1. Install and cable the TS1120 or TS1130 tape drive (IBM service task).

- Update tape drive firmware (3592 Models E05, J1A in same environment)
- Update 3494, TS3500, and 3953 tape system library firmware (System z platforms or 3953 in heterogeneous environment)
- Update 3592 Models C06, J70 Tape Controller firmware (System z platforms or tape controllers in heterogeneous environment) (optional)
- Update TS7700 Virtualization Engine microcode.

2. Encryption-enable the TS1120 or TS1130 tape drive. Refer to *IBM System Storage TS3500 Operator's Guide* for configuring TS1120 or TS1130 tape drive on TS3500. The 3494 Web Specialist can now be used to enable encryption the TS1120 or TS1130 drive in a 3494 Tape Library. Refer to *IBM TotalStorage Automated Tape Library (3494) Operator's Guide*. For TS7700-attached drives, specify the system-managed encryption method. For others, this is an IBM service task.
3. Install tape controller code update, Feature Code 5595 (IBM service task).
4. Install, cable, and configure routers to the Encryption Key Manager, Feature Code 5593 (for out-of-band path to the Encryption Key Manageronly) (IBM service task).
   - Define Primary/Secondary Encryption Key Manager IP ports for the tape controller.
5. Update z/OS and DFSMS host software with appropriate PTFs.
6. Install Feature Code 9900 License Key on TS7700.
7. Set up encryption policies.
   - Update DFSMS Data Class to specify encryption (recording format EE2) and other optional parameters (media type, performance scaling, etc.) as appropriate.
   - Specify the key labels through the DD statement, data class or Encryption Key Manager defaults.
   - Update other DFSMS polices (as appropriate) to steer allocation to correct library.
   - Encryption on the TS7700 VE is controlled on a storage pool basis. Use the Maintenance Interface (MI) web interface for the TS7700 VE "Pool Encryption Settings" panel (in the "Configuration" group) to specify the key labels and modes to use for each storage pool.

   Refer to *IBM z/OS DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drives (3592)*.
8. For in-band key management use the IECIOSxx PARMLIB member or **SETIOS** command to define Primary/Secondary Encryption Key Manager. Also define the IOSAS OMVS segment to RACF.
9. Make the appropriate HCD changes.
10. Determine if coexistence support is needed.
11. Contact your tape management system or application vendor for any required code changes and any installation exit changes that are needed.
12. Set up the system-managed encryption method. For 3494 or stand-alone drives, have the your IBM service representative update the drives. For TS3500, update using the IBM System Storage Tape Library Specialist.
13. Schedule an IPL
14. Verify encryption:

    For in-band path to the Encryption Key Manager:

    a. Use the `DISPLAY IOS,EKM` command (with the VERIFY option) to verify the in-band path to the Encryption Key Manager.

b. Verify that a job (or application) requesting encryption (through data class) has its data encrypted.

For out-of-band path to the Encryption Key Manager:

a. Use RAS functions to verify (IBM service task) Encryption Key Manager paths and encryption configuration.

Setup Tasks for System-Managed Tape Encryption on Open Systems Platforms

Any task not identified as an IBM service task is the responsibility of the customer.

1. Install and cable the TS1120 or TS1130 tape drive (IBM service task) or LTO Ultrium 4 tape drive.
   - Update tape drive firmware (all tape drives in same environment)
   - Update 3494 and TS3500 Tape Library firmware, where applicable
   - For LTO Ultrium 4 Tape Drives, install Feature Code 1604 (IBM service task).
2. Encryption-enable the TS1120 or TS1130 or LTO Ultrium 4 tape drive. Refer to *IBM System Storage TS3500 Operator's Guide* for configuring TS1120/TS1130 or LTO Ultrium 4 Tape Drive on TS3500. The 3494 Web Specialist can now be used to enable encryption the TS1120 or TS1130 drive in a 3494 Tape Library. Refer to *IBM TotalStorage Automated Tape Library (3494) Operator's Guide*. For TS1120 or TS1130 tape drives on others, this is an IBM service task.
3. Update the IBM device driver to the latest level (ftp://ftp.software.ibm.com/storage/devdvr). Refer to *IBM Tape Device Drivers Installation and User's Guide* for details.
4. Update the Encryption Key Manager Proxy Config file with Encryption Key Manager IP Addresses.
5. Update device attributes
   - Use System Encryption FCP Proxy Manager.
   - System Encryption for Write Commands at BOP.

   Refer to *IBM Tape Device Drivers Installation and User's Guide*
6. Use tapeutil functions to verify Encryption Key Manager paths and encryption configuration.

Planning for Library-Managed Tape Encryption

In order to perform encryption the following is required:

- Encryption-capable TS1120 or TS1130 or LTO Ultrium 4 tape drive(s)
- Keystore
- IBM Encryption Key Manager component for the Java platform

Library-Managed Tape Encryption Tasks

Any task not identified as an IBM service task is the responsibility of the customer.

1. Install and cable the TS1120 or TS1130 tape drive (IBM service task) or LTO Ultrium 4 tape drive.
   - Update tape system library firmware (3494 or TS3500)
   - Update tape drive firmware (all tape drives in same library)
   - For TS1120 or TS1130 tape drives in 3494 or TS3500, order Feature Code 9900 for Encryption Configuration (IBM service task)
   - For LTO Ultrium 4 tape drives, install Feature Code 1604 for Transparent LTO Encryption (IBM service task).

2. Use IBM System Storage Tape Library Specialist to enable TS1120 or TS1130 or LTO Ultrium 4 tape drives and 3494 or TS3500 Tape Library for library-managed tape encryption (refer to appropriate tape library operator guide.)
   - Add Encryption Key Manager IP addresses
   - Specify key label
   - Set up scratch encryption policy
3. Set up key mapping for ILEP (optional)
4. Use library diagnostic functions to verify Encryption Key Manager paths and encryption configuration.

TS1120/TS1130 Tape Drive Installation Process for Encryption

Before the IBM service representative installs or upgrades TS1120 (3592 Model E05) or TS1130 (3592 Model E06) Tape Drives for encryption, you must:

- Decide which method of encryption management to use (application-managed encryption, system-managed encryption, or library-managed encryption). Refer to "Managing Encryption" on page 3.
- Install and configure the Encryption Key Manager. Refer to Chapter 3, "Installing the Encryption Key Manager and Keystores," on page 45.

Encryption Setup Procedure for IBM Service

The following steps are performed by the IBM Service Representative:

1. Record serial numbers of all 3592 E05, E06, or EU6 tape drives and provide these to the customer (optional if customer plans to set the Encryption Key Manager configuration to drive.acceptUnknownDrives=true for automatic addition of tape drives to tape drive table).
2. Install 3592 E05/E06/EU6 Tape Drives
   a. If adding new 3592 E05, E06, or EU6 encryption-capable drives to an existing frame, refer to *3494 Maintenance Information* or *3584 Maintenance Information* for installation instructions. When installation is complete, continue at step 3.
   b. If replacing current 3592 E05, E06, or EU6 drives with new ones, refer to *3494 Maintenance Information* or *3584 Maintenance Information* for drive FRU replacement instructions. When replacement is complete, continue at step 3.
   c. If upgrading current 3592 E05, E06, or EU6 drives, refer to *Feature Code 5592 MES Installation Instructions*. When upgrade is complete, continue at step 3.
3. Configure 3592 E05, E06, or EU6 tape drives for Encryption.
   a. If 3592 E05, E06, or EU6 tape drives are installed in an Enterprise System and connected to a 3592 C06 or J70, you must use system-managed encryption only. If the drives are installed in a 3494 or Standalone Frame, go to step 4. If the drives are installed in a 3584 library, configure and encryption-enable the tape drives using the System Storage Tape Specialist web interface. When tape drives are configured, continue at step 5.
   b. If 3592 E05, E06, or EU6 tape drives are installed in a 3494 Open System, the 3494 Web Specialist can now be used to enable-encryption them. Refer to *IBM TotalStorage Automated Tape Library (3494) Operator's Guide*. Then continue at step 5.
   c. If 3592 E05, E06, or EU6 tape drives are installed in a 3584 Open System, the customer configures and encryption-enables the tape drives using the System Storage Tape Specialist web interface. When tape drives are configured, continue at step 5.

4. Encryption-enable the 3592 tape drives by following the procedure for "Setting Drive Encryption" in *3592 Maintenance Information*. When tape drives are encryption-enabled, continue at step 6.
5. Use the System Storage Tape Specialist web interface to verify that the 3592 Tape Drives are encryption-enabled. For example, select **Manage Library** > **By Logical Library** > (Select Library) **Modify Encryption Method** > **GO**.
6. If the 3592 E05, E06, or EU6 tape drives are encryption-enabled for Enterprise Systems, follow the 3592 C06 or J70 *Maintenance Information* and *Installation and Configuration* Guide (ICG) to configure the controllers for encryption, then continue at step 7. If the 3592 E05, E06, or EU6 tape drives are installed in a rack, continue at step 8.
7. Run Library Verify.
8. Go to End-Of-Call Procedures in the appropriate *Maintenance Information*.

LTO Ultrium 4 Tape Drive Installation Process for Encryption

- Decide which method of encryption management to use (application-managed encryption, system-managed encryption, or library-managed encryption). Refer to "Managing Encryption" on page 3.
- Install and configure the Encryption Key Manager. Refer to Chapter 3, "Installing the Encryption Key Manager and Keystores," on page 45.
- Use the System Storage Tape Specialist web interface to verify that the LTO Ultrium 4 Tape Drives are encryption-enabled. Select **Manage Library** > **By Logical Library** > (Select Library) **Modify Encryption Method** > GO.

Refer to "Configuring for LTO 4 Encryption" on page 141 for more information on LTO.

Hardware and Software Requirements

Note: Only the IBM version of the Java Runtime Environment (JRE) for each of the following platforms supports the Encryption Key Manager.

z/OS Solution Components

Operating Systems

TS1120 - z/OS and z/OS.e 1.4 and later - PTFs required for updates to z/OS and DFSMS.

Note: For z/OS V1R6 and V1R7, refer to enabling APAR OA15685 and for z/OS V1R8, refer to enabling APAR OA17562. For z/OS V1R4 and V1R5, refer to enabling APAR OA18111. For additional information on the support being provided, refer to the 3592 PSP bucket.

TS1130 - z/OS 1.7 and later - PTFs required for updates to z/OS and DFSMS.

Note: Refer to enabling APAR OA22119. For additional information on the support being provided, refer to the 3592 PSP bucket.

RACF APAR OA13030 is required on z/OS V1R4, V1R5, V1R6, and V1R7 for greater than 1024 modulus support.

Encryption Key Manager (Running on z/OS or z/OS.e)

The Encryption Key Manager is included with the IBM Java SDK. Currently supported versions of IBM Java SDK for z/OS and z/OS.e are listed at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 and available at http://www.ibm.com/servers/eserver/zseries/software/java. The Encryption Key Manager is not currently supported using the z/OS 64 bit SDKs.

Note: Regardless of which IBM SDK version you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your **$JAVA_HOME/lib/security** directory with an unrestricted version of these files. These unrestricted policy files are required by the Encryption Key Manager in order to serve AES keys. The preferred method to do this on z/OS is to copy the unrestricted policy files that are shipped in the z/OS java SDK build under the jce demo directory. You only need to copy them to the **lib/security** directory just as you see here in this example:

```
cp /usr/lpp/java/J1.4/demo/jce/policy-files/unrestricted/*
/usr/lpp/java/J1.4/lib/security
```

Alternatively, the unrestricted policy files can be downloaded from the following website: https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. Be sure to select the "Unrestricted JCE Policy files for SDK for all newer versions, Version 1.4.2+," which will work for Java 1.4.2 and higher (including Java 5.0 SDK and Java 6.0). Do not select the 1.4.1 version as these are not compatible.

Control Units

Table 3. Control Unit Requirements for z/OS

Control Unit	Model/PID Number	Type of Update
3592 J70	3592-J70	Firmware update shipped with 3592-J70 FC 5595 and FC 9595
TS1120 C06	3592-C06	Firmware update shipped 3592-C06 FC 5595 and FC 9595
Router for Encryption Key Manager Attach (only required on tape controllers for out-of-band support)		FC 5593 or FC 9593
3494 Lxx	3494 Lxx	Firmware shipped with 3592-J70 or C06 FC 5595 and FC 9595
3953 L05	3953-L05	Firmware shipped with 3592-J70 or C06 FC 5595 and FC 9595

Virtualization Engine TS7700

TS7700 Feature Code 9900 is required for encryption. TS7700 Feature Code 0521 provides the latest firmware updates.

Table 4. Virtualization Engine TS7700 Connected Library Requirements for z/OS

Tape Library	Model/PID Number	Type of Update
TS3500	3584-L22, L23, D22, D23	For firmware update, visit http://www.ibm.com/servers/storage/support/lto/3584/downloading.html

Table 4. Virtualization Engine TS7700 Connected Library Requirements for z/OS (continued)

Tape Library	Model/PID Number	Type of Update
3494	3494-D22	Firmware update shipped with FC 5596 and FC 9596

Tape Libraries

Table 5. Tape Library Requirements for z/OS

Tape Library	Model/PID Number	Type of Update
TS3500	3584-L22, L23, D22, D23	For firmware update, visit http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494	3494-L22, D22, D24	Firmware update shipped with FC 5595 and FC 9595

Tape Drive

Table 6. Tape Drive Requirements for z/OS

Tape Drive	Model/PID Number	Type of Update
TS1120	3592-E05 New drive order	Order 3592-E05 FC9592 and 3592-J70 or C06 FC9595 or FC5595
	3592-E05 Field upgrade for encryption	
TS1130	3592-E06	All E06 and EU6 drives are encryption capable. Order FC9596 for the E06 or EU6 to configure for encryption-enablement.
	3592-EU6	

z/VM Solution Components

Operating Systems

z/VM 5.2 and later, plus DFSMS/VM FL221 if running z/VSE guests.

PTFs for the following APARs are required:

z/VM 5.2.0 - VM64063
z/VM 5.2.0 and later - VM64459
DFSMS/VM FL221 - VM64062 & VM64458

Note: The Encryption Key Manager does not run on z/VM but is supported through an out-of-band connection to an Encryption Key Manager server running on z/OS. See “Planning for System-Managed Tape Encryption” on page 14. See “z/OS Solution Components” on page 18 for z/OS prerequisites.

Refer to the latest editions of the following publications for more information:

- *z/VM CP Planning and Administration* (SC24-6083) - Chapter 23 describes the overall usage of encryption support on z/VM.

- *zVM CP Commands and Utilities* (SC24-6081) - Contains specifics about each command.

i5/OS Solution Components

Operating Systems

i5/OS V5R3
i5/OS V5R4
IBM i V6R1

Encryption Key Manager (Running on i5/OS)

The Encryption Key Manager is included with the IBM Java SDK. Currently supported versions of IBM Java SDK for i5/OS and IBM i are listed at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

Tape Libraries

Table 7. Tape Library Requirements for i5/OS

Tape Library	Model/PID Number	Type of Update
TS3100 (LTO-4)	3573-L2U	For firmware update, http://www.ibm.com/servers/storage/support/lto to select your library, then click the **Download** tab.
TS3200 (LTO-4)	3573-L4U	
TS3310 (LTO-4)	3576-L5B, E9U	
TS3400 (TS1120, TS1130)	3577-L5U	
TS3500 (TS1120, TS1130 or LTO-4)	3584-L22, L23, L32, L52, L53, D22, D23, D32, D52, D53	
3494 (TS1120, TS1130)	3494-L22, D22	Firmware update FC 0520

Tape Drive

Table 8. Tape Drive Requirements for i5/OS

Tape Drive	Model/PID Number	Type of Update
TS1120	3592-E05 New drive order	Order 3592-E05 FC 9592 (plant) or FC 5592 (field). When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order 3592-E05 FC 9596 or FC 5596.
	3592-E05 Field upgrade for encryption	
TS1130	3592-E06	All E06 and EU6 drives are encryption-capable. When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order FC 9596 or FC 5596 for the 3592 E06 or EU6.
	3592-EU6	

Table 8. Tape Drive Requirements for i5/OS (continued)

Tape Drive	Model/PID Number	Type of Update
TS1040	3588-F4A	For system-managed and library-managed encryption, order 3584 Lxx FC 1604. No features required for application-managed encryption.
TS2340	3580-L43, 3580-S43	No features required for application-managed encryption.

AIX Solution Components

Operating Systems

AIX 5.1
AIX 5.2 (5692A5L or 5765-E62)
AIX 5.3 (5765-G03)

| Refer to the IBM System Storage Interoperation Center (SSIC) at
| http://www.ibm.com/systems/support/storage/config/ssic/
| displayesssearchwithoutjs.wss?start_over=yes for Open Systems configuration
| information.

Encryption Key Manager (Running on AIX)

| The Encryption Key Manager is included with the IBM Java SDK. Currently
| supported versions of IBM Java SDK for System p server with AIX are listed at
| http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 and available at
| http://www.ibm.com/developerworks/java/jdk/aix/service.html.

Note: Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your *java_home*/**usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files the
| Encryption Key Manager requires in order to serve AES keys. Be sure to
| select the "Unrestricted JCE Policy files for SDK for all newer versions,
| Version 1.4.2+," which will work for Java 1.4.2 and higher (including Java
| 5.0 SDK and Java 6.0). Do not select the 1.4.1 version as these are not
compatible.

Device Driver (AIX System-Managed Tape Encryption Only)

Table 9. Device Driver Requirements for AIX

Device Driver	Type of Update
TS1120, TS1130, and LTO Ultrium 4 Tape Drives	Included in Device Driver Web Download at: ftp://ftp.software.ibm.com/storage/devdrvr/

Tape Libraries

Table 10. Tape Library Requirements for AIX

Tape Library	Model/PID Number	Type of Update
TS3100 (LTO-4)	3573-L2U	For firmware update, http://www.ibm.com/servers/storage/support/lto to select your library, then click the **Download** tab.
TS3200 (LTO-4)	3573-L4U	
TS3310 (LTO-4)	3576-L5B, E9U	
TS3400 (TS1120, TS1130)	3577-L5U	
TS3500 (TS1120, TS1130, or LTO-4)	3584-L22, L23, L32, L52, L53, D22, D23, D32, D52, D53	
3494 (TS1120, TS1130)	3494-L22, D22	Firmware update FC 0520

Tape Drive

Table 11. Tape Drive Requirements for AIX

Tape Drive	Model/PID Number	Type of Update
TS1120	3592-E05 New drive order	Order 3592-E05 FC 9592 (plant) or FC 5592 (field). When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order 3592-E05 FC 9596 or FC 5596.
	3592-E05 Field upgrade for encryption	
TS1130	3592-E06	All E06 and EU6 drives are encryption-capable. When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order FC 9596 or FC 5596 for the 3592 E06 or EU6.
	3592-EU6	
TS1040	3588-F4A	For system-managed and library-managed encryption, order 3584 Lxx FC 1604. No features required for application-managed encryption.
TS2340	3580-L43, 3580-S43	No features required for application-managed encryption.

Linux Solution Components

Operating Systems

System z RHEL 4	System p RHEL 4	System x RHEL 4
System z RHEL 5	System p RHEL 5	System x RHEL 5
System z SLES 9	System p SLES 9	System x SLES 9
System z SLES 10	System p SLES 10	System x SLES 10
	System p Asianux2.0	System x Asianux2.0

Refer to the IBM System Storage Interoperation Center (SSIC) at http://www.ibm.com/systems/support/storage/config/ssic/displayesssearchwithoutjs.wss?start_over=yes for Open Systems configuration information.

Encryption Key Manager (Running on Linux)

The Encryption Key Manager is included with the IBM Java SDK. Currently supported versions of IBM Java SDK for System p, System z, System x, and other Intel servers running Linux are listed at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504. These IBM SDKs are available at http://www.ibm.com/developerworks/java/jdk/linux/download.html.

Note: Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files the Encryption Key Manager requires in order to serve AES keys. Be sure to select the "Unrestricted JCE Policy files for SDK for all newer versions, Version 1.4.2+," which will work for Java 1.4.2 and higher (including Java 5.0 SDK and Java 6.0). Do not select the 1.4.1 version as these are not compatible.

Device Driver (Linux System-Managed Tape Encryption Only)

Table 12. Device Driver Requirements for Linux

Device Driver	Type of Update
TS1120, TS1130, and LTO Ultrium 4 Tape Drives	Included in Device Driver Web Download at: ftp://ftp.software.ibm.com/storage/devdrvr/

Tape Libraries

Table 13. Tape Library Requirements for Linux

Tape Library	Model/PID Number	Type of Update
TS3100 (LTO-4)	3573-L2U	For firmware update, http://www.ibm.com/servers/storage/support/lto to select your library, then click the **Download** tab.
TS3200 (LTO-4)	3573-L4U	
TS3310 (LTO-4)	3576-L5B, E9U	
TS3400 (TS1120)	3577-L5U	
TS3500 (TS1120 or LTO-4)	3584-L22, L23, L32, L52, L53, D22, D23, D32, D52, D53	
3494 (TS1120)	3494-L22, D22	Firmware update FC 0520

Tape Drive

Table 14. Tape Drive Requirements for Linux

Tape Drive	Model/PID Number	Type of Update
TS1120	3592-E05 New drive order	Order 3592-E05 FC 9592 (plant) or FC 5592 (field). When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order 3592-E05 FC 9596 or FC 5596.
	3592-E05 Field upgrade for encryption	
TS1130	3592-E06	All E06 and EU6 drives are encryption-capable. When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order FC 9596 or FC 5596 for the 3592 E06 or EU6.
	3592-EU6	
TS1040	3588-F4A	For system-managed and library-managed encryption, order 3584 Lxx FC 1604. No features required for application-managed encryption.
TS2340	3580-L43, 3580-S43	No features required for application-managed encryption.

HP, Sun, and Windows Solution Components

Operating Systems

HP-UX 11i, 11.23Pl
Sun Solaris 8, 9, 10
Windows Server 2003

Refer to the IBM System Storage Interoperation Center (SSIC) at http://www.ibm.com/systems/support/storage/config/ssic/displayesssearchwithoutjs.wss?start_over=yes for Open Systems configuration information.

Encryption Key Manager (Running on HP, Sun, or Windows)

Currently supported versions of IBM Java SDK for HP servers running HP-UX, Sun Sparc servers running Solaris, or IBM System x™ and other Intel® servers running Windows, are listed at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

The IBM Encryption Key Manager component for the Java platform is required to enable System-Managed and Library-Managed Encryption on HP-UX, Sun Solaris, and Microsoft Windows. The IBM TotalStorage Productivity Center (TPC) Basic Edition licensed program product 5608-B01 is required. TPC Basic Edition electronic availability via Passport Advantage® and media availability via Passport

Advantage and AAS includes the IBM Encryption Key Manager component for the Java platform to run on HP-UX, Sun Solaris, and Microsoft Windows. Note that TPC Basic Edition is a chargeable program product, whereas the previous TPC-LE was a no-charge offering. Several maintenance options are also available to order with the TPC Basic Edition. For more information on ordering and pricing for TPC Basic Edition, licensed program product 5608-B01, refer to http://www.ibm.com/systems/storage/software/center/index.html.

Note: Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk for Java 5.0 on Sun, xSeries® and HP-UX. These install the unrestricted policy files the Encryption Key Manager requires in order to serve AES keys. You will need to get these file from the SUN Java website for the Sun and HP-UX platforms using Java at 1.4.2. Be sure to select the "Unrestricted JCE Policy files for SDK for all newer versions, Version 1.4.2+," which will work for Java 1.4.2 and higher (including Java 5.0 SDK and Java 6.0). Do not select the 1.4.1 version as these are not compatible.

Device Driver (Solaris and Windows System-Managed Tape Encryption Only)

Table 15. Device Driver Requirements for Solaris and Windows

Device Driver	Type of Update
TS1120 TS1130, and LTO Ultrium 4 Tape Drives	Included in Device Driver Web Download at: ftp://ftp.software.ibm.com/storage/devdrvr/

Tape Libraries

Table 16. Tape Library Requirements for HP, Sun, and Windows

Tape Library	Model/PID Number	Type of Update
TS3100 (LTO-4)	3573-L2U	For firmware update, http://www.ibm.com/servers/storage/support/lto to select your library, then click the **Download** tab.
TS3200 (LTO-4)	3573-L4U	
TS3310 (LTO-4)	3576-L5B, E9U	
TS3400 (TS1120)	3577-L5U	
TS3500 (TS1120 or LTO-4)	3584-L22, L23, L32, L52, L53, D22, D23, D32, D52, D53	
3494 (TS1120)	3494-L22, D22	Firmware update FC 0520

Tape Drive

Table 17. Tape Drive Requirements for HP, Sun, and Windows

Tape Drive	Model/PID Number	Type of Update
TS1120	3592-E05 New drive order	Order 3592-E05 FC 9592 (plant) or FC 5592 (field). When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order 3592-E05 FC 9596 or FC 5596.
	3592-E05 Field upgrade for encryption	
TS1130	3592-E06	All E06 and EU6 drives are encryption-capable. When library-managed encryption is used, order 3584 Lxx FC 9900. For customer-enabled encryption via the library no other features required. For IBM CE setup of encryption drives, order FC 9596 or FC 5596 for the 3592 E06 or EU6.
	3592-EU6	
TS1040	3588-F4A	For system-managed and library-managed encryption, order 3584 Lxx FC 1604. No features required for application-managed encryption.
TS2340	3580-L43, 3580-S43	No features required for application-managed encryption.

How to Obtain IBM Encryption Key Manager Java Code for Windows, HP-UX, and Sun Solaris Operating Systems

The Encryption Key Manager consists of four parts:

- The Encryption Key Manager application that runs on IBM Java. This is included in the procedure that follows and is also available for download at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.
- Services and Sample files - also available at the website above.
- Unrestricted policy files. Available for download at https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk.
- IBM Java. Follow the procedure below for Windows, HP-UX, and Sun Solaris. For other operating systems download from links available through http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

1. Obtain IBM ID and password
   - You will need an IBM ID and password to download code. If you have not already registered for these, do this first. Downloads may not be available until ID registration is processed, which could take 24 hours.
   - After your registration request is submitted, a SITE NUMBER may be displayed. SAVE THIS SITE NUMBER IMMEDIATELY. You may need it the first time you logon. If you lose it you will have to wait to logon and download your software until the number is mailed to you, which could take 24 hours or longer.
   - To register for your IBM ID:

   a. Go to https://www.ibm.com/software/howtobuy/passportadvantage/paocustomer/LoginPage?abu=
   b. Click **register** in the text above the empty IBM ID box.
   c. Follow prompt to **Submit** and SAVE THE SITE NUMBER if one is displayed.
2. Purchase IBM TotalStorage Productivity Center Basic Edition
   - If you do not already have an IBM ID and password registered, you may obtain one as described above. Downloads may not be available until ID registration is processed, which could take 24 hours.
   - Purchase IBM TotalStorage Productivity Center Basic Edition (TPC BE), which includes the Encryption Key Manager as part of the package.
     a. Go to: http://www.ibm.com/software/info/app/ecatalog/index.html.
     b. Select Country and Language, then click **Go**.
     c. Click **T**, then scroll to and click **TotalStorage Productivity Center**.
     d. Click **View Pricing**, then scroll to bottom and click checkbox for **IBM TotalStorage Productivity Center Basic Edition**.
     e. Click **Add to Cart** and then **Checkout**.
     f. At this point you are asked to sign in with your IBM ID or to register. Sign in using your previously obtained ID and password (and possibly Site#) or register for an ID by following prompts. Click **Continue**.
     g. In the *Review and Submit Order* window, click **Agree** and then **Submit**.
     h. In the *Order Confirmation and Information* window, locate and save the Order Reference #.
3. Download IBM TotalStorage Productivity Center Basic Edition
   - In the *Order Confirmation and Information* window:
     a. Click **Shopping Cart** at left.
     b. Click **Order Status**.
     c. Click **Download** (you may have to wait until the order processes further to do this).
   - If the Shopping Cart/Order Status is not available, do the following:
     a. Go to: http://www.ibm.com/software/info/app/ecatalog/index.html.
     b. Select Country and Language, then click **Go**.
     c. Click **How to Buy**, then click **Customer Sign In** .
     d. Click **Download and Media Access** at left, then click **Order Status**.
     e. If your order is sufficiently processed click **Download now**.
   - Click **I Agree** and then click **Download** (you may have to wait until your order process further to do this).
4. In the *Download Step 1 of 2* window, select your language and click the checkbox of any platform of TPC Basic Edition (every platform contains the Encryption Key Manager for Windows, HP, and Solaris. Then click **Continue**.
5. In the *Download Step 2 of 2* window, uncheck the box for TotalStorage Productivity Center Basic Edition, then click the + to see Basic Edition downloads. Check the box next to the desired Encryption Key Manager downloads (near bottom of list). Scroll down and click **Agree** then **Download Now**.

Note: If you need live help at any point in the process, you can call the following numbers:
- PPA Hotline - 1-800-266-8270

- IBM Direct - for new customer, no purchase yet - 1-800-426-2255
- eCare - for customers who have already purchased via PPA - 1-800-978-2246

How to Find the Site Number

If you have misplaced the site number, you can find it as follows:

1. Go to Software Online Catalog website at http://www.ibm.com/software/info/app/ecatalog/index.html.
2. Click **Sign on to Passport Advantage Online** to get to https://www-112.ibm.com/software/howtobuy/passportadvantage/paocustomer/LoginPage?abu=.
3. Click **Account Management** at the left and your site number is displayed.

Keystore Considerations

It is impossible to overstate the importance of preserving your keystore data. Without access to your keystore you will be unable to decrypt your encrypted tapes. Carefully read the topics below to understand the methods available for protecting your keystore data.

The Importance of Keys and Certificates

The Encryption Key Manager and all its supported tape drives use symmetric, 256-bit AES keys to encrypt data. However, there are important differences between the way TS1120 or TS1130 tape drives and LTO Ultrium 4 tapes drives handle keys and certificates. The following discussion explains what you should know about these differences.

Encryption Keys and the TS1120 and TS1130 Tape Drives

In addition to 256-bit AES symmetric data keys, the Encryption Key Manager also uses public/private (*asymmetric*) key cryptography to protect the symmetric data encryption keys that are generated and retrieved as they pass between the Encryption Key Manager and tape drives. Public/private key cryptography is also used to verify the identity of the tape drives to which the Encryption Key Manager serves keys.

When a TS1120 or TS1130 tape drive requests a key, the Encryption Key Manager generates a random symmetric data encryption key (DK). Public/private key cryptography is used to *wrap* the data encryption key (DK) using a key encryption key (KEK), the public key of an asymmetric key pair. The *wrapped* data key, along with key label information about what private key is required to unwrap the symmetric key, forms a digital envelope called an Externally Encrypted Data Key (EEDK) structure. The EEDK is stored in the tape leader area of any tape cartridge that holds data encrypted using this method. In this way, the key used to decrypt the data is stored with the data on the tape itself, protected by asymmetric, public/private key wrapping. The public key used to wrap that data key is obtained from one of two sources:

- A certificate (from a business partner, for example) stored in the Encryption Key Manager's keystore, or
- A public key (part of an internally generated public/private key pair) stored in the Encryption Key Manager's keystore

The DK is *only* stored on the tape, in a wrapped, protected form.

When an encrypted tape is to be read by a TS1120 or TS1130 tape drive, the tape drive sends the EEDK to the Encryption Key Manager. The Encryption Key Manager determines from the alias or key label which private KEK from its keystore to use to unwrap the EEDK and recover the DK. Once the DK is recovered, it is then wrapped with a different key, which the tape drive can decrypt, and sent back to the tape drive, enabling the tape drive to decrypt the data.

The Encryption Key Manager uses aliases, also known as key labels, to identify the public/private keys used to wrap the EEDK when encrypting with TS1120 or TS1130 tape drives. Specific aliases may be defined for each tape device in the drive table. The Encryption Key Manager can also be set up to apply global default aliases (see “Global Default Alias (Key Label) for TS1120 and TS1130 Tape Drive Writes” on page 136). If your encryption-enabled tape drives are in a tape library, you can have the library define the key labels and pass them directly to the tape drive. If your encryption-enabled tape drives are in a z/OS environment, you can have the key labels defined through the host and passed directly to the tape drive. Refer to *z/OS DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drives (3592)* for information on specifying key labels at the host.

The Encryption Key Manager requires the definition of at least two aliases (certificates or key labels) for each encrypting tape drive in order to allow 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. If you do not wish to specify two different key labels or aliases, you can define both aliases with the same value. The Encryption Key Manager searches for two alias values in the following order:

1. From the system (for example, from DFSMS for system-managed encryption), from the library (for library-managed encryption), or from the application (for example, from Tivoli Storage Manager for application-managed encryption)
2. From the drive default alias (defined in the tape drive table)
3. From the global default alias (*drive.default.alias1* and *drive.default.alias2* in the configuration file

It is important to note that the Encryption Key Manager requires that two aliases or key labels be associated in pairs for each encrypting tape drive. Regardless whether you use default aliases defined in the drive table, global default aliases, or library-defined key labels, you must use them in pairs (#1 value and #2 value). Table 18 shows how aliases and key labels may be combined.

Table 18. Allowed alias and key label pairs. Define a minimum of one #1 and one #2.

App Key #1	Drive Default #1	Global Default #1	App Key #2	Drive Default #2	Global Default #2
X			X		
X				X	
X					X
	X		X		
	X			X	
	X				X
		X	X		

Table 18. Allowed alias and key label pairs (continued). Define a minimum of one #1 and one #2.

App Key #1	Drive Default #1	Global Default #1	App Key #2	Drive Default #2	Global Default #2
		X		X	
		X			X

Figure 7 shows how keys are processed for encrypted write operation.

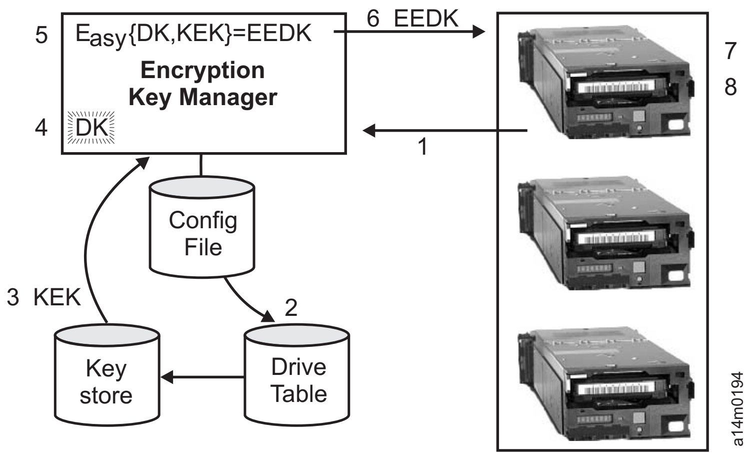


Figure 7. How the Encryption Key Manager Responds to TS1120 and TS1130 Tape Drive Requests for Encryption Write Operation

1. Tape drive requests key to encrypt tape
2. Encryption Key Manager verifies tape device in Drive Table
3. Encryption Key Manager fetches keys and certificates for tape device from keystore
4. Encryption Key Manager generates a random DK
5. Encryption Key Manager wraps DK with public key(s) to create an EEDK(s)
6. Encryption Key Manager sends the EEDK(s) and (separately wrapped) DK to the tape drive
7. Tape drive unwraps the DK and writes the EEDK(s) on tape leader
8. Tape drive encrypts data using DK and writes encrypted data to tape

Figure 8 on page 32 shows how keys are processed for encrypted read operation.

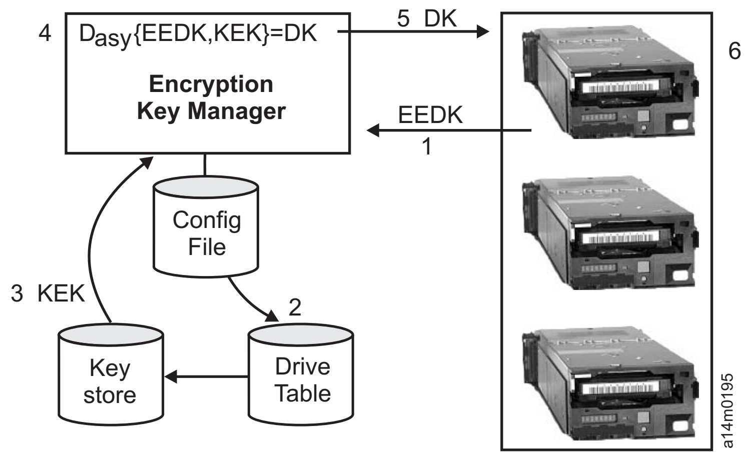


Figure 8. How the Encryption Key Manager Responds to TS1120 and TS1130 Tape Drive Requests for Encryption Read Operation

1. Tape drive receives read request and sends EEDK(s) to Encryption Key Manager
2. Encryption Key Manager verifies tape device in Drive Table
3. Encryption Key Manager fetches keys needed to process EEDK(s) from keystore
4. Encryption Key Manager unwraps EEDK using private key of KEK pair to recover DK
5. Encryption Key Manager wraps the DK with a key the drive can decrypt and sends the wrapped DK to tape drive
6. Tape drive unwraps the DK and uses it to decrypt the data or to perform a write-append

The certificates and keys stored in the Encryption Key Manager ’s keystore are the point of control allowing a tape drive or library to decrypt the data on the tape. This makes the information in the keystore vital; without it the tape cannot be read. Therefore, while it is very important that this information be protected so that others cannot obtain the private keys from the keystore, it is also crucial that this information always be available to you so that you can read the tapes whenever necessary.

Encryption Keys and the LTO Ultrium 4 Tape Drive

When performing encryption tasks on the LTO Ultrium 4 Tape Drive for LTO 4 tape cartridges, the Encryption Key Manager uses 256-bit AES symmetric data keys only.

When a LTO Ultrium 4 Tape Drive requests a key, the Encryption Key Manager uses the alias specified for the tape drive. If no alias was specified for the tape drive, an alias from a key group, key alias list, or range of key aliases specified in the symmetricKeySet configuration property is used. Lacking a specific alias for the tape drive, aliases are selected from the other entities in round robin fashion to balance the use of keys evenly.

The selected alias is associated with a symmetric Data Key (DK) that was preloaded in the keystore. The Encryption Key Manager sends this DK to the LTO 4 tape drive to encrypt the data. The selected alias is also converted to an entity called Data Key identifier (DKi), which is written to tape with the encrypted data.

In this way, the Encryption Key Manager can use the DKi to identify the correct DK needed to decrypt the data when the LTO 4 tape is read.

The **adddrive** and **moddrive** topics in "CLI Commands" on page 161 show how to specify an alias for a tape drive. See "Generating Keys and Aliases for Encryption on LTO 4" on page 94, which includes information on importing keys, exporting keys, and specifying default aliases in the symmetricKeySet configuration property. "Creating and Managing Key Groups" on page 100 shows how to define a key group and populate it with aliases from your keystore.

Figure 9 shows how keys are processed for encrypted write operation.

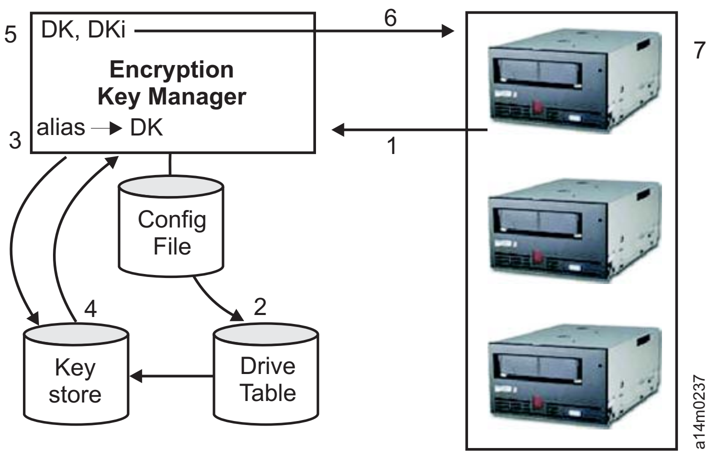


Figure 9. LTO Ultrium 4 Tape Drive Request for Encryption Write Operation

1. Tape drive requests key to encrypt tape
2. Encryption Key Manager verifies tape device in Drive Table
3. If no alias is specified in the request and no alias is specified in the drive table, the Encryption Key Manager selects an alias from the set of aliases or the key group in the keyAliasList
4. Encryption Key Manager fetches a corresponding DK from the keystore
5. Encryption Key Manager converts the alias to a DKi and wraps the DK with a key the drive can decrypt
6. Encryption Key Manager sends the DK and DKi to the tape drive
7. Tape drive unwraps the DK and writes encrypted data and DKi to tape

Figure 10 on page 34 shows how keys are processed for encrypted read operation.

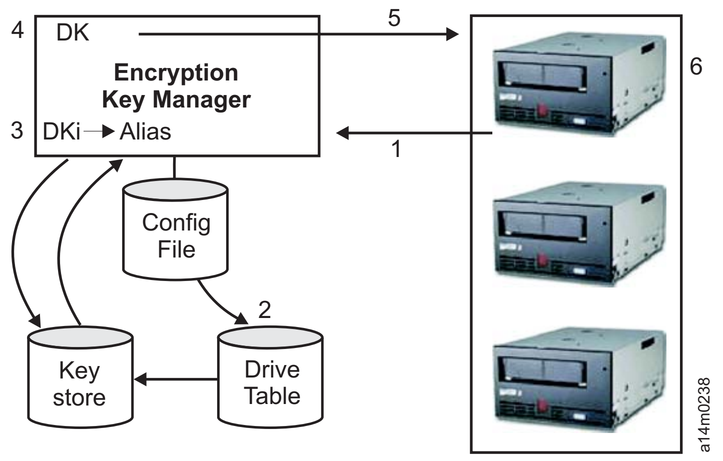


Figure 10. LTO Ultrium 4 Tape Drive Request for Encryption Read Operation

1. Tape drive receives read request and sends DKi to Encryption Key Manager
2. Encryption Key Manager verifies tape device in Drive Table
3. Encryption Key Manager translates DKi to alias and fetches corresponding DK from keystore
4. Encryption Key Manager wraps the DK with a key the drive can decrypt
5. Encryption Key Manager sends the wrapped DK to tape drive
6. Tape drive unwraps the DK and uses it to decrypt the data

Backing up Keystore Data

Note: Due to the critical nature of the keys in your keystore, it is vital that you back up this data on a non-encrypted device so that you can recover it as needed and be able to read the tapes that were encrypted using those certificates associated with that tape drive or library. Failure to backup your keystore properly will result in irrevocably losing all access to your encrypted data.

There are many ways to backup this keystore information. Each keystore type has it own unique characteristics. These are discussed in more detail in “Which Keystore is Right for You” on page 37 These general guidelines apply to all:

- Keep a copy of all certificates loaded into the keystore (usually a PKCS12 format file).
- Use system backup capabilities (such as RACF) to create a backup copy of the keystore information (be careful not to encrypt this copy using the encrypting tape drives as it would impossible to decrypt it for recovery).
- Maintain a primary and secondary Encryption Key Manager and keystore copy (for backup as well as failover redundancy).
- For a JCEKS keystore, simply copy the keystore file and store the clear (unencrypted) copy in a secure location such as a vault (be careful not to encrypt this copy using the encrypting tape drives as it would be impossible to decrypt it for recovery).

At a minimum, you should back up your keystore data whenever you change it. The Encryption Key Manager does not modify keystore data. The only changes to the keystore are those that you apply, so be sure to copy the keystore as soon as you change it.

Multiple Key Managers for Redundancy

The Encryption Key Manager is designed to work with tape libraries to allow redundancy, and thus high availability, so you can have multiple key managers servicing the same tape drives and libraries. Moreover, these key managers need not be on the same systems as the tape drives and libraries. The maximum number of key managers depends on your library or proxy. The only requirement is that they be available to the tape drives through TCP/IP connectivity.

This allows you to have two Encryption Key Managers that are mirror images of each other with built-in backup of the critical information about your keystores, as well as a failover in the event one key manager becomes unavailable. When you configure your device (or proxy) you can point it to two key managers. If one key manager becomes unavailable for any reason, your device (or proxy) will simply use the alternate key manager.

You also have the capability to keep the two Encryption Key Managers synchronized. It is critical that you take advantage of this important function when needed, both for its inherent backup of critical data and also for its failover capability to avoid any outages in your tape operations. Refer to "Synchronizing Data Between Two Key Manager Servers" on page 136.

Note: Synchronization does not include keystores. They must be copied manually.

Encryption Key Manager Server Configurations

The Encryption Key Manager may be installed on a single-server or on multiple servers. The following examples show one- and two-key manager configurations but your library may allow more.

Single-Server Configuration

A single-server configuration, shown in Figure 11, is the simplest Encryption Key Manager configuration. However, because of the lack of redundancy it is not recommended. In this configuration, all tape drives rely on a single key manager server with no backup. Should the server go down, the keystore, configuration file, KeyGroups.xml file, and drive table would be unavailable, making any encrypted tape unreadable. In a single-server configuration you must ensure that backup copies of the keystore, configuration file, KeyGroups.xml file, and drive table are maintained in a safe place, separate from the Encryption Key Manager, so its function can be rebuilt on a replacement server if the server copies are lost.

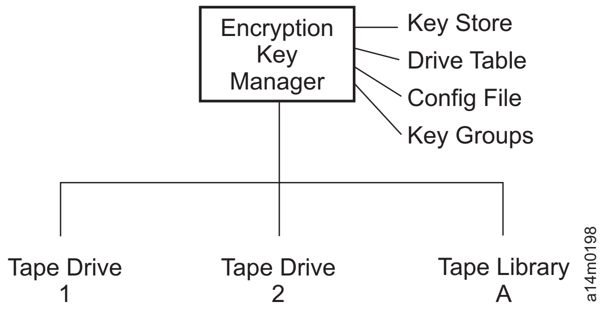


Figure 11. Single Server Configuration

Two-Server Configurations

A two-server configuration is recommended. This Encryption Key Manager configuration will automatically failover to the secondary key manager should the primary be inaccessible for any reason.

Note: When different Encryption Key Manager servers are used to handle requests from the same set of tape drives, the information in the associated keystores MUST be identical. This is required so that regardless which key manager server is contacted, the necessary information is available to support requests from the tape drives.

Identical configurations: In an environment with two Encryption Key Manager servers having identical configurations, such as those shown in Figure 12, processing will automatically failover to the secondary key manager should the primary go down. In such a configuration it is essential that the two key manager servers be synchronized. Updates to the configuration file and drive table of one key manager server can be duplicated on the other automatically using the **sync** command, but updates to one keystore must be copied to the other using methods specific to the keystore(s) being used. The keystores and key groups XML file must be copied manually. Refer to "Synchronizing Data Between Two Key Manager Servers" on page 136 for more information.

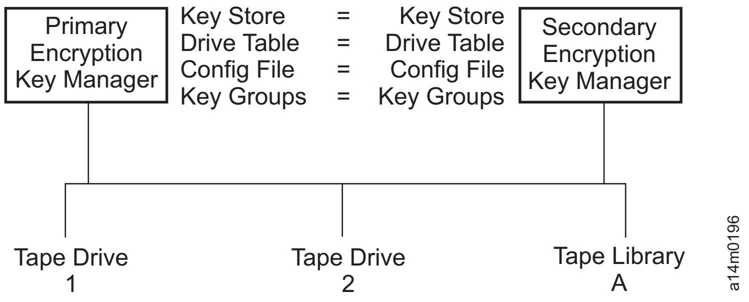

Primary Encryption Key Manager
Key Store = Key Store
Drive Table = Drive Table
Config File = Config File
Key Groups = Key Groups
Secondary Encryption Key Manager
Tape Drive 1
Tape Drive 2
Tape Library A
a14m0196

Figure 12. Two Servers with Shared Configurations

Separate configurations: Two Encryption Key Manager servers may share a common keystore and drive table yet have two different configuration files and two different sets of key groups defined in their XML files. The only requirement is that the keys used to serve the common tape drives must be the same for each server. This allows each key manager server to have its own set of properties. In this type of configuration, shown in Figure 13 on page 37, only the drive table should be synchronized between key manager servers. (Refer to "Synchronizing Data Between Two Key Manager Servers" on page 136 for more information.) Be sure to specify `sync.type = drivetab` (do not specify config or all) to prevent the configuration files from being overwritten.

Note: There is no way to partially share the configuration between servers.

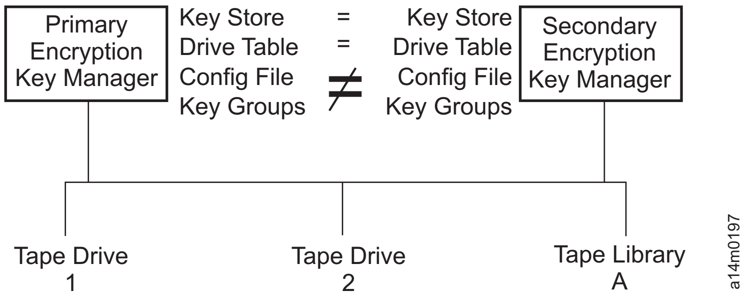


Figure 13. Two Servers with Different Configurations Accessing the Same Devices

Which Keystore is Right for You

The Encryption Key Manager uses standard and operating system-specific Java keystore methods to store the public/private key and certificate information. This information is needed to build and interpret EEDKs sent to and received from a tape drive or library, and used to write and read encrypted tapes. The Encryption Key Manager supports six keystore types. These keystore types are described to help you determine which is best for you.

| *Table 19. Summary of Supported Keystores*

Keystore	**Platform**	**TS1120 & TS1130 (store keypairs & certs)**	**LTO (store symmetric keys)**	**TS1120, TS1130, & LTO**	**Symmetric key tools available**
JCEKS	all	X	X	X	keytool
PKCS11ImplKS	open	X	X	X	keytool
IBMi5OSKeystore	i5/OS	X			keytool
JCE4758KS JCECCAKS	z/OS	X	X	X	keytool
JCERACFKS	z/OS	X			
JCE4758RACFKS JCECCARACFKS	z/OS	X			

z/OS Keystore Options

Throughout the course of this document you may see “JCE4758...” and “JCECCA...” used interchangeably. Both represent the same keystore type, however, “JCE4758...” must be used with SDK 1.4.2 (SDK 1.4.2 does not support “JCECCA...”). SDK 5.0 supports both “JCE4758...” and “JCECCA...” although “JCECCA...” is recommended.

The following keystore types are supported on z/OS:

- JCEKS (UNIX System Services file based)

  A file-based keystore supported on all platforms where the Encryption Key Manager runs. Thus it is relatively easy to copy the contents of this keystore for back up and recovery, and to keep two Encryption Key Manager instances synchronized for failover. JCEKS provides password-based protection of the contents of the keystore for security, and provides relatively good performance. File copy methods such as FTP may be used.

- JCE4758KS/JCECCAKS (Certificates in a file and keys may be protected by ICSF based on options chosen)

  A file-based keystore supported on the z/OS platform only. This keystore can be created and managed through the JVM hwkeytool command and is capable of creating ICSF key entries. Possible hwkeytool hardwarekeytype(s) that can be defined when creating your RSA key pair are:

  PKDS
  : Key resides in the PKDS and is protected by ICSF

  CLEAR
  : Key resides in a java keystore file. There is no ICSF security with using this option.

  Note: JCE4758KS/JCECCAKS does support symmetric keys, however, you must have installed SDK 142 sr9 or SDK 50 sr6 at a minimum. If you only use JCEKS on z/OS, the minimum SDK installation is 142sr8 or 50sr5.

- JCE4758RACFKS/JCECCARACFKS (Certificates stored in RACF and keys protected by ICSF)

  A SAF keyring/ICSF-based keystore supported on the z/OS platform only. This keystore can be created and managed through the RACDCERT or equivalent SAF certificate management command. This keystore uses certificates generated in RACF or SAF equivalent where the key material is stored in ICSF. The JCE4758RACFKS/JCECCARACFKS keystore makes use of all the security advantages of both RACF/SAF and ICSF.

  Note: This keystore type does not support symmetric keys. Therefore, to support LTO 4 tape drives, a JCEKS or JCE4758KS/JCECCAKS keystore must be used. If you are using only TS1120 or TS1130 tape drives, then a JCEKS or JCE4758RACFKS/JCECCARACFKS keystore can be used. If you are using only LTO 4 tape drives or using a combination of TS1120 or TS1130 and LTO 4 tape drives with the same tape library, then a JCEKS or JCE4758KS/JCECCAKS keystore **must** be used. If you are using both types of tape drives and you have a different tape library for each type, then you can have one Encryption Key Manager running with an JCE4758RACFKS/JCECCARACFKS keystore for the TS1120 or TS1130 tape drive/library and one Encryption Key Manager running with a JCEKS or JCE4758KS/JCECCAKS keystore for the LTO 4 tape drive/library. The two Encryption Key Manager servers can run on the same system if they listen on different ports.

- JCERACFKS (Certificates and key material stored in RACF)

  A SAF keyring-based keystore supported on the z/OS platform only. This keystore can be created and managed through the RACDCERT or equivalent SAF management command. This keystore uses certificates generated in RACF/SAF where the key material is stored in RACF/SAF. The JCERACFKS keystore makes use of all the security advantages of RACF/SAF while using software cryptography and software based security.

  Note: This keystore type does not support symmetric keys. Therefore, to support LTO 4 tape drives, a JCEKS or JCE4758KS/JCECCAKS keystore must be used. If you are using only TS1120 or TS1130 tape drives, then a JCEKS or JCERACFKS keystore can be used. If you are using only LTO 4 tape drives or using a combination of TS1120 or TS1130 and LTO 4 tape drives with the same tape library, then a JCEKS or JCE4758KS/JCECCAKS keystore **must** be used. If you are using both types of tape drives and you have a different tape library for each type, then you can have one Encryption Key Manager running with an JCERACFKS keystore for the TS1120 or TS1130 tape drive/library and one Encryption Key Manager running with a JCEKS or JCE4758KS/JCECCAKS keystore for the LTO 4 tape drive/library. The two Encryption Key Manager servers can run on the same system if they listen on different ports.

AIX, Linux, Solaris, HP-UX and Windows Keystore Options

On AIX, Linux, Solaris, HP-UX and Windows the following keystore types are supported:

- JCEKS (Unix System Services file based)

  A file-based keystore supported on all platforms where the Encryption Key Manager runs. Thus it is relatively easy to copy the contents of this keystore for back up and recovery, and to keep two Encryption Key Manager instances synchronized for failover. JCEKS provides password-based protection of the contents of the keystore for security, and provides relatively good performance. File copy methods such as FTP may be used.
- PKCS11IMPLKS (PKCS11 hardware crypto) (Not supported on HP-UX)

  Stores keys and certificates in a PKCS11 hardware cryptographic device. This enables the Encryption Key Manager to take advantage of the characteristics of these devices. These devices vary greatly but usually provide additional security and performance acceleration. A list of supported devices is available at http://www.ibm.com/developerworks/java/jdk/security/50/secguides/pkcs11implDocs/IBMPKCS11SupportList.html.

  Note: It is not always possible to make a backup of the certificates and keys once they are stored in a PKCS11 device. Some of these devices provide added security features that never allow this vital information to be retrieved in the clear. This makes the production of a backup copy in the event the device is destroyed more secure, however also more difficult. This is an important consideration in the selection process.

i5/OS Keystore Options

On the i5/OS platform, the supported keystores are:

- JCEKS (Unix System Services file based)

  A file-based keystore supported on all platforms where the Encryption Key Manager runs. Thus it is relatively easy to copy the contents of this keystore for back up and recovery, and to keep two Encryption Key Manager instances synchronized for failover. JCEKS provides password-based protection of the contents of the keystore for security, and provides relatively good performance. File copy methods such as FTP may be used.
- IBMi5OSKeystore

  A keystore that is created and managed using Digital Certificate Manager (DCM). The contents of the keystore are password protected for security. The format of this keystore is not compatible with other platforms (cannot FTP the entire keystore to another platform). However, the certificates in the keystore can be exported to a password protected file, then imported into another keystore type (such as JCEKS) on another platform in order to keep two Encryption Key Manager instances synchronized for failover. Certificates exported from another keystore type can also be imported into this type of keystore.

Note: This keystore type does not support symmetric keys. Therefore, to support LTO 4 tape drives on i5/OS, a JCEKS keystore must be used. If you are using only TS1120 or TS1130 tape drives, then either a JCEKS or IBMi5OSKeystore keystore can be used. If you are using only LTO 4 tape drives or using a combination of TS1120 or TS1130 and LTO 4 tape drives with the same tape library, then a JCEKS keystore **must** be used. If you are using both types of tape drives and you have a different tape library for each type, then you can have one Encryption Key Manager running with an IBMi5OSKeystore keystore for the TS1120 or TS1130 tape drive/library and one Encryption Key Manager running with a JCEKS keystore for the LTO 4 tape drive/library. The two Encryption Key Manager servers can run on the same system if they listen on different ports.

Managing Keystores

Once you have decided which keystore is best for your environment, you can create a new keystore. If you already have a keystore, you can import keys and certificates.

Keystore Passwords Must Not Be Longer than 127 Characters

Starting with build 20080530, the passwords for keystores in use by the Encryption Key Manager are restricted to 127 or fewer characters. Keystore passwords 128 characters or greater in length cause the obfuscation code to fail with a NegativeArraySizeException on Encryption Key Manager startup. This restriction is enforced as follows:

- If you are prompted for a password of fewer than 128 characters, startup does not proceed until you provide a password of acceptable length.
- In the event, the Encryption Key Manager is installed as a Windows Service and the keystore passwords are in the KeyManagerConfig.properties file and are 128 characters in length or greater, the Encryption Key Manager will fail to start because it has no way to prompt for a password of acceptable length. See “Log Entries for Keystore Passwords Greater than 127 Characters” on page 172 for more information on this failure.

If any keystores are already created with passwords 128 characters in length or greater, these keystore passwords can be changed using **keytool**. Keep in mind that the password cannot be changed only on the keystore itself, but must be changed for each key in the keystore. See “Changing Keystore Passwords” on page 96.

Managing Keystores on System z Platforms

For file-based (JCEKS) keystores
The standard Java tool for creating a JCEKS keystore and managing its keys and certificates is **keytool**. Visit http://www.ibm.com/developerworks/java/jdk/security/142/ for details on **keytool** usage.

For RACF keystores (keyrings) which may optionally use ICSF
The **RACDCERT** command is the interface used to create and manage keys, digital certificates, key rings, and digital certificate mappings in RACF. The RACDCERT command is documented and discussed in the *z/OS Security Server RACF Command Language Reference*. This publication can be found in the z/OS internet library at the URL: http://publibz.boulder.ibm.com/epubs/pdf/ichza460.pdf .

For ICSF-based keystores not using RACF

The java **hwkeytool** application creates and manages RSA keypairs and corresponding certificates without the use of RACF/SAF. If you also want to create symmetric keys for use with LTO 4 drives, you must use the commands in **keytool** (currently not supported in the **hwkeytool**). You can create the RSA key pairs in ICSF PKDS using **hwkeytool** and then use **keytool** against the same keystore to create the symmetric keys. See http://www.ibm.com/servers/eserver/zseries/software/java/hwkeytool.html for details.

Managing Keystores on Open Systems Platforms

For file-based (JCEKS) keystores

The **iKeyMan** utility with the **gsk7cmd** command provides a graphical user interface from which to manage the keys and certificates in a JCEKS keystore. See http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/security/50/GSK7c_SSL_IKM_Guide.pdf.

The standard Java tool, **keytool**, can also be used for creating a JCEKS keystore and managing its keys and certificates. Visit http://www.ibm.com/developerworks/java/jdk/security/142/ for details on **keytool** usage.

For PKCS11 keystores

Use the **iKeyMan** utility with the **gsk7cmd**. See http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/security/50/GSK7c_SSL_IKM_Guide.pdf.

For i5/OS keystores

The Digital Certificate Manager (DCM) GUI interface is the interface used to manage certificates when using the IBMi5OSkeystore option. See the Digital Certificate Manager topic in the iSeries information center for your version and release at http://publib.boulder.ibm.com/iseries/.

Disaster Recovery Site Considerations

If you plan to use a disaster recovery (DR) site, the Encryption Key Manager provides a number of options to enable that site to read and write encrypted tapes. These are:

- Create a duplicate Encryption Key Manager at the DR site.

  Set up a duplicate Encryption Key Manager at the DR site with the same information as your local Encryption Key Manager (configuration file, tape drive table, key groups XML file, and keystore). This key manager would then be in place and capable of taking over for one of your existing production key managers to read and write encrypted tapes.
- Create a backup copy of the three Encryption Key Manager data files to be able to recover as needed.

  If you create a current copy of the four data elements needed by the Encryption Key Manager (configuration file, tape drive table, key groups XML file, and keystore) then you would be able to start a key manager at any time to act as a duplicate at the DR site. (Remember that you should not use the Encryption Key Manager to encrypt the copies of these files as you would not be able to decrypt it without a functioning key manager). If your DR site uses different tape drives from your primary site, the configuration file and tape drive table must contain the correct information for the DR site.
- TS1120 or TS1130 tape drives may use the second EEDK on each tape to encrypt tapes such that a private key, which is unique to the DR site, is one of the entities that can read the encrypted tape. To do this, import the DR site's public key or simply copy their keystore. You could also choose to write an alternate

certificate for the DR site. This consists of using the certificate of the DR site to write your existing tapes in exactly the same way that you would provide this capability to another organization. In other words, in addition to storing your data encryption key on your tapes, wrapped using your organization's public/private key, the data key would also be stored on the same tapes wrapped using the DR site's public key (certificate). This allows a functioning Encryption Key Manager at that site to use its own keystore, with its own public/private key, to read the tapes. See "Considerations for Sharing Encrypted Tapes Offsite" for more information. If your DR site uses different tape drives from your primary site, the configuration file and tape drive table must contain the correct information for the DR site or the tape drive table will be of no use at the DR site.

i5/OS Disaster Recovery Considerations:

The i5/OS support will require the Encryption Key Manager server to be running on a different partition or system other than where the encrypted save is being performed. Failure to do so could result in data loss. Prior to recovering encrypted data, the Encryption Key Manager must be running or recovered on another system.

Maintaining primary and secondary Encryption Key Manager servers is desired for maximum availability of encrypted backup and recovery. The Encryption Key Manager and its associated data must be saved regularly without encryption. If the keystore password is specified on the strEKM script call (and not stored in the KeyManagerConfig.properties file), then you must keep a copy of the password in a secure location. The keystore password must be available to recover the Encryption Key Manager.

Encrypted save or archive operations must not be performed on the partition or system where the Encryption Key Manager server is running. If data on the system where the Encryption Key Manager is running is encrypted, the Encryption Key Manager cannot be recovered without availability of a secondary Encryption Key Manager server.

For additional disaster recovery information, refer to this Software Knowledge Base document: http://www-912.ibm.com/8625680A007CA5C6/1AC66549A21402188625680B0002037E/99ECAE271619B371862571CC0006652D.

Considerations for Sharing Encrypted Tapes Offsite

It is common practice to share tapes with other organizations for data transfer, joint development, contracting services, or other purposes. The methods for
| sharing encrypted tapes differ for TS1120 (or TS1130) and LTO 4 tapes.

Note: It is important to verify the validity of any certificate received from a business partner by checking the chain of trust of such a certificate back to the Certificate Authority (CA) that ultimately signed it. If you trust the CA, then you can trust that certificate. Alternately, the validity of a certificate can be verified if it was securely guarded in transit. Failure to verify a certificate's validity in one of these ways may open the door to a "Man-in-the-Middle" attack.

Sharing TS1120 or TS1130 Tapes

The Encryption Key Manager can store two sets of wrapped encryption keys on the IBM TotalStorage Enterprise Tape Cartridge 3592. This allows another organization to read that specific tape without you providing them any shared secret information or compromising the security of your certificates and keys.

This is done by adding the public part of the other organization's public/private certificate and keys to your Encryption Key Manager's keystore using a second alias (or key label). When the tape is written the encryption keys are stored on the tape, protected by two sets of public/private keys, yours and the other organization's. The other organization must have an encryption-enabled TS1120 or TS1130 Tape Drive. If so, it is then able to use their Encryption Key Manager and their private key to unwrap the data key that allows them to read that specific tape.

To reiterate, your Encryption Key Manager must have the certificate of the partner organization. The other organization must have the associated private key in the keystore used by the other organization's Encryption Key Manager. This gives you the flexibility to make a specific tape readable by both your own, and another, organization. If you wish to take advantage of this capability you must add that other organization's certificate, which contains the public key, to your keystore. See "Encryption Keys and the TS1120 and TS1130 Tape Drives " on page 29 for more information.

Sharing LTO 4 Tape

In order to share encrypted data on a LTO 4 tape, a copy of the symmetric key used to encrypt the data on the tape must be made available to the other organization to enable them to read the tape. In order for the symmetric key to be shared, the other organization must share their public key with you. This public key will be used to wrap the symmetric key when it is exported from the Encryption Key Manager keystore using keytool (see "Exporting Data Keys Using Keytool -exportseckey " on page 97). When the other organization imports the symmetric key into their Encryption Key Manager keystore, it will be unwrapped using their corresponding private key (see "Importing Data Keys Using Keytool -importseckey " on page 97). This ensures that the symmetric key is safe in transit since only the holder of the private key is able to unwrap the symmetric key. With the symmetric key that was used to encrypt the data in their Encryption Key Manager keystore, the other organization will then be able to read the data on the tape.

Federal Information Processing Standard 140-2 Considerations

Federal Information Processing Standard 140-2 has become important now that the Federal government requires all its cryptographic providers to be FIPS 140 certified. This standard has also been adopted in a growing private sector community. The certification of cryptographic capabilities by a third party in accordance with government standards is felt to have increased value in this security-conscious world.

The Encryption Key Manager does not provide cryptographic capabilities itself and therefore does not require, nor is it allowed to obtain, FIPS 140-2 certification. However, the Encryption Key Manager takes advantage of the cryptographic capabilities of the IBM JVM in the IBM Java Cryptographic Extension component and allows the selection and use of the IBMJCEFIPS cryptographic provider, which has a FIPS 140-2 level 1 certification. By setting the **fips** configuration parameter to **on** in the Configuration Properties file, you make the Encryption Key Manager use the IBMJCEFIPS provider for all cryptographic functions.

Note: You should not use hardware-based keystore types when the fips parameter is set on.

You can find more information on the IBMJCEFIPS provider and its selection and use at http://www.ibm.com/developerworks/java/jdk/security/50/FIPShowto.html.

See the documentation from specific hardware and software cryptographic providers for information on whether their products are FIPS 140-2 certified.

Chapter 3. Installing the Encryption Key Manager and Keystores

The Encryption Key Manager is shipped with the IBM Java Virtual Machine installation, and requires the IBM Software Developer Kit for Linux, and the IBM Runtime Environment for Windows (see "Hardware and Software Requirements" on page 18). If you are uncertain whether you have the latest version of Encryption Key Manager, "Determining Which Version of Encryption Key Manager is Installed" on page 131 explains how to learn if a newer version is available. It is a good idea to get the latest version of Encryption Key Manager, which may not be in your Java installation.

⚠ The Encryption Key Manager performs the function of requesting the generation of encryption keys and passing those keys to the TS1120, TS1130, or LTO-4 tape drives. The key material, in wrapped (encrypted) form does reside in system memory during processing by the Encryption Key Manager. Note that the key material (in either wrapped or clear form) must be transferred without error to the appropriate tape drive so that data written on a cartridge may be recovered (decrypted). If for some reason the key material (in either wrapped or clear form) is corrupted due to a bit error in system memory, and that key material is used to write data to a cartridge, then the data written to that cartridge will be unrecoverable (not able to be decrypted). There are safeguards in place to make sure that such data errors do not occur. However, if the machine hosting the Encryption Key Manager is not using Error Correction Code (ECC) memory there remains a possibility that the key material may become corrupted while in system memory and the corruption could then cause data loss. Although the risk is slight, it is always recommended that machines hosting the Encryption Key Manager use ECC memory.

Installing on z/OS

This section is intended to give you instructions to setup your z/OS environment to run the Encryption Key Manager. This was discussed briefly in Chapter 2, "Planning Your Encryption Key Manager Environment," on page 13, but is discussed in detail here. There are many possible ways you can setup your Encryption Key Manager on z/OS. This section shows you how to setup keys for the four possible key types (JCEKS, JCE4758KS, JCERACFKS and JCE4758RACFKS) and how to run the Encryption Key Manager in a z/OS production mode.

Install Java SDK and Verify Version

If you have not done so already, you will need to install a Java SDK. The minimum Java levels required to run the Encryption Key Manager on the z/OS platform are Java SDK 1.4.2 Service Refresh (SR) 8 or Java SDK 5.0 Service Refresh (SR) 5. Table 20 on page 46 contains a link where you can install the latest java.

Note: The Encryption Key Manager is not currently supported using the z/OS 64 bit SDKs.

Table 20. Encryption Key Manager Minimum Software Requirements for z/OS

IBM Software Developer Kit	Model/PID Number
IBM 31-bit SDK for z/OS, Java 2 Technology, V1.4	5655-I56 (at the SDK 1.4.2 SR8 level or above)
IBM 31-bit SDK for z/OS, Java 2 Technology, V5.0 (z/OS 1.6 and above only)	5655-N98 (at the SDK 5.0 SR5 level or above)
Available at : http://www.ibm.com/servers/eserver/zseries/software/java	

Verify the Java Version

It is important that you verify you have the correct version of Java installed in order to use the Encryption Key Manager. To do this you will need to add the Java bin directory to your PATH, which can be done by using the USS export command as shown below. Replace the path with the location where your Java SDK was installed. Then issue the **java -version** command and expect to see results similar to those shown here:

SDK 1.4.2:

```
export PATH=/usr/lpp/java/J1.4/bin:$PATH
java -version
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2)
Classic VM (build 1.4.2, J2RE 1.4.2 IBM z/OS Persistent Reusable VM build cm142-20070329 (SR8)
(JIT enabled: jitc))
```

SDK 5.0:

```
export PATH=/usr/lpp/java/J5.0/bin:$PATH
java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-20070426 (SR5))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-20070426 (JIT enabled)
J9VM - 20070420_12448_bHdSMr
JIT - 20070419_1806_r8
GC - 200704_19)
JCL - 20070425
```

As you plan for production deployment of the Encryption Key Manager on z/OS, it is important to note that there is a difference in the cryptographic capabilities between 1.4.2 and 5.0 Java SDKs. Java 5.0 allows for stronger strength keys when the zOSCompatibility flag is set to `false`. These stronger strength keys can only reside in host storage in an unencrypted form and cannot be encrypted by the ICSF host master key. Both 1.4.2 and 5.0 support lesser strength keys when the zOSCompatibility flag is set to `true`. These lesser strength keys are eligible to be encrypted under the ICSF host master key if the flag requireHardwareProtectionForSymmetricKeys is set to `true`.

Another aspect to consider is which platforms and SDK levels will be used by your partners reading the tapes written from your z/OS system. This must be understood to ensure that the cryptographic capabilities of the reader are compatible with the SDK level that you have chosen for your z/OS Encryption Key Manager deployment. This is discussed further in "Configuring on z/OS" on page 142.

Copy the Unrestricted Policy Files

Regardless of which IBM SDK version you use, you must replace the US_export_policy.jar and local_policy.jar files in your $JAVA_HOME/lib/security directory with an unrestricted version of these files. These unrestricted policy files are required by the Encryption Key Manager in order to serve AES keys.

The preferred method to do this on z/OS is to copy the unrestricted policy files that are shipped in the z/OS Java SDK build under the jce demo directory. You only need to copy them to the **lib/security** directory as shown in this example:

```
cp /usr/lpp/java/J1.4/demo/jce/policy-files/unrestricted/*
/usr/lpp/java/J1.4/lib/security
```

Alternatively, the unrestricted policy files can be downloaded from the following website:https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk.

| Be sure to select the "Unrestricted JCE Policy files for SDK for all newer versions,
| Version 1.4.2+," which will work for Java 1.4.2 and higher (including Java 5.0 SDK
| and Java 6.0). Do not select the 1.4.1 version, which is not compatible.

Upgrade to the Latest Encryption Key Manager JAR

Once you have installed (or upgraded) your Java SDK, you may need to upgrade to the latest Encryption Key Manager. Please see "Downloading the Latest Version Encryption Key Manager" on page 132.

| **Note:** When upgrading from an Encryption Key Manager build earlier than
| 20070503 you must define the Audit.metadata.file.name property in the
| KeyManagerConfig.properties file. This is the name of the XML file where
| metadata is saved. This property is required to start versions of Encryption
| Key Manager with build date 20070503 (when metadata support was added)
| and later. See "Determining Which Version of Encryption Key Manager is
| Installed" on page 131. See Chapter 9, "Using Metadata," on page 199 for
| more information on the Audit.metadata.file.name property.

Add the Java Hardware Provider (Only if Using ICSF)

If you are not sure what keystore type you want to use, please read "Which Keystore is Right for You" on page 37. If you have decided to use a keystore of type JCE4758KS/JCECCAKS or JCE4758RACFKS/JCECCARACFKS to make use of the security advantages of ICSF, you must add the Java hardware provider.

Note: You may decide that for test purposes you want to start out by using the JCEKS software keystore which does not use ICSF and thus does not require you to add the Java hardware provider at this time.

To add the Java hardware provider, you must edit the **$JAVA_HOME/lib/security/java.security** file and add the hardware provider so that it is the second provider in the list as shown in the examples below. Be sure to change the security.provider.# so that the providers are listed in order from 1, 2, 3, and so on.

For SDK 1.4.2, add the IBMJCE4758 provider as you see here (note: IBMJCECCA is not supported in SDK 1.4.2):

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=com.ibm.jsse.IBMJSSEProvider
security.provider.2=com.ibm.crypto.hdwrCCA.provider.IBMJCE4758
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
```

For SDK 5.0 and higher, it is recommended that you add the IBMJCECCA provider as you see here:

```
#
# List of providers and their preference orders (see above):
#
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
```

For more detailed information on the java hardware provider, please visit this website: http://www.ibm.com/servers/eserver/zseries/software/java/jcecca14.html.

Set Up a User ID to Run the Encryption Key Manager

The Encryption Key Manager process on z/OS requires a z/OS user ID that identifies the Encryption Key Manager process to z/OS. For production deployments, it is recommended that the Encryption Key Manager be launched using the JZOS launcher and executed as a started task on z/OS (this is discussed later). In addition, the Encryption Key Manager must be able to retrieve the private key of your X.509 Digital Certificate(s) when servicing Tape Write and Read requests. For RACF type keystores (that is, JCERACFKS and JCE4758RACFKS) this means the user ID under which the Encryption Key Manager runs must be the owner of the certificate. You may wish to consult the *z/OS Security Server Security Administrator's Guide* for an explanation of the rules that govern access to private keys and certificates.

In all the following examples, the user ID of EKMSERV is used.

This user ID must have an OMVS segment with a UID and GID defined. The UID need not be zero and can be any value. The home directory in this user ID's OMVS segment will be where the Encryption Key Manager will be started. The user ID must also run the standard shell at login (/bin/sh), and be connected to a default group that has a GID. You may allow RACF to automatically assign the UID or explicitly define the UID. The EKMSERV userID is a protected user, meaning that it can not be logged on to. The use of italics indicates fields that you can customize in this example.

```
AU EKMSERV DFLTGRP(SYS1) OMVS(AUTOUID HOME(/u/ekmserv)PROGRAM(/bin/sh))
  NOPASSWORD NOOIDCARD
```

Obtaining Digital Certificates

Before starting your Encryption Key Manager, you will need to have at least one X.509 digital certificate (contains a public/private key pair) that will be used to protect the data encryption key that the Encryption Key Manager will create when encrypting data to tape. The use of certificates, their public key and the corresponding private key is discussed in "The Importance of Keys and Certificates" on page 29. The Encryption Key Manager allows for two digital certificate aliases to be defined per write request. There is a requirement that one of the two aliases/labels specified must have a private key in the Encryption Key Manager's keystore when the tape is created. This guarantees that the creator of the tape will be able to read the tape. The other label/alias could be a public key from a business partner, which they will be able to decrypt with their private key. In order to read an encrypted tape, the correct private key is needed.

There are two methods of setting up digital certificates:

- Creating your own public/private key pair and corresponding certificate to be used to write/encrypt to tape so that you can read/decrypt the tape at a later date.
- Obtaining a business partner's public key and corresponding certificate to be used to write/encrypt tapes that can be read/decrypted by your business partner.

Note: The Encryption Key Manager does not read certificates with NO-TRUST status. To verify the status with RACF, issue a RACDCERT LIST command to display the certificate. This pertains to ACF2 and other security products as well.

Creating Your Own Public/Private Key Pair and Corresponding Certificate

There are several ways to setup your own public/private key pair for use by the Encryption Key Manager:

Using Certificates You Already Have: You may already have certificates and their associated public/private keys that are suitable for using with Encryption Key Manager. These certificates may be used as long as they are one of the following:

- Existing certificate and corresponding public/private keys which can be exported and then imported into java JCEKS or JCE4758KS keystore (thus a 2nd copy of the key would exist in a keystore file or PKDS).
- Existing RACF keys that are connected to RACF key rings so they can be accessed by the Encryption Key Manager.

Note: Keys with no corresponding X.509 certificate cannot be used with the Encryption Key Manager(for example, an ICSF public/private key that was created using ICSF tools does not have a corresponding certificate).

Generate a New Public/Private Key Pair and Corresponding Certificate: You may wish to generate a new public/private key pair and corresponding certificate to be used exclusively for processing and protecting your data on tape. This could be a self-signed certificate, a certificate signed using an internal z/OS Certificate Authority, or a certificate signed by a third party Certificate Authority such as VeriSign.

There are several tools you can use to create the public/private key pair and certificate. The examples in this publication illustrate how to create keys using the Java keytool (using software encryption), hwkeytool (using z/OS cryptography provided by ICSF) and RACF's RACDCERT command (usings RACF and, optionally, ICSF).

Note: You may also use an equivalent z/OS security product other than the IBM z/OS Security Server RACF product. If you do, consult the publications associated with that product to find the functionally equivalent operations and steps with respect to the RACF RACDCERT commands shown in the following examples.

Obtain a Public Key and Corresponding Certificate From a Business Partner

You may want to exchange encrypted tapes with a business partner. In order to write encrypted tapes to be sent to a business partner, you must import a public key/certificate from your business partner (they will be able to read the encrypted tape with their corresponding private key). The reverse is also true. In order to have a business partner create encrypted tapes that you will be able to read, you must export a public key/certificate from one of your public/private key pairs. Then you will be able to read the encrypted tape with your private key. The following examples illustrate how you would export and import a public key/certificate using the various tools (java keytool, java hwkeytool and RACF's RACDCERT).

Examples of How to Set Up Digital Certificates

The following examples show how you can use the Java tools (automatically available to you in the Java installation) and the RACF RACDCERT command to setup digital certificates for use by Encryption Key Manager. The examples are organized by the keystore type you select to start the Encryption Key Manager (for more information see "Which Keystore is Right for You" on page 37).

- "Example 1: Using the Java Keytool and JCEKS on z/OS" (software keys)
- "Example 2: Using the JCE4758KS or JCECCAKS Keystore with the Java Hwkeytool on z/OS" on page 52 (ICSF keys not using RACF)
- "Example 3: Using the JCERACFKS or JCE4758RACFKS/JCECCARACFKS Keystore on z/OS" on page 55 (RACF keys which are not in ICSF)

Example 1: Using the Java Keytool and JCEKS on z/OS: If you have decided to use the JCEKS keystore which is a file-based keystore supported on all of the Encryption Key Manager supported platforms, it is relatively easy to copy the contents of this keystore for back up and recovery, and to keep two Encryption Key Manager instances synchronized for failover. JCEKS provides password-based protection of the contents of the keystore and provides relatively good performance. When used on z/OS, the keystore is protected by the z/OS file system and z/OS security product. The Java keytool provides management of the JCEKS based keystore and its contents, such as the private keys and their associated X.509 certificates, and the certificate chains that authenticate the authenticity of a certificate. For more information on the java keytool, please see "Managing Keystores on System z Platforms" on page 40.

Verify Java is in Your Path

In order to use the keytool command you will need to make sure Java is in your path. See "Verify the Java Version" on page 46.

Generate a Public/Private Key and Corresponding Certificate for Your Encryption Key Manager

This example generates a public/private key pair and associated self-signed certificate, which is stored in a keystore. The resulting keystore can be used by your Encryption Key Manager to Write and Read encrypted tapes. You may wish to use this key to define a default alias in your **KeyManagerConfig.properties** file, (that is, drive.default.alias1 = MyEKMServerJCEKS).

Alternately, a certificate request can be generated and submitted to a third party certificate authority (or internal z/OS certificate authority). The fulfilled certificate and the certificate authority certificate can be imported back into a JCEKS keystore for use by the Encryption Key Manager.

For information on using the Java keytool to generate a certificate request and import the results back onto your JCEKS keystore, see this website publication:

http://www-128.ibm.com/developerworks/java/jdk/security/142/secguides/keytoolDocs/KeyToolUserGuide-142.html#certreqCmd.

For information on how to use SSL to create an internal z/OS Certificate Authority refer to this publication: *z/OS Cryptographic Services System Secure Sockets Layer Programming* SC24-5901.

Generate an RSA private/public key pair and associated certificate. Note that the keypass and storepass values must be the same since the Encryption Key Manager allows you to specify only one password to the keystore (in the KeyManagerConfig.properties ″config.keystore.password =″) and does not provide a way to send a specific password in with each keylabel. In this example, the key alias is MyEKMServerJCEKS, the distinguished name of the subject is myCo, the keystore file name is EKMKeystore. The password used to protect this keystore is ′somesecretphrase′ and the expiration of the certificate will be 999 days. The key size generated is 2048 bits in length.

```
keytool -genkey -alias MyEKMServerJCEKS -dname "CN=myCo"
-keystore EKMKeystore -provider IBMJCE -keyalg RSA -keysize 2048
-keypass "somesecretphrase" -storepass "somesecretphrase"
-storetype JCEKS  -validity 999
```

List the contents of the keystore where the certificate was created

```
keytool -list -keystore EKMKeystore -storetype JCEKS
-storepass "somesecretphrase"
```

Exchange a Public Key/Certificate with a Business Partner

The following example shows how to export a certificate and public key to a business partner, and how the business partner would import it so that the business partner could write encrypted tapes that can be read by your Encryption Key Manager. In this example we are sending the public key and corresponding certificate (alias/label MyEKMServerJCEKS) that was created above for use by your Encryption Key Manager. Note that we only send the public key (not the private key) so there is no security compromise.

Export the self signed certificate & public key “MyEKMServerJCEKS” to a file called **ExportedPublicKey.cer**.

```
keytool -export -file ExportedPublicKey.cer -keystore EKMKeystore
-alias MyEKMServerJCEKS -storepass "somesecretphrase" -storetype JCEKS
-provider IBMJCE -keypass "somesecretphrase"
```

Print the newly created certificate file using the keytool printcert utility.

```
keytool -printcert -file ExportedPublicKey.cer –storetype JCEKS
```

Now you can send the **ExportedPublicKey.cer** file to your Business Partner and may possibly need to tell them the alias “MyEKMServerJCEKS” if they are not able to use an encoding mechanism of Public Key Hash. This is explained in more detail below.

Import the certificate and public key from your business partner

Note: The *alias* you specify on import must match the *alias* that was used by the business partner (in this example, MyEKMServerJCEKS) if you plan to specify an encoding mechanism of Label “L” when encrypting tapes. Optionally, you can specify an encoding mechanism of Public Key Hash “H” that will use a Hash value rather than the KeyLabel to identify the key. While Hash gives slightly less performance, it allows you to import a certificate/public key from a business partner without knowing the alias/KeyLabel the business partner used to create/export the key. In addition it gives you the freedom specify the label you want to use to identify your business partner's public key. Therefore using Public Key Hash would be the preferred method. An example of how the z/OS encoding mechanism is specified is included below.

Import the certificate and public key from the business partner contained in ExportedPublicKey.cer to the EKMKeystore with an alias CompanyXPublicKeyJCEKS. Note that in this example the imported alias of CompanyXPublicKeyJCEKS does not match the original business partner's alias of MyEKMServerJCEKS. This only works if you plan to specify an encoding mechanism of Public Key Hash "H" for this key as shown in the example following this import example. Otherwise the alias on the import command would need to be provided to you by your Business Partner.

```
keytool -import  -file ExportedPublicKey.cer -keystore EKMKeystore
-alias CompanyXPublicKeyJCEKS -storepass "somesecretphrase"
-storetype JCEKS -provider IBMJCE -keypass "somesecretphrase"
```

List the contents of the keystore the certificate was imported to:

```
keytool -list -keystore EKMKeystore -storetype JCEKS -storepass "somesecretphrase"
```

On z/OS the encoding key mechanism can be specified in the data class or JCL. The following is an example of how it would be specified in the JCL. As is shown in this example, it is recommended that you specify an encoding mechanism of Label "L" when defining your own Key (for slightly better performance). However, you should specify an encoding mechanism of Public Key Hash "H" when defining a business partner key.

```
//C02STRW1 JOB CONSOLE,
// MSGCLASS=H,MSGLEVEL=(1,1),CLASS=B,
// TIME=1440,REGION=2M
/*JOBPARM SYSAFF=*
//*
//* ENC KEY MASTER JOB
//*
//CREATE1 EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//SEQ001 DD DSN=TAPE.C02M5CX2.PC5.NOPOOL.C02STRS1.MASTER,
// KEYLABL1='MyEKMServerJCEKS',
// KEYENCD1=L,
// KEYLABL2='CompanyXPublicKeyJCEKS',
// KEYENCD2=H,
// LABEL=(1,SL),UNIT=C02M5CX2,DISP=(,CATLG),
// DCB=(DSORG=PS,RECFM=FB,LRECL=2048,BLKSIZE=6144)
//SYSIN DD *
DSD OUTPUT=(SEQ001)
FD NAME=A,STARTLOC=1,LENGTH=10,FORMAT=ZD,INDEX=1
FD NAME=B,STARTLOC=11,LENGTH=13,PICTURE=13,'PRIMER RECORD'
CREATE QUANTITY=25,FILL='Z',NAME=(A,B)
END
/*
```

For more Information on the Hash encoding mechanism, please refer to *z/OS DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drives (3592)*, SC26-7514.

Example 2: Using the JCE4758KS or JCECCAKS Keystore with the Java Hwkeytool on z/OS: If you have decided to use the JCE4758KS/JCECCAKS keystore, you may take advantage of ICSF Security, which is supported only on the z/OS platform. This keystore is a file-based keystore where the certificate and public key are stored in a file and the private key may be stored in ICSF depending on the option specified when the key is created or imported. For information on ICSF back up and recovery as well as Master Key management refer to the following publications: *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide*, and *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide*.

The Java hwkeytool provides management of the JCE4758KS/JCECCAKS based keystore. For more information on the java hwkeytool, please see "Managing Keystores on System z Platforms" on page 40.

Verify Java is in Your Path

In order to use the keytool command you will need to make sure Java is in your path. See "Verify the Java Version" on page 46.

ICSF Must be Started

The ICSF address space must be started for the key generation to be successful.

Generate a Public/Private Key and Corresponding Certificate for Your Encryption Key Manager

This example generates a public/private key pair and associated self-signed certificate where the private key is stored in the ICSF PKDS. The resulting keystore can be used by your Encryption Key Manager to Write and Read encrypted tapes. You may wish to use this key to define a default alias in your config file (that is, drive.default.alias1 = MyEKMServerJCE4758KS).

Alternately, a certificate request can be generated and submitted to a third party certificate authority (or internal z/OS certificate authority). The fulfilled certificate and the certificate authority certificate can be imported back into a JCE4758KS/JCECCAKS keystore for use by the Encryption Key Manager.

For information on using the Java keytool to generate a certificate request and import the results back onto your JCEKS keystore, see this website publication: http://www-128.ibm.com/developerworks/java/jdk/security/142/secguides/keytoolDocs/KeyToolUserGuide-142.html#certreqCmd.

For information on how to use SSL to create an internal z/OS Certificate Authority refer to this publication: *z/OS Cryptographic Services System Secure Sockets Layer Programming* SC24-5901.

Generate an RSA private/public key pair and associated certificate where the private key is stored in the ICSF PKDS by specifying the –hardwaretype PKDS option. Note that the default –hardwaretype (when no option is specified) is CLEAR, which will not store the private key in the ICSF PKDS but rather in the Java keystore file where the public key and certificate are being stored (in this example, EKMKeystore4758). The keypass and storepass values must be the same since THE Encryption Key Manager allows you to specify only one password to the keystore (in the KeyManagerConfig.properties ″config.keystore.password =″) and does not provide a way to send a specific password in with each keylabel. In this example, the key alias is MyEKMServerJCE4758KS, the distinguished name of the subject is myCo, the keystore file name is EKMKeystore4758. The password used to protect this keystore is ′somesecretphrase′ and the expiration of the certificate will be 999 days. The key size generated is 2048 bits in length.

```
hwkeytool -genkey  -alias MyEKMServerJCE4758KS  -dname "CN=myCo"
-keystore EKMKeystore4758 -provider IBMJCE4758  -keyalg RSA
-keysize 2048 -storetype JCE4758KS -keypass "somesecretphrase"
-storepass "somesecretphrase" -hardwaretype PKDS
```

List the contents of the keystore where the certificate was created.

```
hwkeytool -list  -keystore EKMKeystore4758 -storepass "somesecretphrase"
-storetype JCE4758KS -provider IBMJCE4758
```

Note: Specify the Correct provider and storetype for your Java SDK.

You may have to modify the **hwkeytool** commands above depending on which Java SDK you are using:

- You must use –provider IBMJCE4758 and –storetype JCE4758KS when using Java SDK 1.4.2. (at any SR level)
- It is recommended that you use –provider IBMJCECCA and –storetype JCECCAKS when using Java SDK 5.0 and higher (although specifying provider IBMJCE4758 and –storetype JCE4758KS is allowed).

Exchange a Public Key/Certificate with a Business Partner

The following example shows how to export a certificate and public key to a business partner and how the business partner would import it so that the business partner could write encrypted tapes that can be read by your Encryption Key Manager. In this example we are sending the public key and corresponding certificate (alias/label MyEKMServerJCE4758KS) that was created above for use by your Encryption Key Manager. Note that we only send the public key (not the private key) so there is no security compromise.

Export the self signed certificate & public key "MyEKMServerJCE4758KS" to a file called ExportedPublicKey.cer.

```
hwkeytool -export -file ExportedPublicKey.cer -keystore EKMKeystore4758
-alias MyEKMServerJCE4758KS -storepass "somesecretphrase"
-storetype JCE4758KS -provider IBMJCE4758 -keypass "somesecretphrase"
```

Print the newly created certificate file using the hwkeytool printcert utility.

```
hwkeytool -printcert -file ExportedPublicKey.cer –storetype JCE4758KS
```

Now you can send the **ExportedPublicKey.cer** file to your Business Partner and may possibly need to tell them the alias "MyEKMServerJCE4758KS" if they are not able to use an encoding mechanism of Public Key Hash. This is explained in more detail below.

Import the certificate and public key from your business partner:

Note: The *alias* you specify on import must match the *alias* that was used by the business partner (in this example, MyEKMServerJCE4758KS) if you plan to specify an encoding mechanism of Label "L" when encrypting tapes. Optionally, you can specify an encoding mechanism of Public Key Hash "H" which will use a Hash value rather than the KeyLabel to identify the key. While Hash gives slightly less performance, it allows you to import a certificate/public key from a business partner without knowing the alias/KeyLabel the business partner used to create/export the key. In addition it gives you the freedom specify the label you want to use to identify your business partner's public key. Therefore using Public Key

Hash would be the preferred method. An example of how the z/OS encoding mechanism is specified is included below.

Import the certificate and public key from the business partner contained in ExportedPublicKey.cer to the EKMKeystore4758 with an alias CompanyXPublicKeyJCE4758KS. Note that in this example the imported alias of CompanyXPublicKeyJCE4758KS does not match the original business partner's alias of MyEKMServerJCE4758KS. This only works if you plan to specify an encoding mechanism of Public Key Hash "H" for this key as shown in the example following this import example. Otherwise the alias on the import command would need to be provided to you by your Busines Partner.

```
hwkeytool -import -file ExportedPublicKey.cer -keystore EKMKeystore4758
-alias CompanyXPublicKeyJCE4758KS -storepass "somesecretphrase"
-storetype JCE4758KS -provider IBMJCE4758 -keypass "somesecretphrase"
```

List the contents of the keystore the certificate was imported to:

```
hwkeytool -list -keystore EKMKeystore4758 -storetype JCE4758KS
-storepass "somesecretphrase"
```

On z/OS the encoding key mechanism can be specified in the data class or JCL. The following is an example of how it would be specified in the JCL. As is shown in this example, it is recommended that you specify an encoding mechanism of Label "L" when defining your own Key (for slightly better performance). However, you should specify an encoding mechanism of Public Key Hash "H" when defining a business partner key.

```
//C02STRW1 JOB CONSOLE,
// MSGCLASS=H,MSGLEVEL=(1,1),CLASS=B,
// TIME=1440,REGION=2M
/*JOBPARM SYSAFF=*
//*
//* ENC KEY MASTER JOB
//*
//CREATE1 EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//SEQ001 DD DSN=TAPE.C02M5CX2.PC5.NOPOOL.C02STRS1.MASTER,
// KEYLABL1='MyEKMServerJCEKS',
// KEYENCD1=L,
// KEYLABL2='CompanyXPublicKeyJCEKS',
// KEYENCD2=H,
// LABEL=(1,SL),UNIT=C02M5CX2,DISP=(,CATLG),
// DCB=(DSORG=PS,RECFM=FB,LRECL=2048,BLKSIZE=6144)
//SYSIN DD *
DSD OUTPUT=(SEQ001)
FD NAME=A,STARTLOC=1,LENGTH=10,FORMAT=ZD,INDEX=1
FD NAME=B,STARTLOC=11,LENGTH=13,PICTURE=13,'PRIMER RECORD'
CREATE QUANTITY=25,FILL='Z',NAME=(A,B)
END
/*
```

| For more Information on the Hash encoding mechanism, please refer to the *z/OS*
| *DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drives*
| *(3592)*, SC26-7514 publication.

Example 3: Using the JCERACFKS or JCE4758RACFKS/JCECCARACFKS Keystore on z/OS: If you have decided to use the JCERACFKS or JCE4758RACFKS/JCECCARACFKS keystore you may take advantage of RACF or both RACF and ICSF Security, which is supported only on the z/OS platform. This is a keyring-based keystore where the certificate and public/private key pair are stored in RACF, or the public/private key pair may be stored in ICSF depending on the option specified when the key is created or imported.

The JCERACFKS uses SAF and RACF services to protect key material and certificates. The JCE4758RACFKS keystore uses SAF and RACF services with the addition of ICSF to protect certificates and key material. For SAF/RACF-stored key rings, the RACF RACDCERT command is the interface used to manage the keyring. The RACDCERT command is documented and discussed in the *z/OS Security Server RACF Command Language Reference* publication, which can be found in the z/OS internet library at the URL: http://publibz.boulder.ibm.com/epubs/pdf/ichza460.pdf.

For information on ICSF back up and recovery as well as Master Key management refer to the following publications:

- *z/OS Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide*
- *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide*

Note: RACF APAR OA13030 provides additional support to the RACDCERT command with respect to ICSF keys. If you intend to generate RSA keys using RACF to be used with JCE4758RACFKS or JCECCARACFKS keystores and your z/OS platform is z/OS 1.4 - 1.7, you should validate that the PTF for this APAR has been applied.

The following examples illustrate several ways to generate a public/private key pair and associated certificate where the public/private key pair is stored in the ICSF PKDS. The resulting certificate is then connected to a keyring which can be used by the Encryption Key Manager to Write and Read encrypted tapes. In addition, this example shows how to export the certificate and public key and import them so that they could be used by a business partner to Write encrypted tapes that can be read by your Encryption Key Manager.

Define a Keyring

The Encryption Key Manager requires you to define a keyring for the Encryption Key Manager Server user ID on z/OS when using the JCERACFKS or JCE4758RACFKS/JCE4758CCARACFKS keystore types. This is accomplished using the zOS RACF **RACDCERT** command as shown in the following example. **RACDCERT** creates the keyring with the name of EKMRing for the user ID EKMSERV.

It is assumed that this command is being issued by an administrator who has authority to issue the **RACDCERT** command. Consult the *z/OS Security Server RACF Command Language Reference* for additional information and a detailed explanation of the **RACDCERT** command and parameters. If you are using another z/OS security product, you must perform this task using the tooling that is appropriate for that security product.

```
RACDCERT ID(EKMSERV) ADDRING(EKMRing)
```

Ensure that the Encryption Key Manager server is authorized to read from its keyring, and is authorized to use the key ICSF key label. Verify that the required RACF FACILITY class profiles are defined, and if not, issue the **RDEFINE** commands as shown to define these profiles which protect the use of keyring functions:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(EKMSERV) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(EKMSERV) ACC(READ)
```

Generate a Certificate/RSA Key Pair

This example generates an RSA key pair and self-signed certificate, and stores the RSA keys in the ICSF PKDS. The ICSF address space must be started for the key generation to be successful. The PCICC parameter of the RACDCERT GENCERT command indicates to RACF that the crypto hardware is to be used in the creation of the key pair. The SIZE indicates the modulus to be used in the creation of key pairs. The key label for the PKDS that is specified on the PCICC parameter must follow the z/OS dataset naming conventions.

Note: The use of ICSF is optional. If you choose not to use ICSF, the key that you generate is saved in the RACF database and limited in strength to 1024 bits. This may be acceptable in a testing environment.

You may use the self-signed certificate as is, have it signed by a self-signed certificate authority certificate within RACF, or choose to export a certificate request and submit to a third party certificate authority if your installation uses a third party certificate authority. All three cases are illustrated in the examples below.

Use the RACDCERT **GENCERT** command to generate the RSA public and private key pair and create a self-signed certificate. In this example, the Encryption Key Manager instance on z/OS will be executing with a z/OS user ID of EKMSERV. The subject distinguished name in the certificate that identifies this Encryption Key Manager instance will have a common name of ITOperations, for the company MyCo, in the United States. The certificate will have a label associated with it of EKMServer to easily identity the certificate.

The RACDCERT **GENCERT** command has two possible keywords, **PCICC** (used in the following examples) and **ICSF**. It should be noted that both will store the private key in the PKDS of ICSF with the following differences:

PCICC keyword
ICSF subsystem must be operational and configured for PKA operations.

A PCI-class cryptographic coprocessor must be operational.

The private key generated with RSA algorithm and stored as an ICSF RSA Chinese Remainder Theorem (CRT) key token in the PKDS.

ICSF keyword
ICSF subsystem must be operational and configured for PKA operations.

The private key is generated with RSA algorithm and stored as an ICSF RSA Modulus-Exponent (ME) key token in the PKDS.

For additional information refer to the RACF and ICSF documentation mentioned earlier in this document.

1. **Generating a self-signed certificate**
   a. Generate an RSA key pair and certificate for the Encryption Key Manager server instance on z/OS. The following RACDCERT command illustrates that the certificate generated will use ICSF for private key storage. The key size is 2048 bits and the private key will be saved in the ICSF PKDS in an encrypted form.
   ```
   RACDCERT GENCERT SUBJECTSDN(CN('ITOperations')
   O('MyCo') C('US')) WITHLABEL('EKMServer')
   PCICC(ITOPS.EKM.CERT) SIZE(2048)
   ```

If you are not using ICSF omit the PCICC keyword and change the keysize to 1024.

Note: This certificate can be used as is as a self-signed certificate or submitted to a third party certificate provider for signing. See item 3 on page 59 for information on submitting to a third party certificate provider.

b. You may send this certificate to other business partners or sites within your enterprise so that this certificate that identifies the Encryption Key Manager instance on z/OS will be known to your partners. By using this self signed certificate your business partners or remote sites agree to trust this certificate. This certificate may be imported into the keystore that is being used by the Encryption Key Manager at your partner's locations. In order to send this certificate, you must EXPORT it to a dataset

```
RACDCERT EXPORT (LABEL('EKMServer'))
DSN('hlq.PUBKEY.S2048.ITOPS') FORMAT(CERTDER)
```

See section "Business Partner and Remote z/OS Systems " on page 60 for information on what the setup should be at the Business Partners site.

c. You must ensure that the Encryption Key Manager Server certificate is connected to the Encryption Key Manager's keyring. This example shows connecting the certificate that identifies the Encryption Key Manager server to the Encryption Key Manager keyring. You will need to modify these command examples to suit your installation's needs.

```
RACDCERT ID(EKMSERV) CONNECT(LABEL('EKMServer')RING(EKMRing))
```

d. If you are using ICSF, ensure that the Encryption Key Manager Server instance has RACF authority to the key label of the private key stored in the ICSF PKDS. Also be sure to refresh the in storage copies of the CSFKEYS Class profiles:

```
RDEFINE CSFKEYS ITOPS.EKM.CERT  UACC(NONE)
PERMIT ITOPS.EKM.CERT CLASS(CSFKEYS) ID(EKMSERV) ACCESS(READ)
SETROPTS RACLIST(CSFKEYS) GENERIC(CSFKEYS) REFRESH
```

2. **Generating a certificate signed by an Internal Certificate Authority**

   a. Generate a self-signed certificate authority certificate.

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('MyLocalzOSCA')O('MyCo')C('US'))
WITHLABEL('LocalRACF CA') PCICC(LOCAL.RACF.CERTAUTH) SIZE(2048)
```

   If you are not using ICSF omit the PCICC keyword and change the keysize to 1024.

   b. Generate an RSA key pair and certificate for the Encryption Key Manager server instance on z/OS. The following RACDCERT command illustrates that the certificate generated will use ICSF for private key storage, and it is signed with the local certificate authority certificate generated in step 1.

```
RACDCERT ID(EKMSERV) GENCERT SUBJECTSDN(CN('ITOperations')
O('MyCo') C('US')) WITHLABEL('EKMServer') PCICC(ITOPS.EKM.CERT)
SIZE(2048) SIGNWITH(CERTAUTH LABEL('LocalRACF CA'))
```

   If you are not using ICSF omit the PCICC keyword and change the keysize to 1024.

   c. You may send this certificate to other business partners or sites within your enterprise so that this certificate that identifies the Encryption Key Manager instance on z/OS will be known to your partners. You should validate with your business partners or remote sites that you trust a common certificate authority (CA) whether third party or self signed, depending on your business and security practices. This certificate may be imported into the keystore that is being used by the Encryption Key Manager at your partner's locations. In order to send this certificate, you must EXPORT it to a dataset.

```
RACDCERT EXPORT (LABEL('EKMServer')) DSN('hlq.PUBKEY.S2048.ITOPS')
FORMAT(CERTDER)
```

See "Business Partner and Remote z/OS Systems " on page 60 for information on what the setup should be at the Business Partners site.

d. You must ensure that the Encryption Key Manager Server certificate and its designated certificate authority certificate are connected to the Encryption Key Manager's keyring. These examples show connecting a certificate authority certificate, and connecting the certificate that identifies the Encryption Key Manager server to the Encryption Key Manager keyring. You will have to modify these command examples to suit your installation's needs.

```
RACDCERT ID(EKMSERV) CONNECT(CERTAUTH LABEL('LocalRACF CA') RING(EKMRing))
RACDCERT ID(EKMSERV) CONNECT(LABEL('EKMServer')RING(EKMRing))
```

e. If you are using ICSF, ensure that the Encryption Key Manager Server instance has RACF authority to the key label of the private key stored in the ICSF PKDS. Also be sure to refresh the in storage copies of the CSFKEYS Class profiles:

```
RDEFINE CSFKEYS ITOPS.EKM.CERT  UACC(NONE)
PERMIT ITOPS.EKM.CERT CLASS(CSFKEYS) ID(EKMSERV) ACCESS(READ)
SETROPTS RACLIST(CSFKEYS) GENERIC(CSFKEYS) REFRESH
```

3. **Generating a certificate signed by a third party certificate authority**

a. Generate a RSA key pair for the Encryption Key Manager server instance on z/OS. The following RACDCERT command illustrates that the certificate generated will use ICSF for private key storage.

```
RACDCERT ID(EKMSERV) GENCERT SUBJECTSDN(CN('ITOperations')
O('MyCo') C('US')) WITHLABEL('EKMServer') PCICC(ITOPS.EKM.CERT) SIZE(2048)
```

If you are not using ICSF omit the PCICC keyword and change the keysize to 1024. The certificate can be submitted to a third party certificate provider for signing.

b. Generate and save a certificate request to a dataset (hlq.PUBKEY.REQUEST.ITOPS)

```
RACDCERT GENREQ (LABEL('EKMServer')) DSN('hlq.PUBKEY.S2048.ITOPS')
```

c. Submit certificate request, hlq.PUBKEY.S2048.ITOPS to your certificate provider. The response you receive will be an X.509 certificate. This example assumes that the certificate you receive from your third party certificate authority will be saved in the dataset 'hlq.THIRD.PARTY.CERT' on z/OS. The contents of this dataset will be imported into RACF. Note that this dataset contains only the signed certificate that identifies the Encryption Key Manager instance running on z/OS, and perhaps the certificate authority certificate. The private key for the Encryption Key Manager certificate remains protected by either ICSF in the PKDS or RACF.

d. Receive the response into dataset 'hlq.THIRD.PARTY.CERT'.

e. Add the certificate to RACF

```
RACDCERT ADD('hlq.THIRD.PARTY.CERT') TRUST
WITHLABEL('EKMServer') ID(EKMSERV)
```

If the CA certificate is not contained in the dataset 'hlq.THIRD.PARTY.CERT' you will need to acquire the CA certificate that signed the Encryption Key Manager certificate from the External Certificate Authority and add it to RACF as a CERTAUTH.

f. You may send this certificate to other business partners or sites within your enterprise so that this certificate that identifies the Encryption Key Manager instance on z/OS will be known to your partners. You should validate with your business partners or remote sites that you trust a common certificate authority (CA) whether third party or self signed, depending on your business and security practices. This certificate may be imported into the keystore that is being used by the Encryption Key Manager at your partner's locations. In order to send this certificate, you must EXPORT it to a dataset

   ```
   RACDCERT EXPORT (LABEL('EKMServer'))
   DSN('hlq.PUBKEY.S2048.ITOPS') FORMAT(CERTDER)
   ```

   See "Business Partner and Remote z/OS Systems " for information on what the setup should be at the Business Partners site.

g. You must ensure that the Encryption Key Manager Server certificate and its designated certificate authority certificate are connected to the Encryption Key Manager's keyring. These examples show connecting a certificate authority certificate, and connecting the certificate that identifies the Encryption Key Manager server to the Encryption Key Manager keyring. You will need to modify these command examples to suit your installation's needs.

   ```
   RACDCERT ID(EKMSERV) CONNECT(CERTAUTH LABEL('External CA label') RING(EKMRing))

   RACDCERT ID(EKMSERV) CONNECT(LABEL('EKMServer')RING(EKMRing))
   ```

   See "Business Partner and Remote z/OS Systems " for information on what the setup should be at the Business Partners site.

h. If you are using ICSF, ensure that the Encryption Key Manager Server instance has RACF authority to the key label of the private key stored in the ICSF PKDS. Also be sure to refresh the in storage copies of the CSFKEYS Class profiles:

   ```
   RDEFINE CSFKEYS ITOPS.EKM.CERT  UACC(NONE)
   PERMIT ITOPS.EKM.CERT CLASS(CSFKEYS) ID(EKMSERV) ACCESS(READ)
   SETROPTS RACLIST(CSFKEYS) GENERIC(CSFKEYS) REFRESH
   ```

Business Partner and Remote z/OS Systems

Another z/OS business partner, for example, or perhaps a remote z/OS site within your business would import the certificate into a z/OS dataset and use the RACDCERT to add that certificate to RACF. The public key in the certificate may also be saved in the ICSF PKDS depending on the operands supplied to the RACDCERT command.

Note: The KeyLabel you specify on the WITHLABEL option of the **RACDCERT ADD** command must match the KeyLabel that was used when the certificate was created if you plan to specify an encoding mechanism of Label "L" when encrypting tapes. Optionally, you can specify an encoding mechanism of Public Key Hash "H", which will use a Hash value rather than the KeyLabel to identify the key. While Hash gives slightly less performance, it allows you to import a certificate/public key without knowing the KeyLabel that was used to create/export the key. In addition it gives you the freedom to specify the KeyLabel you want to use to identify your business partner's public key. Therefore using the Public Key Hash would be the preferred method. An example of how the z/OS encoding mechanism is specified is included below.

The following **RADCERT ADD** command imports the certificate and public key from the business partner contained in the 'dataset_containing_the_cert_received' with an alias CompanyXEKMServer. Note that in this example the imported alias of CompanyXEKMServer does not match the original business partner's alias of EKMServer. This only works if you plan to specify an encoding mechanism of Public Key Hash "H" for this key as shown in the example following this import example. Otherwise the alias on the import command would need to be provided to you by your Busines Partner.

```
RACDCERT ID(EKMServ) ADD('dataset_containing_the_cert_received')
TRUST WITHLABEL('CompanyXEKMServer') PCICC(companyX.EKMServ.cert)
```

If you are not using ICSF omit the PCICC keyword and change the keysize to 1024.

The WITHLABEL keyword associates a string or *friendly name* for the certificate being imported, and this name is used by Encryption Key Manager when accessing the certificate. See the *z/OS Security Server RACF Command Language Reference* for detailed discussion of the **RACDCERT** command.

On z/OS the encoding key mechanism can be specified in the data class or JCL. The following is an example of how it would be specified in the JCL. As is shown in this example, it is recommended that you specify an encoding mechanism of Label "L" when defining your own Key (for slightly better performance) but you should specify an encoding mechanism of Public Key Hash "H" when defining a Business Partner key.

```
//C02STRW1 JOB CONSOLE,
// MSGCLASS=H,MSGLEVEL=(1,1),CLASS=B,
// TIME=1440,REGION=2M
/*JOBPARM SYSAFF=*
//*
//* ENC KEY MASTER JOB
//*
//CREATE1 EXEC PGM=IEBDG
//SYSPRINT DD SYSOUT=*
//SEQ001 DD DSN=TAPE.C02M5CX2.PC5.NOPOOL.C02STRS1.MASTER,
// KEYLABL1='EKMServer',
// KEYENCD1=L,
// KEYLABL2='CompanyXEKMServer',
// KEYENCD2=H,
// LABEL=(1,SL),UNIT=C02M5CX2,DISP=(,CATLG),
// DCB=(DSORG=PS,RECFM=FB,LRECL=2048,BLKSIZE=6144)
//SYSIN DD *
DSD OUTPUT=(SEQ001)
FD NAME=A,STARTLOC=1,LENGTH=10,FORMAT=ZD,INDEX=1
FD NAME=B,STARTLOC=11,LENGTH=13,PICTURE=13,'PRIMER RECORD'
CREATE QUANTITY=25,FILL='Z',NAME=(A,B)
END
/*
```

| For more Information on the Hash encoding mechanism, please refer to the *z/OS*
| *DFSMS Software Support for IBM System Storage TS1130 and TS1120 Tape Drive*
| *(3592)*, SC26-7514 publication.

You will also need to ensure that this certificate is connected (or associated) with the Encryption Key Manager server's keyring. This is accomplished using the **RACDCERT** command as shown in the following example. This example assumes that the Encryption Key Manager keyring on this z/OS system is EKMRing, and that the z/OS user ID associated with the Encryption Key Manager process is EKMSERV.

Note: As this certificate contains only a public key, it is very important that the USAGE(CERTAUTH) option be used. If it is not specified, the Encryption Key Manager will not start as it believes that the certificate that was added should also contain a private key.

```
RACDCERT ID(EKMSERV) CONNECT(LABEL('CompanyXEKMServer')
RING(EKMRing) USAGE(CERTAUTH))

RACDCERT ID(EKMSERV) CONNECT(CERTAUTH LABEL('GENERATED CA Label FROM ADD')
RING(EKMRing))
```

On this *remote* z/OS system, ensure that the Encryption Key Manager server is authorized to read from its keyring, and authorized to use the key ICSF key label. Ensure that the required RACF FACILITY class profiles are defined. If not, issue the RDEFINE commands as shown below to define these profiles which protect the use of keyring functions:

```
RDEFINE FACILITY IRR.DIGTCERT.LIST UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)

PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY) ID(EKMSERV) ACC(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(EKMSERV) ACC(READ)
```

If using ICSF, ensure that the Encryption Key Manager Server instance has RACF authority to the key label of the private key stored in the ICSF PKDS. Also, issue refresh for the in storage copies of the CSFKEYS Class profiles:

```
RDEFINE CSFKEYS REMOTE.EKM.CERT  UACC(NONE)
PERMIT REMOTE.EKM.CERT CLASS(CSFKEYS) ID(EKMSERV) ACCESS(READ)
SETROPTS RACLIST(CSFKEYS) GENERIC(CSFKEYS) REFRESH
```

For information on how to use this certificate containing the public key when
| encrypting data to tape, see *z/OS DFSMS Software Support for IBM System Storage*
| *TS1130 and TS1120 Tape Drives (3592)*, SC26-7514.

Verify the Keyring Can be Accessed by the Java Keytool/Hwkeytool

Before starting the Encryption Key Manager you should validate that the keystore is accessible via the Java hwkeytool (for JCE4758RACFKS/ICSF keys) or Java keytool (for JCERACFKS/non-ICSF keys).

The following examples shows how you would use both **hwkeytool** and **keytool** to list the contents of the keyring.

For ICSF keys:

```
hwkeytool -debug -J-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider
-list -keystore safkeyring://EKMSERV/EKMRing -storetype JCE4758RACFKS
```

For keys not in ICSF:

```
keytool -debug -J-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
-list -keystore safkeyring://EKMSERV/EKMRing -storetype JCERACFKS
```

Creating Symmetric Keys for Use with LTO 4 Drives

Although z/OS does not support LTO drives, you may wish to run your Encryption Key Manager on z/OS and allow an off-platform LTO drive to retrieve keys from your z/OS Encryption Key Manager. Your Encryption Key Manager can
| service TS1120, TS1130, and LTO 4 drives at the same time. However, unlike
| TS1120 or TS1130 drives, for LTO 4 drives you must manually create the symmetric

keys in your Encryption Key Manager keystore to be used for data encryption. See "How the Encryption Key Manager Processes Encryption Keys " on page 10 for an overview.

Symmetric keys are not supported by RACF so your Encryption Key Manager keystore must be of type JCEKS or JCE4758KS/JCECCAKS in order to create symmetric keys for use with LTO 4 drives. Also note that while the minimum required SDK installation for creating symmetric keys in a JCEKS type keystore is 142sr8/50sr5, it is 142sr10/50sr6 for a JCE4758KS/JCECCAKS type keystore.

For instructions and examples on creating, importing and exporting symmetric keys using the java **keytool**, see "Generating Keys and Aliases for Encryption on LTO 4" on page 94.

Sample Alias and Symmetric Key Setup for LTO 4 Encryption

```
/u/giampor/tkms:>cat populatesymmkeys.jce4758ks.sh
#
echo "Creating RSA Certificate and Public/Private KeyPair in ekm2sharedkeysjce4758ksPKDSlabel"
#
keytool -genkey -alias symmkeywrapper -dname "CN=sharedkeysjce4758ksSymmetricKeyWrapper" \
  -keystore ekm2sharedkeysjce4758ksPKDSlabel -provider IBMJCE4758 -keyalg RSA -keysize 2048 \
  -keypass "password" -storepass "password" -storetype JCE4758KS -validity 999
#
echo "List ekm2sharedkeysjce4758ksPKDSlabel"
keytool -list -keystore ekm2sharedkeysjce4758ksPKDSlabel -storepass "password" \
 -storetype JCE4758KS
#
#
echo "Exporting RSA Certificate/Public Key to ekm2sharedkeysjce4758ksPKDSlabelCA.crt"
#
keytool -export -alias symmkeywrapper -file ekm2sharedkeysjce4758ksPKDSlabelCA.crt \
  -keystore ekm2sharedkeysjce4758ksPKDSlabel -provider IBMJCE4758 -storepass "password" \
  -storetype JCE4758KS
#
#
echo "Creating Symmetric keys in symmkeystorejce4758ks"
#
keytool -genseckey -keystore symmkeystorejce4758ks -storetype JCE4758KS \
  -storepass "symmpassword" -aliasrange ibm01-05 -keyAlg DESede
#
echo "Listing Symmetric keys created"
#
keytool -list -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
#
echo "import public keys from keystores who want a copy of symmetric keys using a different
 alias - in this case ekm2sharedkeysjce4758ksPKDSlabelCA.crt from
 ekm2sharedkeysjce4758ksPKDSlabel"
#
keytool -import -trustcacerts -alias sharedkeysjce4758ksCA \
 -file ekm2sharedkeysjce4758ksPKDSlabelCA.crt \
 -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
#
echo "Listing keystore with public key imported and Symmetric keys"
#
keytool -list -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
#
echo "Export the Symmetric keys from symmetrickeystore for
 sharedkeysjce4758ksCA/ekm2sharedkeysjce4758ksPKDSlabel"
#
keytool -exportseckey -aliasrange ibm01-05 -keyalias sharedkeysjce4758ksCA \
 -keystore symmkeystorejce4758ks \
 -storepass "symmpassword" -storetype JCE4758KS -keypass "symmpassword" \
 -exportfile symKeysexported.jce4758.cer
#
```

```
#
echo "Import the Symmetric keys into ekm2sharedkeysjce4758ksPKDSlabel
  - i.e., sharedkeysjce4758ksCA but assume must use my orignal alias symmkeywrapper
  or it won't know how to get the private key"
#
keytool -importseckey -keyalias symmkeywrapper -keypass "password" \
 -keystore ekm2sharedkeysjce4758ksPKDSlabel \
 -storepass "password" -storetype JCE4758KS -importfile symKeysexported.jce4758.cer
#
#
echo "list ekm2keystore containing RSA keypair and symmetric keys"
#
keytool -list -keystore ekm2sharedkeysjce4758ksPKDSlabel -storepass "password" \
 -storetype JCE4758KS
/u/giampor/tkms:>
```

Setting Up the Encryption Key Manager Keystore to Communicate with Tape Drives

The following is an example of a shell script you can create to setup a JCE4758KS
| type keystore with 15 symmetric keys that could be used to serve keys to TS1120,
| TS1130, and LTO 4 drives . Note that if using SDK 5.0 instead of 1.4.2, you can
replace ″4758″ with ″CCA″

```
/u/ekmserv/temp/symmkeytest:>cat createekmkeys.sh
##########################################
# Setup EKM Keystore
##########################################
echo "\nCreating the EKM Server Certificate and corresponding RSA Public/Private KeyPair in
 EKMKeystore4758 where private key is stored in ICSF PKDS. \n"#
hwkeytool -genkey -alias MyEKMServerJCE4758KS -dname "CN=MyCo EKM Server" -keystore EKMKeystore4758 \
 -provider IBMJCE4758 -keyalg RSA -keysize 2048 -keypass "ekmpassword" -storepass "ekmpassword" \
 -storetype JCE4758KS -hardwaretype PKDS -validity 999
#
echo "\nCreating a Certificate and correpsonding RSA Public/Private Keypair to be used to
 wrap symmetric keys that will be imported into EKMKeystore4758.
 Private key is also stored in ICSF PKDS. \n"
#
hwkeytool -genkey -alias symmkeywrapper -dname "CN=EKM Symmetric Key Wrapper" \
 -keystore EKMKeystore4758 -provider IBMJCE4758 -keyalg RSA -keysize 2048 -keypass "ekmpassword" \
 -storepass "ekmpassword" -storetype JCE4758KS -hardwaretype PKDS -validity 999
#
echo "\nList contents of the EKMKeystore4758. \n"
keytool -list -keystore EKMKeystore4758 -storepass "ekmpassword" -storetype JCE4758KS
#
echo "\nExporting the Symmetric Key Wrapper Certificate/Public Key to myCoSymmKeyWrapper.crt. \n"
#
keytool -export -alias symmkeywrapper -file myCoSymmKeyWrapper.crt -keystore EKMKeystore4758 \
 -provider IBMJCE4758 -storepass "ekmpassword" -storetype JCE4758KS
#
##################################################################################################
# Create symmetric keys
#
# This example creates 15 symmetric keys in a separate keystore and exports them to a
# file to be imported into the EKM Keystore.
# Optionally the symmetric keys can be created right into the EKM keystore.
# This example is to show the various symmetric/secure key commands.
# Note that only the keytool can be used (not hwkeytool), thus the there is no option to
# specify -hardwaretype PKDS to store the symmetric keys in ICSF but rather the symmetric
# keys will be stored in a password protected keystore.
# Also note that the -keyAlg must be DESede if specifying -storetype JCE4758KS/JCECCAKS or
# planning to exchange tapes with a z/OS EKM defined with that storetype
# (i.e., zOSCompatibility flag = true).  For DESede, the -keysize default is 168.
# For -storetype JCEKS, -keyAlg should be AES and -keysize 256
# (unless zOSCompatibility flag is set to true).
#################################################################################################
#
echo "\nCreating 15 Symmetric keys in symmkeystorejce4758ks. \n"
#
keytool -genseckey -keystore symmkeystorejce4758ks -storetype JCE4758KS -storepass "symmpassword" \
 -keypass "symmpassword" -aliasrange ibm01-0F -keyAlg DESede
#
echo "\nListing Symmetric keys created. \n"
#
keytool -list  -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
echo "\nImport public keys from keystores who want a copy of symmetric keys..in this case alias
 symmkeywrapper which was exported from EMKeystore4758 into myCoSymmKeyWrapper.crt.  Note that I
 can change the alias on import and that by specifying noprompt the certificate will be imported
 as a trusted certificate. \n"
#
keytool -import -noprompt -alias ExternalCoCA -file myCoSymmKeyWrapper.crt \
 -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
echo "\nListing keystore with public key imported and Symmetric keys. \n"
#
keytool -list -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS
#
echo "\nExport the Symmetric keys from the symmetric keystore for ExternalCoCA
 (i.e., symmkeywrapper in EKMKeystore4758). \n"
#
keytool  -exportseckey -aliasrange ibm01-0F -keyalias ExternalCoCA \
 -keystore symmkeystorejce4758ks -storepass "symmpassword" -storetype JCE4758KS \
 -keypass "symmpassword" -exportfile symKeysexported.ExternalCoCA.cer
#
###############################################################################################
# Import the symmetric keys into the EKMKeystore4758.  Again note that there is currently no
# option to specify the keys to go into the ICSF PKDS but rather the keys will be stored in the
# password protected keystore.
###############################################################################################
echo "\nImport the Symmetric keys into EKMKeystore4758.  Note that you must use the correct
 alias, in this case 'symmkeywrapper' or keytool won't know how to find the private key. \n"
#
keytool  -importseckey -keyalias symmkeywrapper -keypass "ekmpassword" -keystore EKMKeystore4758 \
 -storepass "ekmpassword" -storetype JCE4758KS -importfile symKeysexported.ExternalCoCA.cer
#
echo "\nList EKMKeystore4758 containing 2 RSA keypairs and 15 imported symmetric keys.\n"
#
keytool -list -keystore EKMKeystore4758  -storepass "ekmpassword"  -storetype JCE4758KS
```

Sample Output from Shell Script

The following is an example of the output of running the above shell script:

```
/u/ekmserv/temp/symmkeytest:>. createekmkeys.sh

Creating the EKM Server Certificate and corresponding RSA Public/Private KeyPair in
EKMKeystore4758 where private key is stored in ICSF PKDS.


Creating a Certificate and correpsonding RSA Public/Private Keypair to be used
to wrap symmetric keys that will be imported into EKMKeystore4758.  Private key
is also stored in ICSF PKDS.


List contents of the EKMKeystore4758.


Keystore type: JCE4758KS
Keystore provider: IBMJCE4758

Your keystore contains 2 entries

symmkeywrapper, Jul 31, 2007, keyEntry,
Certificate fingerprint (MD5): 89:E0:0A:32:A7:B4:A4:F1:F6:4D:B9:F5:68:69:91:C3
myekmserverjce4758ks, Jul 31, 2007, keyEntry,
Certificate fingerprint (MD5): 16:B1:94:79:C1:C6:77:C5:6E:84:99:5C:D1:88:9E:65

Exporting the Symmetric Key Wrapper Certificate/Public Key to myCoSymmKeyWrapper.crt.
```

```
Certificate stored in file <myCoSymmKeyWrapper.crt>

Creating 15 Symmetric keys in symmkeystorejce4758ks.

KeyTool is generating batch keys. This process will take a while, be patient ...
15 secret keys have been generated

Listing Symmetric keys created.


Keystore type: JCE4758KS
Keystore provider: IBMJCE4758

Your keystore contains 15 entries

ibm000000000000000008, Jul 31, 2007, keyEntry,
ibm00000000000000000f, Jul 31, 2007, keyEntry,
ibm000000000000000007, Jul 31, 2007, keyEntry,
ibm00000000000000000e, Jul 31, 2007, keyEntry,
ibm000000000000000006, Jul 31, 2007, keyEntry,
ibm00000000000000000d, Jul 31, 2007, keyEntry,
ibm000000000000000005, Jul 31, 2007, keyEntry,
ibm00000000000000000c, Jul 31, 2007, keyEntry,
ibm000000000000000004, Jul 31, 2007, keyEntry,
ibm00000000000000000b, Jul 31, 2007, keyEntry,
ibm000000000000000003, Jul 31, 2007, keyEntry,
ibm00000000000000000a, Jul 31, 2007, keyEntry,
ibm000000000000000002, Jul 31, 2007, keyEntry,
ibm000000000000000001, Jul 31, 2007, keyEntry,
ibm000000000000000009, Jul 31, 2007, keyEntry,

Import public keys from keystores who want a copy of symmetric keys..in this case alias
symmkeywrapper which was exported from EMKeystore4758 into myCoSymmKeyWrapper.crt.
Note that I can change the alias on import and that by specifying noprompt the certificate
will be imported as a trusted certificate.

Certificate was added to keystore

Listing keystore with public key imported and Symmetric keys.


Keystore type: JCE4758KS
Keystore provider: IBMJCE4758

Your keystore contains 16 entries

ibm000000000000000008, Jul 31, 2007, keyEntry,
ibm00000000000000000f, Jul 31, 2007, keyEntry,
ibm000000000000000007, Jul 31, 2007, keyEntry,
ibm00000000000000000e, Jul 31, 2007, keyEntry,
ibm000000000000000006, Jul 31, 2007, keyEntry,
ibm00000000000000000d, Jul 31, 2007, keyEntry,
ibm000000000000000005, Jul 31, 2007, keyEntry,
ibm00000000000000000c, Jul 31, 2007, keyEntry,
ibm00000000000000000b, Jul 31, 2007, keyEntry,
ibm000000000000000004, Jul 31, 2007, keyEntry,
ibm00000000000000000a, Jul 31, 2007, keyEntry,
ibm000000000000000003, Jul 31, 2007, keyEntry,
ibm000000000000000002, Jul 31, 2007, keyEntry,
ibm000000000000000001, Jul 31, 2007, keyEntry,
externalcoca, Jul 31, 2007, trustedCertEntry,
Certificate fingerprint (MD5): 89:E0:0A:32:A7:B4:A4:F1:F6:4D:B9:F5:68:69:91:C3
ibm000000000000000009, Jul 31, 2007, keyEntry,

Export the Symmetric keys from the symmetric keystore for ExternalCoCA
 (i.e., symmkeywrapper in EKMKeystore4758).
```

```
15 secret keys have been successfully exported

Import the Symmetric keys into EKMKeystore4758.  Note that you must use the correct alias,
 in this case 'symmkeywrapper' or keytool won't know how to find the private key.

15 secret keys have been imported

List EKMKeystore4758 containing 2 RSA keypairs and 15 imported symmetric keys.


Keystore type: JCE4758KS
Keystore provider: IBMJCE4758

Your keystore contains 17 entries

ibm000000000000000008, Jul 31, 2007, keyEntry,
ibm00000000000000000f, Jul 31, 2007, keyEntry,
ibm000000000000000007, Jul 31, 2007, keyEntry,
ibm00000000000000000e, Jul 31, 2007, keyEntry,
ibm000000000000000006, Jul 31, 2007, keyEntry,
ibm00000000000000000d, Jul 31, 2007, keyEntry,
ibm000000000000000005, Jul 31, 2007, keyEntry,
ibm00000000000000000c, Jul 31, 2007, keyEntry,
ibm000000000000000004, Jul 31, 2007, keyEntry,
ibm00000000000000000b, Jul 31, 2007, keyEntry,
ibm000000000000000003, Jul 31, 2007, keyEntry,
ibm00000000000000000a, Jul 31, 2007, keyEntry,
ibm000000000000000002, Jul 31, 2007, keyEntry,
ibm000000000000000001, Jul 31, 2007, keyEntry,
symmkeywrapper, Jul 31, 2007, keyEntry,
Certificate fingerprint (MD5): 89:E0:0A:32:A7:B4:A4:F1:F6:4D:B9:F5:68:69:91:C3
myekmserverjce4758ks, Jul 31, 2007, keyEntry,
Certificate fingerprint (MD5): 16:B1:94:79:C1:C6:77:C5:6E:84:99:5C:D1:88:9E:65
ibm000000000000000009, Jul 31, 2007, keyEntry,
```

Set Up the Encryption Key Manager Configuration File

Before starting the Encryption Key Manager you must create your Encryption Key Manager's configuration file, which defines many parameters including which port the Encryption Key Manager will run on and the keystore or keyring where it can find the X.509 Digital Certificates. Please read Chapter 5, "Configuring the Encryption Key Manager," on page 135, including the section "Configuring on AIX and other Operating Systems" on page 149. You can find additional information in Appendix B, Default Configuration File.

Create Filesystem and Mountpoint to Contain Configuration Files

The configuration file is initially read by the Encryption Key Manager server upon startup and will be written back to the file when the Encryption Key Manager server is stopped. During the startup process, the Encryption Key Manager attempts a filecreate to the directory for a file with the name **.backup**. This is done to ensure that the server can write back to the configuration file when stopped.

Create a new filesystem, **hlq.ekmserv.etc** and mount at **/ekmetc**. Define a directory for each system to contain an Encryption Key Manager configuration file unique to that specific system. Ensure the RACF ID that the Encryption Key Manager server uses, EKMSERV, is the owner of the **/ekmetc** and subsequent directories and files.

The following are samples of configuration files for each keystore type, which may help you to get started. The examples show various keystore locations, directories and filenames, which you must customize for your system configuration. The examples for JCE4758RACFKS and JCERACFKS show setting up a system named JA0, which is a member of a sysplex that is setup to use Unix Systems Services Shared HFS.

Example Configuration File for JCEKS

This example configuration file for JCEKS is named **/u/ekmserv/KeyManagerConfig.properties.jceks**.

```
Admin.ssl.keystore.name = /u/ekmserv/EKMKeystore
Admin.ssl.keystore.password = password
Admin.ssl.keystore.type = jceks
Admin.ssl.truststore.name = /u/ekmserv/EKMKeystore
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /u/ekmserv/keymanager/keymanager/audit
Audit.handler.file.name = ekmaudit.log.jceks
Audit.handler.file.size = 10000
Audit.metadata.file.name = /u/ekmserv/metafile.xml
config.drivetable.file.url = FILE:///u/ekmserv/keymanager/drivetable
config.keystore.file = /u/ekmserv/EKMKeystore
config.keystore.password = password
config.keystore.provider = IBMJCE
config.keystore.type = jceks
drive.acceptUnknownDrives = true
drive.default.alias1 = MyEKMServerJCEKS
drive.default.alias2 = MyEKMServerJCEKS
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = /u/ekmserv/EKMKeystore
TransportListener.ssl.keystore.password = password
TransportListener.ssl.keystore.type = jceks
TransportListener.ssl.port = 5443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = /u/ekmserv/EKMKeystore
TransportListener.ssl.truststore.type = jceks
TransportListener.tcp.port = 3801
zOSCompatibility = true
```

Figure 14. Example of configuration file for jceks. Example of configuration file for jceks

If supporting LTO drives, add

```
symmetricKeySet = ibm01-1F4
```

This would represent 500 symmetric keys. Symmetric keys are only required when supporting LTO encryption drives.

Example Configuration File for JCE4758KS

This example configuration file for JCE4758KS is named **/u/ekmserv/ KeyManagerConfig.properties.jce4758ks**.

```
Admin.ssl.keystore.name = /u/ekmserv/EKMKeystore4758
Admin.ssl.keystore.password = password
Admin.ssl.keystore.type = jce4758ks
Admin.ssl.truststore.name = /u/ekmserv/EKMKeystore4758
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /u/ekmserv/keymanager/audit
Audit.handler.file.name = ekm.audit.log.jce4758ks
Audit.handler.file.size = 10000
Audit.metadata.file.name = /u/ekmserv/metafile.xml
config.drivetable.file.url = FILE:///u/ekmserv/keymanager/drivetable
config.keystore.file = /u/ekmserv/EKMKeystore4758
config.keystore.password = password
config.keystore.provider = IBMJCE4758
config.keystore.type = jce4758ks
drive.acceptUnknownDrives = true
drive.default.alias1 = MyEKMServerJCE4758KS
drive.default.alias2 = MyEKMServerJCE4758KS
fips = Off
requireHardwareProtectionForSymmetricKeys = true
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = /u/ekmserv/EKMKeystore4758
TransportListener.ssl.keystore.password = pasword
TransportListener.ssl.keystore.type = jce4758ks
TransportListener.ssl.port = 5443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = /u/ekmserv/EKMKeystore4758
TransportListener.ssl.truststore.type = jce4758ks
TransportListener.tcp.port = 3801
zOSCompatibility = true
```

Figure 15. Example of configuration file for JCE4758KS. Example of configuration file for JCE4758KS

If supporting LTO drives, add

```
symmetricKeySet = ibm01-1F4
```

This would represent 500 symmetric keys. Symmetric keys are only required when supporting LTO encryption drives.

Example Configuration File for JCE4758RACFKS

This example configuration file for JCE4758RACFKS is named **/ekmetc/JA0/KeyManagerConfig.properties.JCE4758RACFKS** .

```
Admin.ssl.keystore.name = safkeyring://EKMSERV/EKMRing
Admin.ssl.truststore.name = safkeyring://EKMSERV/EKMRing
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /ekmlogs/JA0/audit
Audit.handler.file.name = audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /u/ekmserv/metafile.xml
config.drivetable.file.url = FILE:/ekmetc/JA0/filedrive.table
config.keystore.file = safkeyring://EKMSERV/EKMRing
config.keystore.password = password
config.keystore.provider = IBMJCE4758
config.keystore.type = JCE4758RACFKS
drive.acceptUnknownDrives = true
drive.default.alias1 = EKMServer
drive.default.alias2 = EKMServer
fips = Off
requireHardwareProtectionForSymmetricKeys = true
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = safkeyring://EKMSERV/EKMRing
TransportListener.ssl.keystore.password = password
TransportListener.ssl.keystore.type = JCE4758RACFKS
TransportListener.ssl.port = 1443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = safkeyring://EKMSERV/EKMRing
TransportListener.ssl.truststore.type = JCE4758RACFKS
TransportListener.tcp.port = 3801
zOSCompatibility = true
```

Figure 16. Example of configuration file for JCE4758RACFKS. Example of configuration file for JCE4758RACFKS

Example Configuration File for JCERACFKS

This example configuration file for JCERACFKS is named **/ekmetc/JA0/ KeyManagerConfig.properties.JCERACFKS** .

```
Admin.ssl.keystore.name = safkeyring://EKMSERV/EKMRing
Admin.ssl.truststore.name = safkeyring://EKMSERV/EKMRing
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /ekmlogs/JA0/audit
Audit.handler.file.name = audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /u/ekmserv/metafile.xml
config.drivetable.file.url = FILE:/ekmetc/JA0/filedrive.table
config.keystore.file = safkeyring://EKMSERV/EKMRing
config.keystore.password = password
config.keystore.provider = IBMJCE
config.keystore.type = JCERACFKS
drive.acceptUnknownDrives = true
drive.default.alias1 = EKMServer
drive.default.alias2 = EKMServer
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = safkeyring://EKMSERV/EKMRing
TransportListener.ssl.keystore.password = password
TransportListener.ssl.keystore.type = JCERACFKS
TransportListener.ssl.port = 1443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = safkeyring://EKMSERV/EKMRing
TransportListener.ssl.truststore.type = JCERACFKS
TransportListener.tcp.port = 3801
zOSCompatibility = true
```

Figure 17. Example of configuration file for JCERACFKS. Example of configuration file for JCERACFKS

Quick Test Running Encryption Key Manager Under USS

For test purposes you may want to start the Encryption Key Manager from USS or OMVS to run in the foreground so that you can quickly attempt to write or read an encrypted tape. However, it is recommended for production purposes that you follow the steps in the following section, “Set Up and Run Encryption Key Manager in z/OS Production Mode” on page 72.

To run the Encryption Key Manager in the foreground, you need to have either a USS or OMVS session and verify that java in your path, see “Verify the Java Version” on page 46.

From your USS or OMVS session you can run one of the following commands based on the keystore type you are using. Be sure to replace the properties file with the location of your properties file:

JCEKS

```
/u/ekmserv/:>  java com.ibm.keymanager.EKMServer
/u/ekmserv/KeyManagerConfig.properties.jceks
```

JCE4758KS

```
/u/ekmserv/:>  java com.ibm.keymanager.EKMServer
/u/ekmserv/KeyManagerConfig.properties.jce4758ks
```

JCE4758RACFKS

```
u/ekmserv/:>  java -Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider
com.ibm.keymanager.EKMServer
/u/ekmserv/KeyManagerConfig.properties.jce4758racfks
```

JCERACFKS

```
u/ekmserv/:>  java -Djava.protocol.handler.pkgs=com.ibm.crypto.provider
com.ibm.keymanager.EKMServer
/u/ekmserv/KeyManagerConfig.properties.jceracfks
```

Notes:

For Encryption Key Manager versions later than 05032007, the Encryption Key Manager server is started with the EKMServer command instead of the KMSAdminCmd. In Encryption Key Manager versions with a build date of 05032007 and earlier, the EKMServer command is not recognized. Instead use the KMSAdminCmd, for example:`/u/ekmserv/:>  java com.ibm.keymanager.KMSAdminCmd`.

Unlike JCEKS and JCE4758KS, both JCE4758RACFKS and JCERACFKS require the **–Djava.protocol.handler.pkgs** parameters, which are defined differently for each as shown above.

Once you submit one of the above commands, a # prompt should appear as you see below.

```
/u/ekmserv:>java com.ibm.keymanager.EKMServer KeymanagerConfig.properties
#
Loaded drive key store successfully
Starting the Encryption Key Manager 2.1-20070719
Processing Arguments
Processing
Server is started
#
```

At this point you should be able to attempt to write and read encrypted tapes using a tape drive against this Encryption Key Manager Server. If you are setting up a z/OS tape drive to run in-band with your Encryption Key Manager, see "Note About z/OS Configuration Steps for z/OS In-band Encrypted Tape Drive" on page 147.

To stop the Encryption Key Manager server submit the 'quit" command. Shutting down the Encryption Key Manager server overwrites the KeyManagerConfig.properties.jceks with any changes that may have been made to it while the Encryption Key Manager Server is running. Please read Chapter 6, "Administering the Encryption Key Manager," on page 157 for more information on Encryption Key Manager commands.

```
# quit
Stopping the EKM admin service...
/u/ekmserv:>
```

Set Up and Run Encryption Key Manager in z/OS Production Mode

Define the Encryption Key Manager as z/OS Started Task

A procedure consists of a set of job control language statements that are frequently used together to achieve a certain result. Procedures usually reside in the system procedure library, SYS1.PROCLIB, which is a partitioned data set. A started procedure is usually started by an operator, but can be associated with a functional subsystem.

IBM recommends that the Encryption Key Manager run as a started procedure on z/OS using the JZOS batch launcher which is shipped as part of the z/OS Java product. In order to define the Encryption Key Manager as a started procedure update the started class table with the z/OS user ID of the Encryption Key Manager. Details of RACF's processing and the definition of started procedures is discussed in the *z/OS Security Server RACF Security Administrator's Guide*.

1. In this example, the user ID of the Encryption Key Manager instance on z/OS will be EKMSERV, and the group associated with that started procedure will be sys1. You can tailor these examples to suit your needs. To set up the STARTED class, enter these commands (for example):

   ```
   SETROPTS GENERIC(STARTED)
   RDEFINE STARTED EKM*.* STDATA(USER(EKMSERV) GROUP(STCGROUP) TRACE(YES))
   SETROPTS CLASSACT(STARTED) SETROPTS RACLIST(STARTED)
   SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
   ```
2. Create a home directory, which is the home directory specified on the RACF **adduser** command issued in "Set Up a User ID to Run the Encryption Key Manager" on page 48, such as shown in this example:

   ```
   Example:   /u/ekmserv
   ```
3. The **JZOSEKMFiles.pax.Z** download file consists of **jzosekm.jar**, sample JCL , the STDENV file for the sample JCL and a **EKMConsoleWrapper.README.txt** that explains where each file should be located and installation-specific tailoring that may be required. To extract the contents of the download file, issue:

   ```
   pax -rf JZOSEKMFiles.pax.Z
   ```

   If you are not using SDK 1.4.2 SR7, SDK 5.0 SR4 ,or a newer SR, follow instructions for using JZOSEKMFiles.pax.Z for installation of the **jzosekm.jar** into **$JAVA_HOME/lib/ext**. SDK 1.4.2 SR7 or 5.0 SR4 and later releases already include the **jzosekm.jar** in **$JAVA_HOME/lib/ext**.
4. Place the **jzosekm.jar** file into the lib/ext directory of the java installation, for example: /java/J1.4/lib/ext. Ensure the appropriate permission bits are set to allow execution.

   Note: For users of the z/OS Console Wrapper Function:
   - The materials in the pax file supersede any documentation found in other publications including other readme, sample JCL, or configuration files.
   - In future levels of the z/OS Java products, the console wrapper file will be included so that the download and copy will not be necessary. Until then, the console wrapper function contained in JZOSEKMFiles.pax.Z can be downloaded from the Encryption Key Manager webpage at http://www.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc=en_US&cs=utf-8&cc=us&lang=en.

Create a File System and Mount Point for Encryption Key Manager Logging of Debug and Audit Logs

The Encryption Key Manager can create audit records that wrap the log to three files. Once the last file is full, the first file is rewritten. Chapter 8, "Audit Records," on page 191 discusses the events that the Encryption Key Manager audits and the format of the audit records. On z/OS, you will need to allocate file system space for the Encryption Key Manager audit logs, and if requested by IBM Service, to enable the Encryption Key Manager debug log.

If the file system fills up and can no longer be extended, the Encryption Key Manager will continue to run without logging, however, a noticeable performance degradation may be encountered if the file system is a HFS. If using ZFS, there should be no change in performance.

1. You may choose to allocate a file system specifically for use by the Encryption Key Manager for audit and debug file storage. Assume 500 cylinders of space allocated to the Encryption Key Manager's audit and debug log file system until you have observed based on tape and Encryption Key Manager activity how quickly the audit logs wrap. The file system should not be shared by the Encryption Key Manager instances running in a sysplex environment, but they should be private to each Encryption Key Manager instance. This prevents any possible interleaving of audit or debug logs between Encryption Key Manager instances.
2. Mount the ekmlogs filesystem and create a directory for each system that the Encryption Key Manager will run on. For example the two file systems created may be ekmlogs with JA0 and JB0 being two system names of two images within a sysplex.

   ```
   /ekmlogs/JA0
   /ekmlogs/JB0
   ```

Create a New PDS to Contain the Shell Script for the JZOS Launcher

1. Create a new PDS to contain the STDENV environmental variables. In this example, a portioned dataset was allocated whose name is EKMSERV.ENCRYPT.CONFIG

   ```
   EKMSERV.ENCRYPT.CONFIG
    Organization  . . . : PO
    Record format . . . : FB
    Record length . . . : 80
    Block size  . . . . : 6160
    1st extent cylinders: 3
        Secondary cylinders : 1
   ```

Create a Member in EKMSERV.ENCRYPT.CONFIG

Create/edit the shell script contents as shown in Figure 18 on page 75. The text below that is preceded by # designates a comment. The example shown below is setup for the system JA0 and is a member of a PDS "EKMSERV.ENCRYPT.CONFIG" that is pointed to by the Encryption Key Manager start procedure shown in Figure 19 on page 76 that follows. In SDK 1.4.2 SR7 and later, this example may be found in **J1.4/mvstools/samples/jcl/ PROCLIB.EKM2ENV**.

Shell script example to create member in EKMSERV.ENCRYPT.CONFIG

```
# This is a shell script which configures
# any environment variables for the Java JVM.
# Variables must be exported to be seen by the launcher.

. /etc/profile

export JAVA_HOME="/java/ekm/J1.4"
export PATH=/bin:"${JAVA_HOME}"/bin:

LIBPATH=/lib:/usr/lib:"${JAVA_HOME}"/bin
LIBPATH="$LIBPATH":"${JAVA_HOME}"/bin/classic

export LIBPATH="$LIBPATH":

# Customize your CLASSPATH here
CLASSPATH=${JAVA_HOME}/lib
CLASSPATH=/u/ekmserv:$CLASSPATH

export CLASSPATH="$CLASSPATH":

# Set JZOS specific options
export EKMCLASS="com.ibm.keymanager.EKMServer"
export EKMARGS="/ekmetc/JA0/KeyManagerConfig.properties.jce4758racfks"
export JZOS_MAIN_ARGS="$EKMCLASS $EKMARGS"

# Configure JVM options (if any)
# for JCECCARACFKS/JCE4758RACFKS, following IJO definition is required
IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider"
# for JCERACFKS, following IJO definition is required
# IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.provider"
# for JCEKS and JCE4758KS/JCECCAKS, no IJO definition is required
export IBM_JAVA_OPTIONS="$IJO"

#export JAVA_DUMP_HEAP=false
#export JAVA_PROPAGATE=NO
#export IBM_JAVA_ZOS_TDUMP=NO
```

Figure 18. Shell script example to create member in EKMSERV.ENCRYPT.CONFIG

Note: For Encryption Key Manager versions later than 05032007, the Encryption Key Manager server is started with the EKMServer command instead of the KMSAdminCmd. The contents of the EKMCLASS and EKMARGS variables are changed to show the new EKMServer command. In Encryption Key Manager versions with a build date of 05032007 and earlier, the EKMServer command is not recognized. Instead use the KMSAdminCmd, for example:

```
| Set JZOS specific options
| export EKMCLASS="com.ibm.keymanager.KMSAdminCmd"
| export EKMARGS="/ekmetc/JA0/KeyManagerConfig.properties.jce4758racfks"
| ...
```

Customize and Install Encryption Key Manager Start Procedure

Create/edit contents as shown in Figure 19 on page 76. The text in italics for comment purposes only and need not be entered. Be sure to review the installation documentation for the z/OS Java product which contains additional guidelines, annotated samples, and step by step installation instructions for the JZOS Batch Launcher function.

The JZOS Batch Launcher is supported by its own set of environment variables. For further information about the JZOS Batch Launcher, refer to *JZOS Batch Launcher and Toolkit Installation and User's Guide* located at http://www.ibm.com/servers/eserver/zseries/software/java/jzos/overview.html.

JCL example to create member in EKMSERV.ENCRYPT.CONFIG. For SDK 1.4.2 SR7 and later, this sample may be in **J1.4/mvstools/samples/jcl/PROCLIB.EKM2**.

```
//EKM PROC JAVACLS='com.ibm.jzosekm.EKMConsoleWrapper',
//   ARGS=,                              < Args to Java class
//   LIBRARY='JZOS.LOADLIB',             < STEPLIB FOR JVMLDM module
//   VERSION='14',                       < JVMLDM version: 14, 50, 56
//    LOGLVL='+T',                       < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                       < EXECUTION REGION SIZE
//    LEPARM=''
//********************************************************
//*EKM PROC JAVACLS='EKMConsoleWrapper', < fully qualified Java class
//*
//*Stored procedure for executing the JZOS Batch Launcher
//*Specifically, to execute the Enterprise Key Manager (EKM) under JZOS
//*
//********************************************************
//EKM  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
//*
//* &SYSNAME symbolic for the system unique KeyManagerConfigFile
//*
 //*********************************************************************
//STDENV DD DSN=EKMSERV.ENCRYPT.CONFIG(&SYSNAME.),DISP=SHR
//
```

Figure 19. Example of Encryption Key Manager start procedure for z/OS. Example of Encryption Key Manager start procedure for z/OS

Create Encryption Key Manager Configuration File

If you have not already done so, you need to create an Encryption Key Manager configuration file for each system on which you plan to run an Encryption Key Manager. In this example, the configuration file is called **/ekmetc/JA0/KeyManagerConfig.properties.jce4758racfks** where JAO is the system name. Please read all of Chapter 5, "Configuring the Encryption Key Manager," on page 135 for information on how to set this file up for z/OS production mode.

Starting and Stopping Encryption Key Manager on z/OS

The Encryption Key Manager process may now be started with the operator **start** command, as a started task. The operator's console may be used to issue commands through operator modify commands. The sample excerpt from the z/OS operators console, shows the start of the Encryption Key Manager, an operator issued modify command to list the drives known to the Encryption Key Manager, and its termination.

```
S EKM
$HASP100 EKM ON STCINRDR
IEF695I START EKM WITH JOBNAME EKM IS ASSIGNED TO USER EKMSERV, GROUP SYS1
$HASP373 EKM      STARTED
EKM console interaction is now available. 546
To submit commands to the EKM from the console:
    F EKM,APPL='EKM command'
To stop the EKM properly:
    P EKM
Loaded drive key store successfully
Starting the Encryption Key Manager 2.0-20070419
Processing Arguments
Processing
Server is started
Server is running. TCP port: 3801, SSL port: 443

F EKM,APPL='LISTDRIVES'
Drive entries: 14
SerialNumber = 00000AZ00011
SerialNumber = 000001365050
SerialNumber = 000001365043
SerialNumber = 000001365042
SerialNumber = 000001365041
SerialNumber = 000001365012
SerialNumber = 000001365067
SerialNumber = 000001365037
SerialNumber = 00000AZ00127
SerialNumber = 000001350808
SerialNumber = 00000AZ00125
SerialNumber = 000001365089
SerialNumber = 000001365088
SerialNumber = 000001365031

 P EKM
 Stopping the EKM admin service...
 Server is not started
 EKM stop command received
 IEF170I 1 EKM      - ================================================
 IEF170I 1 EKM      -                                     REGION
 IEF170I 1 EKM      - STEPNAME PROCSTEP PGMNAME     CC     USED      CP
 IEF170I 1 EKM      -          EKM      JVMLDM14    00     100K   00:00
 IEF170I 1 EKM      - ================================================
 IEF170I 1 EKM      - NAME-                    TOTALS: CPU TIME=  00:0
 IEF170I 1 EKM      - ================================================
 $HASP395 EKM      ENDED
```

Figure 20. Example contents of z/OS operator console. Example contents of z/OS operator console

Please note that to properly end the Encryption Key Manager server, issue the MVS STOP Command,

```
STOP EKM
```

from the operator console. If the Encryption Key Manager is canceled (**CANCEL** command) instead of being stopped, then critical data may be lost.

In general, commands that may be issued from the shell environment to manage the Encryption Key Manager can be issued from the operator's console. These commands are specified via the modify command (F), with the syntax used for Encryption Key Manager as documented in this publication enclosed in single quotes. For example, to enter Encryption Key Manager commands from the operator console to the started task when EKM is the name of the started task, use the following syntax. `f ekm,appl='ekm_command'`. If the ekm_command is not bounded by single quotes, the command will be rejected by EKMConsoleWrapper with an error message to the console. Invalid ekm_commands bounded by single quotes will be rejected by the Encryption Key Manager server with an error message to the console. Consult "CLI Commands" on page 161 for the valid Encryption Key Manager command syntax.

Installing on Unix-based and Windows Operating Systems

The following topics show examples of how to install the Encryption Key Manager on various operating systems.

- "Install the IBM Software Developer Kit on AIX"
- "Install the IBM Software Developer Kit on Sun Solaris" on page 79
- "Install the IBM Software Developer Kit on HP-UX" on page 80
- "Install the IBM Software Developer Kit Manually on Linux (Intel-based)" on page 81
- "Install the IBM Software Developer Kit on Linux (p-Series)" on page 82
- "Install the IBM Software Developer Kit on Linux (z-Series)" on page 83
- "Installing the Encryption Key Manager and the IBM Software Developer Kit on Windows" on page 84
- "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87

Note: The following procedures illustrate one way of installing and configuring the SDK and then installing the Encryption Key Manager for use in a tape drive encryption environment. Other methods and procedures can be used, depending on the operating system. These procedures are intended only as examples.

Install the IBM Software Developer Kit on AIX

1. From http://www.ibm.com/developerworks/java/jdk/aix/service.html, download the correct IBM runtime environment for Java based on your operating system:
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5 (64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java tar file in a working directory on your host:
   ```
   ost28> pwd
   /tape/Encryption/java/5.0.0
   ost28> ls
   j532redist.tar
   ```
3. Software such as Java can reside in the **/usr** dir on AIX. You can place your extracted files there in a directory name of your choice. Remember this location for updating the **.profile** later..
   ```
   Create a directory in /usr called java5.
   Extract the tar files to the new directory /usr/java5/.
   ost28> cd /usr/java5
   ost28> tar -xvf /tape/Encryption/java/5.0.0/j532redist.tar
   ost28> ls
   license sdk
   ```
4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files required by theEncryption Key Manager in order to serve AES keys.
5. Edit the **/home/root/.profile** file with the JAVA_HOME and the bin dir for the Java you installed.

a. Add these lines:

```
JAVA_HOME=/usr/java5/sdk/jre/
PATH=$JAVA_HOME/bin:$PATH
```

6. Logout and log back into your host for the **.profile** entries to take affect or issue these commands:

```
ost28> export JAVA_HOME=/usr/java5/sdk/jre/
ost28> export PATH=$JAVA_HOME/bin:$PATH
```

7. After you log back in, issue the **java -version** command. You should see these results:

```
ost28> java -version
java version "1.5.0" Java(TM) 2 Runtime Environment,
Standard Edition (build pap32dev-20070426 (SR5)) IBM J9 VM (build 2.3,
J2RE 1.5.0 IBM J9 2.3 AIX ppc-32 j9vmap3223-20070426 (JIT enabled)
J9VM - 20070420_12448_bHdSMr
JIT - 20070419_1806_r8
GC - 200704_19)
JCL - 20070425
ost28> which java
/usr/java5/sdk/jre/bin/java
```

Install the IBM Software Developer Kit on Sun Solaris

1. For Sun Solaris, the IBM Java Runtime Environment is installed from CD. See "Encryption Key Manager (Running on HP, Sun, or Windows)" on page 25:
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5 (64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java tar file in a working directory on your host:

```
# pwd
/tape/Encryption/java/5.0/
# ls
sol64devhybrid-20070511a-jre.tar
```

3. Software such as Java can reside in the **/usr** dir on Solaris. You can place your extracted files there in a directory name of your choice. Remember this location for updating the **.profile** later.

```
Create a directory in /usr called java5.
Extract the tar files to the new directory /usr/java5/.
# tar -xvf sol64devhybrid-20070511a-jre.tar
# ls
CHANGES                          ibmorbguide.htm
COPYRIGHT                        javaws
Notices.htm                      lib
README                           plugin
README_FIRST                     securityguide.ref.htm
Welcome.html                     sol64devhybrid-20070511a-jre.tar
bin

# mkdir /usr/java5
# mv sol64devhybrid-20070511a-jre.tar ../
# cp -rf * /usr/java5
# ls
/usr/java5/
CHANGES                          ibmorbguide.htm
COPYRIGHT                        javaws
Notices.htm                      lib
README                           plugin
README_FIRST                     securityguide.ref.htm
Welcome.html                     bin
license sdk
```

4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your *java_home***/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files required by the Encryption Key Manager in order to serve AES keys.
5. Edit the **/.profile** file with the JAVA_HOME, CLASSPATH, and the bin dir for the Java you installed.
   a. Add these two lines:
   ```
   JAVA_HOME=/usr/java5/
    CLASSPATH=/usr/java5/lib
   ```
   b. Append this to the PATH variable line:
   ```
   :$JAVA_HOME:/usr/java5/bin
   ```
6. Logout and log back into your host for the **.profile** entries to take effect or issue the export command line commands:
   ```
   # export JAVA_HOME=/usr/java5/
   # export CLASSPATH=/usr/java5/lib
   # export PATH=/usr/java5/bin/:$PATH
   ```
7. After you log back in, issue the **java -version** command. You should see these results:
   ```
   java version "1.5.0_06"
   Java(TM) 2 Runtime Environment, Standard Edition (IBM build 1.5.0_06-erdist-20070426)
   Java HotSpot(TM) Server VM (build 1.5.0_06-erdist-20070426, mixed mode)
   IBM Java ORB build orb50-20070504 (SR5)
   XML build XSLT4J Java 2.7.4
   XML build IBM JAXP 1.3.5
   XML build XML4J 4.4.5
   # which java
    /usr/java5/bin/java
   ```

Install the IBM Software Developer Kit on HP-UX

1. For HP-UX, Sun Solaris, and Windows, the IBM Java Runtime Environment is installed from CD. See “Encryption Key Manager (Running on HP, Sun, or Windows)” on page 25.
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5 (64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java tar file in a working directory:
   ```
   root@megas)/tape/Encryption/java/5.0# pwd
   /tape/Encryption/java/5.0
   (root@megas)/tape/Encryption/java/5.0# ls
   hpux64devhybrid-20070511a-jre.tar
   ```
3. Software such as Java can reside in the **/usr** dir on HP-UX. You can place your extracted files there in a directory name of your choice. Remember this location for updating the **.profile** later..
   ```
   Create a directory in /usr called java5.
   Extract the tar files to the new directory /usr/java5/.
   (root@megas)/tape/Encryption/java/5.0# tar -xvf hpux64devhybrid-20070511a-jre.tar
   (root@megas)/tape/Encryption/java/5.0# ls
   .systemPrefs/                      ibmorbguide.htm
   COPYRIGHT*                         java.os11.release.notes.html*
   LICENSE*                           javaws/
   README_FIRST                       lib/
   THIRDPARTYLICENSEREADME.txt*       man/
   bin/                               plugin/
   ```

```
hpia64devhybrid-20070511a-jre.tar   securityguide.ref.htm

(root@megas)/tape/Encryption/java/5.0# mkdir /usr/java5
(root@megas)/tape/Encryption/java/5.0# mv hpia64devhybrid-20070511a-jre.tar ../
(root@megas)/tape/Encryption/java/5.0# cp -rf * /usr/java5
(root@megas)/tape/Encryption/java/5.0# ls /usr/java5/
.systemPrefs/                       ibmorbguide.htm
COPYRIGHT*                          java.os11.release.notes.html*
LICENSE*                            javaws/
README_FIRST                        lib/
THIRDPARTYLICENSEREADME.txt*        man/
bin/                                plugin/
securityguide.ref.htm
```

4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files required by the Encryption Key Manager in order to serve AES keys.
5. Edit the **/home/root/.profile** file with the JAVA_HOME, CLASSPATH, and the bin dir for the Java you installed.
   a. Add these two lines:
   ```
   JAVA_HOME=/usr/java5/
    CLASSPATH=/usr/java5/lib
   ```
   b. Append this to the PATH variable line:
   ```
   :$JAVA_HOME:/usr/java5/bin
   ```
6. Logout and log back into your host for the **.profile** entries to take affect or issue these commands:
   ```
   # export JAVA_HOME=/usr/java5/
   # export CLASSPATH=/usr/java5/lib
   # export PATH=/usr/java5/bin/:$PATH
   ```
7. After you log back in, issue the **java -version** command. You should see these results:
   ```
   (root@megas)/home/root# java -version
   java version "1.5.0.05-_04_may_2007"
   Java(TM) 2 Runtime Environment, Standard Edition (IBM build 1.5.0.05-_04_may_2007-16_39 20070504)
   Java HotSpot(TM) 64-Bit Server VM (build 1.5.0.05 jinteg:02.14.06-00:56 IA64W, mixed mode)
   IBM Java ORB build orb50-20070504 (SR5)
   XML build XSLT4J Java 2.7.4
   XML build IBM JAXP 1.3.5
   XML build XML4J 4.4.5
   (root@megas)/home/root# which java
   /usr/java5/bin/java
   ```

Install the IBM Software Developer Kit Manually on Linux (Intel-based)

1. From http://www.ibm.com/developerworks/java/jdk/linux/download.html, download the correct IBM runtime environment for Java based on your operating system (Java 5 is recommended but Java 1.4.2 can be used if it is already installed or is required in your environment):
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5(64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java linux rpm file in a working directory:

```
mordor:~ #/tape/Encryption/java/1.4.2# pwd
/tape/Encryption/java/1.4.2
mordor:~ #/tape/Encryption/java/1.4.2# ls
ibm-java2-i386-jre-5.0-2.0.i386.rpm
```

3. Install the rpm package:

   ```
   mordor:~ #rpm -ivh ibm-java2-i386-jre-5.0-2.0.i386.rpm
   ```

   This will place the files in the **/opt/ibm/java2-i386-50/** dir

   ```
   mordor:~ #/opt/ibm/java2-i386-50/jre # ls
   .systemPrefs  bin  javaws  lib
   ```

4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files required by the Encryption Key Manager in order to serve AES keys.
5. Edit (or create if necessary) the file **/etc/profile.local** with the JAVA_HOME, CLASSPATH, and the bin dir for the Java you installed. Add these three lines:

   ```
   JAVA_HOME=/opt/ibm/java2-i386-50/jre
   CLASSPATH=/opt/ibm/java2-i386-50/jre/lib
   PATH=$JAVA_HOME:opt/ibm/java2-i386-50/jre/bin/:$PATH
   ```

6. Logout and log back into your host for the **/etc/profile.local** entries to take effect or issue the export command line commands:

   ```
   mordor:~ # export JAVA_HOME=/opt/ibm/java2-i386-50/jre
   mordor:~ # export CLASSPATH=/opt/ibm/java2-i386-50/jre/lib
   mordor:~ # export PATH=/opt/ibm/java2-i386-50/jre/bin/:$PATH
   ```

7. After you log back in, issue the **java -version** command. You should see these results:

   ```
   mordor:~ # java -version
   java version "1.5.0"
   Java(TM) 2 Runtime Environment, Standard Edition (build pmz31dev-20070426 (SR5))
   IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux x86-32 j9vmxi3223-20070504 (JIT enabled)
   J9VM - 20070420_12448_bHdSMr
   JIT  - 20070419_1806_r8
   GC   - 200704_19)
   JCL  - 20070425

   mordor:~ # which java
   /opt/ibm/java2-i386-50/jre/bin/java
   ```

Install the IBM Software Developer Kit on Linux (p-Series)

1. From http://www.ibm.com/developerworks/java/jdk/linux/download.html, download the correct IBM runtime environment for Java based on your operating system:
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5 (64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java linux rpm file in a working directory:

   ```
   tapeiot2:/tape/Encryption/java/5.0# pwd
   /tape/Encryption/java/5.0
   tapeiot2:/tape/Encryption/java/5.0# ls
   ibm-java2-ppc64-jre-5.0-2.0.ppc64.rpm
   ```

3. Install the rpm package:

   ```
   tapeiot2:rpm -ivh ibm-java2-ppc64-jre-5.0-2.0.ppc64.rpm
   ```

This will place the files in the **/opt/ibm/java2-ppc64-50/jre** dir.

```
tapeiot2:ls
.systemPrefs  bin  javaws  lib
```

4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk. These install the unrestricted policy files required by the Encryption Key Manager in order to serve AES keys.
5. Edit the file **/etc/profile.local** with the JAVA_HOME, CLASSPATH, and the bin dir for the Java you installed. Add these three lines:

```
JAVA_HOME=/opt/ibm/java2-ppc64-50/jre
CLASSPATH=/opt/ibm/java2-ppc64-50/jre/lib
PATH=$JAVA_HOME:/opt/ibm/java2-ppc64-50/jre/bin/:$PATH
```

6. Logout and log back into your host for the **/etc/profile.local** entries to take effect or issue the export command line commands:

```
tapeiot2: export JAVA_HOME=/opt/ibm/java2-i386-50/jre
tapeiot2: export CLASSPATH=/opt/ibm/java2-i386-50/jre/lib
tapeiot2: export PATH=/opt/ibm/java2-i386-50/jre/bin/:$PATH
```

7. After you log back in, issue the **java -version** command. You should see these results:

```
tapeiot2: java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pxp64dev-20070511 (SR5))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux ppc64-64 j9vmxp6423-20070504 (JIT enabled)
J9VM - 20070420_12448_bHdSMr
JIT  - 20070419_1806_r8
GC   - 200704_19)
JCL  - 20070425
tapeiot2: which java
/opt/ibm/java2-ppc64-50/jre/bin/java
```

Install the IBM Software Developer Kit on Linux (z-Series)

1. From http://www.ibm.com/servers/eserver/zseries/software/java/, download the correct IBM runtime environment for Java based on your operating system:
   - Java 5 SR5 (32-bit) or later
   - Java 5 SR5 (64-bit) or later
   - Java 1.4.2 SR8 (32-bit) or later
   - Java 1.4.2 SR8 (64-bit) or later
2. Place the IBM Java linux rpm file in a working directory:

```
[root@zlinux9 bin]#/tape/Encryption/java/5.0# pwd
/tape/Encryption/java/5.0
[root@zlinux9 bin]#/tape/Encryption/java/5.0# ls
ibm-java2-s390x-jre-5.0-2.0.s390x.rpm
```

3. Install the rpm package:

```
[root@zlinux9 bin]#rpm -ivh ibm-java2-s390x-jre-5.0-2.0.s390x.rpm
```

This will place the files in the **/opt/ibm/java2-s390-50/jre** dir

```
[root@zlinux9 bin]#ls
bin  lib
```

4. Regardless which version IBM SDK you use, you must replace the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory with new files you can download from https://www14.software.ibm.com/webapp/iwm/web/

preLogin.do?source=jcesdk. These install the unrestricted policy files required by the Encryption Key Manager in order to serve AES keys.

5. Edit the file **/etc/profile.local** with the JAVA_HOME, CLASSPATH, and the bin dir for the Java you installed. Add these three lines:

```
JAVA_HOME=/opt/ibm/java2-s390-50/jre
CLASSPATH=/opt/ibm/java2-s390-50/jre/lib
PATH=$JAVA_HOME:/opt/ibm/java2-s390-50/jre/bin/:$PATH
```

6. Logout and log back into your host for the **/etc/profile.local** entries to take effect or issue the export command line commands:

```
[root@zlinux9 bin]# export JAVA_HOME=/opt/ibm/java2-s390-50/jre
[root@zlinux9 bin]# export CLASSPATH=/opt/ibm/java2-s390-50/jre/lib
[root@zlinux9 bin]# export PATH=/opt/ibm/java2-s390-50/jre/bin/:$PATH
```

7. After you log back in, issue the **java -version** command. You should see these results:

```
[root@zlinux9 bin]# java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pxz31dev-20070511 (SR5))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Linux s390-31 j9vmxz3123-20070505 (JIT enabled)
J9VM - 20070420_12448_bHdSMr
JIT  - 20070419_1806_r8
GC   - 200704_19)
JCL  - 20070425

[root@zlinux9 bin]# which java
/opt/ibm/java2-s390-50/jre/bin/java
```

Installing the Encryption Key Manager and the IBM Software Developer Kit on Windows

Note: Regardless which version SDK you use, you must ensure that the **US_export_policy.jar** and **local_policy.jar** files in your in your ***java_home*/usr/java5/jre/lib/security** directory are the correct version. Please refer to the installation instructions included in your authorization code kit for guidance.

1. Insert the CD from the IBM TotalStorage Productivity Center - Basic Edition (TPC-BE) - LPP 5608-B01 and select your IBM Java Runtime Environment. Refer to "How to Obtain IBM Encryption Key Manager Java Code for Windows, HP-UX, and Sun Solaris Operating Systems" on page 27 for more information.

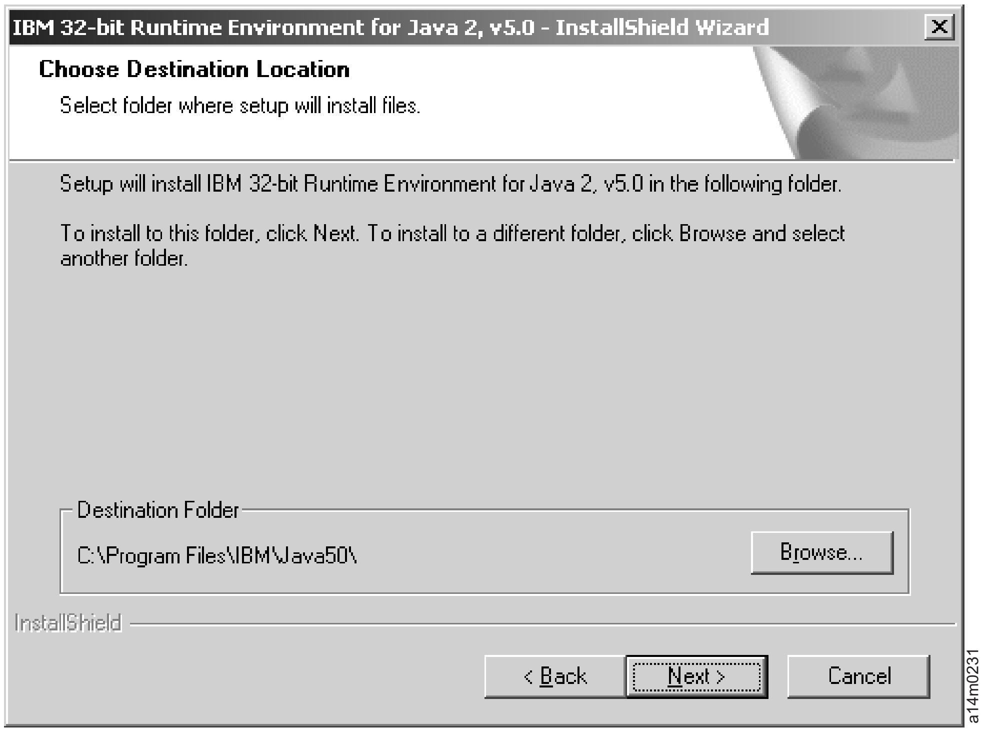
   Note: IBM TotalStorage Productivity Center - Limited Edition (TPC-LE) - LPP 5608-VC6 is withdrawn and replaced by TPC/BE, a chargeable product.

   Java 5.0 Service Release 5 (Windows/AMD64/EM64T)

   Java 5.0 Service Release 5 (Windows/IA32)

   Java 1.4.2 Service Release 8 (Windows/IA64)

2. When the InstallShield Wizard opens, click **Next**.
3. Read the License Agreement and click **Yes**.
4. When the Choose Destination Location window opens (Figure 21 on page 85), choose a folder and make note of it. You will need this Java path to launch the Encryption Key Manager.



IBM 32-bit Runtime Environment for Java 2, v5.0 - InstallShield Wizard

Choose Destination Location

Select folder where setup will install files.

Setup will install IBM 32-bit Runtime Environment for Java 2, v5.0 in the following folder.

To install to this folder, click Next. To install to a different folder, click Browse and select another folder.

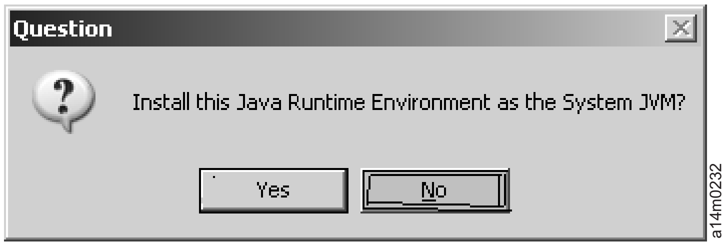
Destination Folder

C:\Program Files\IBM\Java50\

Figure 21. Choose Destination Location window

Click **Next**.

5. A window opens asking if you want this Java Runtime Environment as the default system JVM (Figure 22).



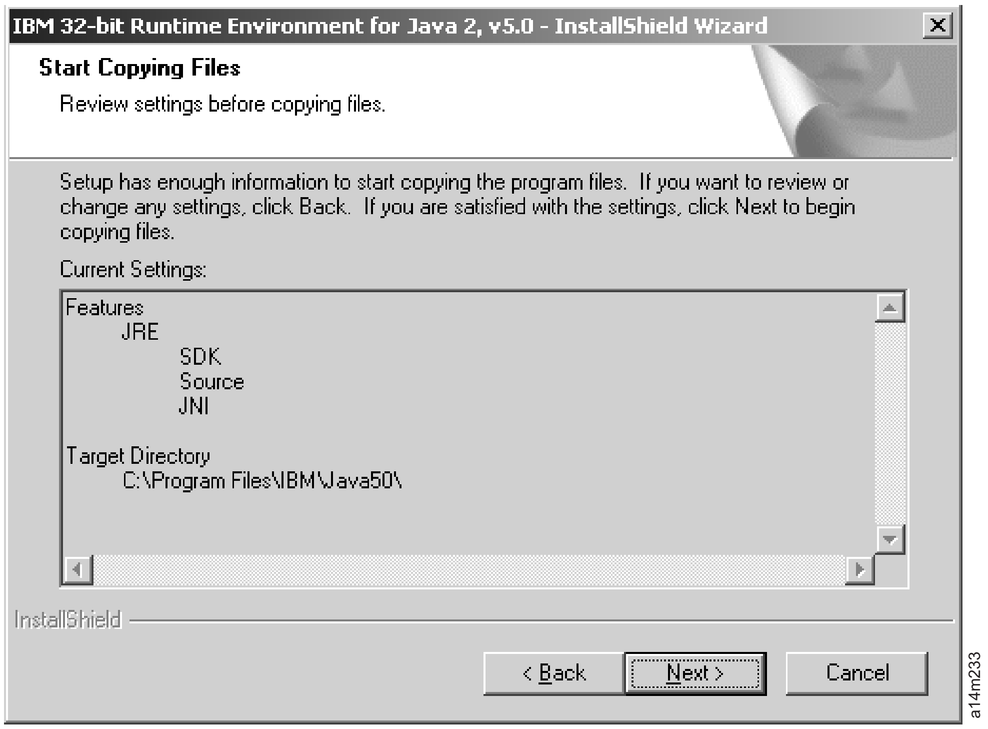
Question

Install this Java Runtime Environment as the System JVM?

Figure 22. Set this version of JVM to default

Click **No**.

6. The Start Copying Files window opens (Figure 23 on page 86). Make sure you have taken note of the target directory.


IBM 32-bit Runtime Environment for Java 2, v5.0 - InstallShield Wizard

Start Copying Files

Review settings before copying files.

Setup has enough information to start copying the program files. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Next to begin copying files.

Current Settings:

```
Features
        JRE
                SDK
                Source
                JNI

Target Directory
        C:\Program Files\IBM\Java50\
```

Figure 23. Start Copying Files window

Click **Next**.

7. The status window indicates installation progress.
8. The Browser Registration window opens. Choose a browser to use with Encryption Key Manager. Click **Next**.
9. When the InstallShield Wizard Complete window opens, click **Finish**.

   After installation, you can open a command prompt to query the Java version installed:

```
C:\WinEKM>C:\"Program Files"\IBM\Java50\jre\bin\java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pwi32dev-20070511 (SR5))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Windows Server 2003 x86-32 j9vmwi3223-20070504
 (JIT enabled)
J9VM - 20070420_12448_bHdSMr
JIT  - 20070419_1806_r8
GC   - 200704_19)
JCL  - 20070425
```

10. Update the PATH variable (required for Encryption Key Manager 2.1 but optional for build date of 05032007 and earlier).

    If you will be invoking the java SDK from a command window, you might wish to set the PATH variable if you want to be able to run the Java 2 SDK executables (java.exe) from any directory without having to enter the full path of the command. If you don’t set the PATH variable, you must specify the full path to the executable every time you run it, such as:

```
C:> \Program Files\IBM\Java50\jre\bin\java ....
```

    To set the PATH permanently (required for Encryption Key Manager 2.1), add the full path of the java bin directory to the PATH variable. Typically this full path looks similar to

```
C:\Program Files\IBM\Java50\jre\bin
```

    To set the PATH permanently in Microsoft Windows 2000 and 2003:

Select **Start > Settings > Control Panel** and double-click **System**. Select the Advanced tab and then Environment Variables. Look for ″Path″ in the User Variables and System Variables. The PATH can be a series of directories separated by semi-colons (;). Microsoft Windows looks for programs in the PATH directories in order, from left to right. You should only have one bin directory for a Java SDK in the path at a time (those following the first are ignored), so if one is already present, you can update it. If you're not sure where to add the path, add it to the beginning of the ″Path″ in the User Variables. A typical value for PATH is shown above. Capitalization does not matter.

Click **Set**, **OK** or **Apply**.

The new path takes effect in each new Command Prompt window you open after setting the PATH variable.

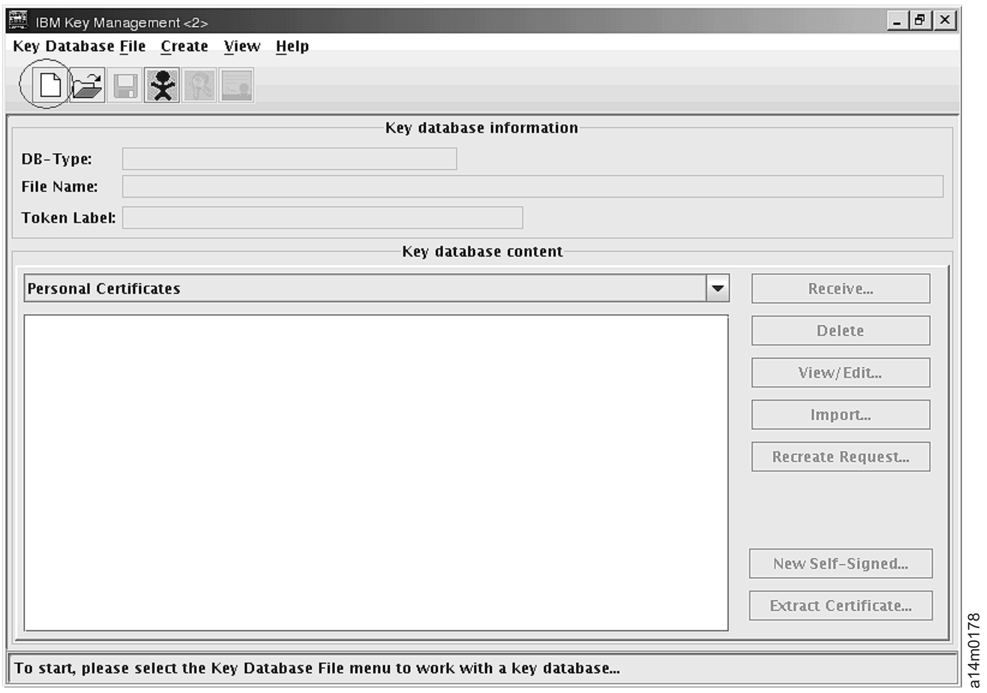
Create a Keystore and Certificates (Unix-based and Windows Operating Systems)

Before configuring and launching the Encryption Key Manager, you must create at least one new keystore and at least one self-signed certificate. In the following example we created one new keystore containing two certificates.

1. Issue the ikeyman command:

   ```
   ost28> java com.ibm.gsk.ikeyman.Ikeyman&
   ```

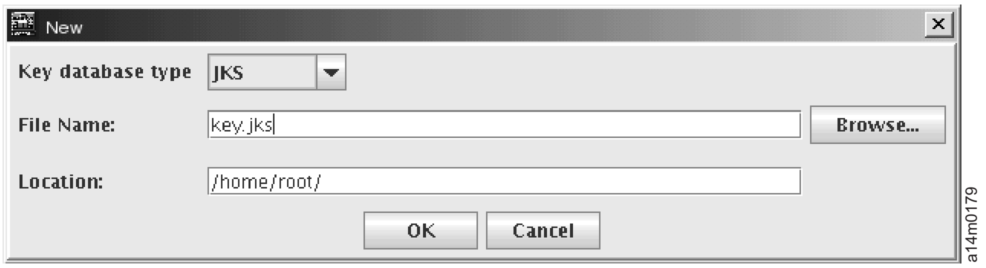
2. The "IBM Key Management" window opens (Figure 24. Click the new file (circled) icon below the menu bar or click **Key Database** > **File** > **New**.



To start, please select the Key Database File menu to work with a key database...

Figure 24. ikeyman Key Management window with new file icon circled

3. The "New Key Database" window opens (Figure 25).



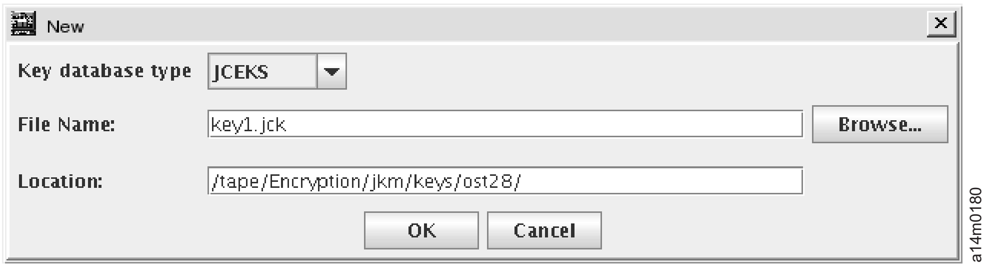
Key database type: JKS

File Name: key.jks

Location: /home/root/

Figure 25. New Key Database window

4. Use the **Key database type** pulldown to click **JCEKS** and enter the keystore filename and location (Figure 26). You will also be required to specify this filename and location in the Encryption Key Manager configuration file.



Key database type: JCEKS

File Name: key1.jck

Location: /tape/Encryption/jkm/keys/ost28/

Figure 26. Changed Values in New Key Database window

Select **OK**.

5. The "Password" window opens (Figure 27 on page 89). Specify a password. You will specify this same password in the Encryption Key Manager configuration file.

| **Note:** The keystore password must not be greater than 127 characters in
| length.

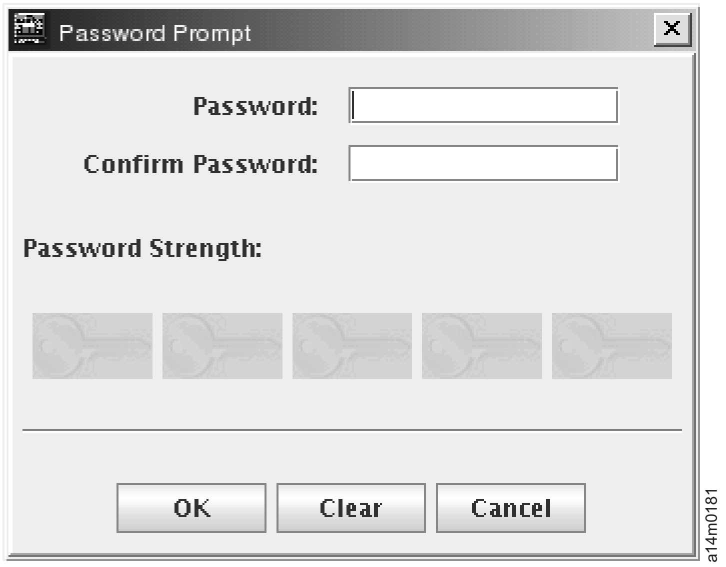
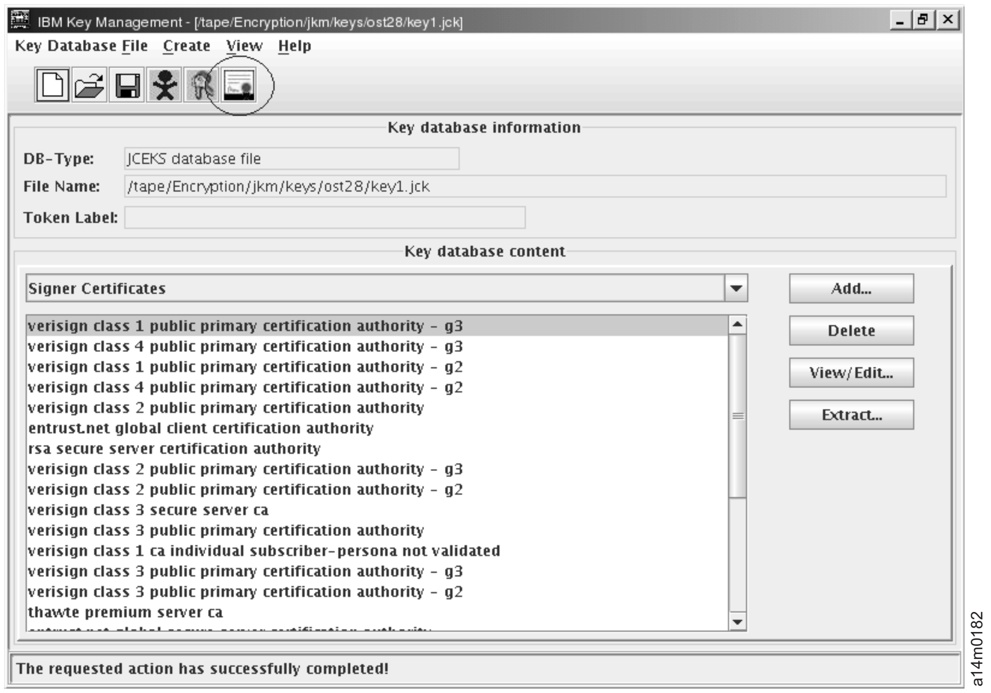


Figure 27. Keystore Password window

Click **OK**.

6. In the “IBM Key Management” window (Figure 28), click the certificate (circled) icon or click **Create** > **Create new self signed certificate**.



DB-Type: JCEKS database file
File Name: /tape/Encryption/jkm/keys/ost28/key1.jck
Token Label:

Signer Certificates

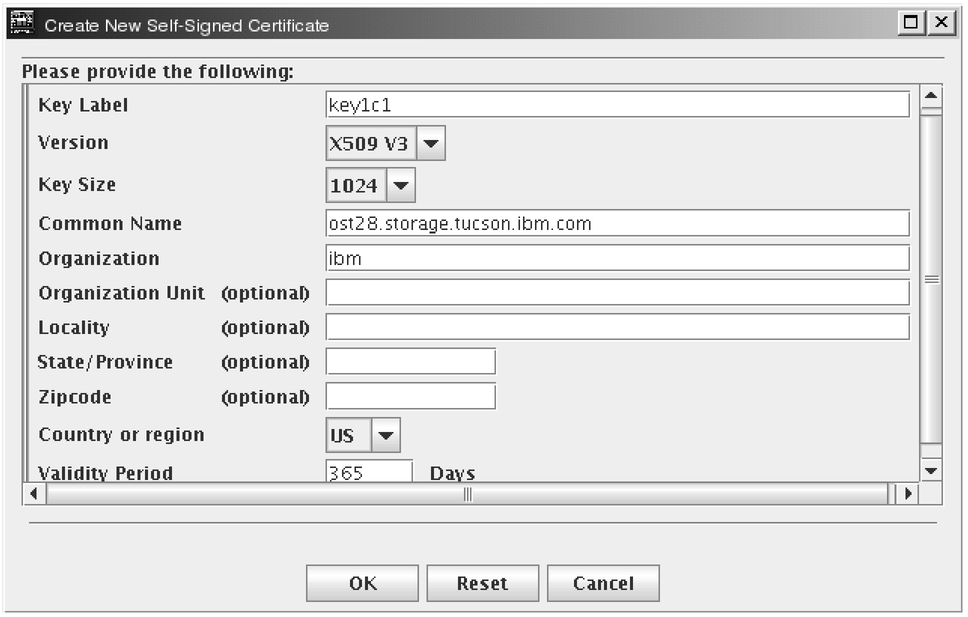
verisign class 1 public primary certification authority - g3
verisign class 4 public primary certification authority - g3
verisign class 1 public primary certification authority - g2
verisign class 4 public primary certification authority - g2
verisign class 2 public primary certification authority
entrust.net global client certification authority
rsa secure server certification authority
verisign class 2 public primary certification authority - g3
verisign class 2 public primary certification authority - g2
verisign class 3 secure server ca
verisign class 3 public primary certification authority
verisign class 1 ca individual subscriber-persona not validated
verisign class 3 public primary certification authority - g3
verisign class 3 public primary certification authority - g2
thawte premium server ca

The requested action has successfully completed!

Figure 28. ikeyman Key Management window with certificate icon circled

7. Specify the following values in the "Create new self signed certificate" window (Figure 29)
   - Key label can be any value but must not contain any blanks. The key label values correspond to the values you will specify for *alias1* and *alias2* when editing the Encryption Key Manager configuration file.
   - Click **X509 V3** in the "version" pulldown menu.
   - Click **2048** in the "key size" menu pulldown.
   - The "Common name" field defaults to the name of the host that launched the ikeyman application. Any name can be specified.
   - An organization name must be specified to create a certificate.

   The remaining fields are optional or default values.



Create New Self-Signed Certificate

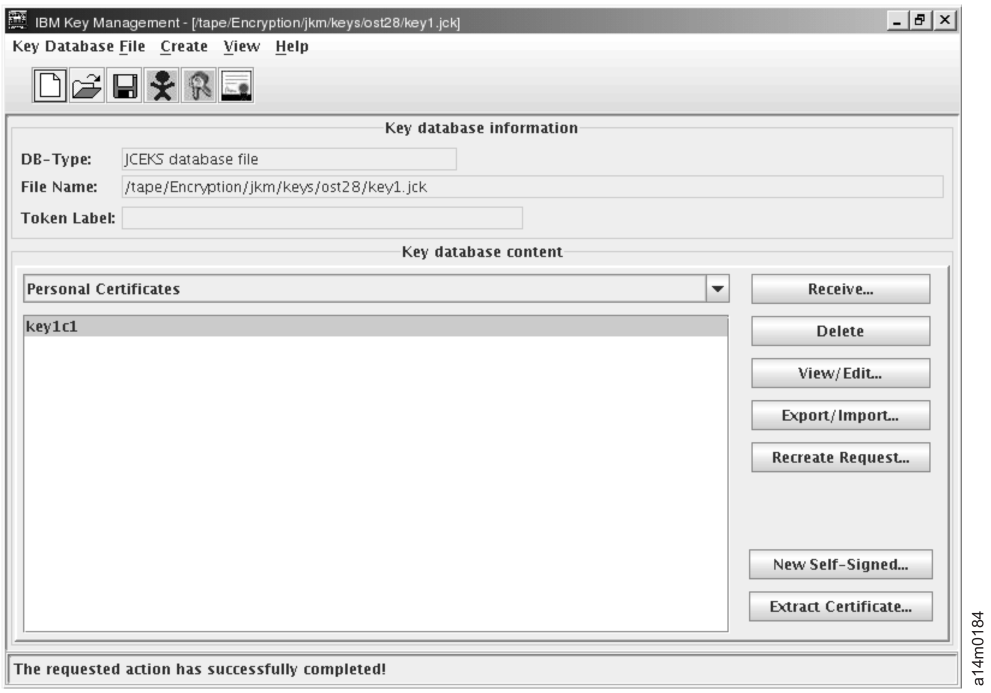
Please provide the following:

Key Label	key1c1
Version	X509 V3
Key Size	1024
Common Name	ost28.storage.tucson.ibm.com
Organization	ibm
Organization Unit (optional)	
Locality (optional)	
State/Province (optional)	
Zipcode (optional)	
Country or region	US
Validity Period	365 Days

Figure 29. Create new self signed certificate window

Click **OK**.

8. The “IBM Key Management” window (Figure 30) opens showing the new certificate.



Key database information

DB-Type: JCEKS database file

File Name: /tape/Encryption/jkm/keys/ost28/key1.jck

Token Label:

Key database content

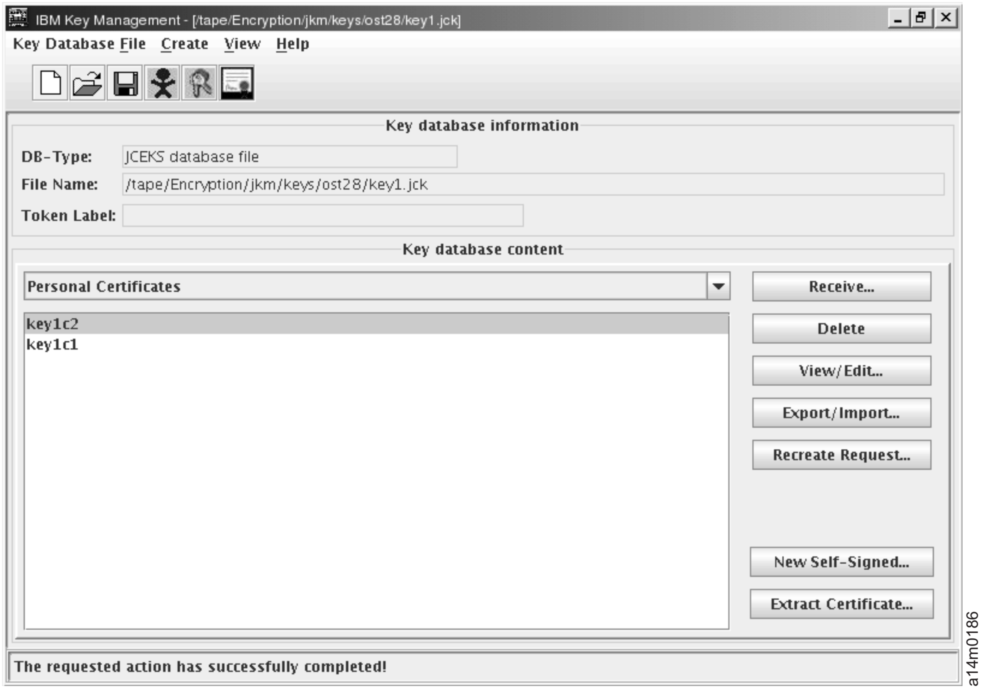
Personal Certificates

key1c1

The requested action has successfully completed!

Figure 30. ikeyman Key Management window showing new certificate

9. Repeat steps 6 on page 89 and 7 on page 90. In this example, we created two certificates, shown in Figure 31.



Key database information

DB-Type: JCEKS database file

File Name: /tape/Encryption/jkm/keys/ost28/key1.jck

Token Label:

Key database content

Personal Certificates

key1c2
key1c1

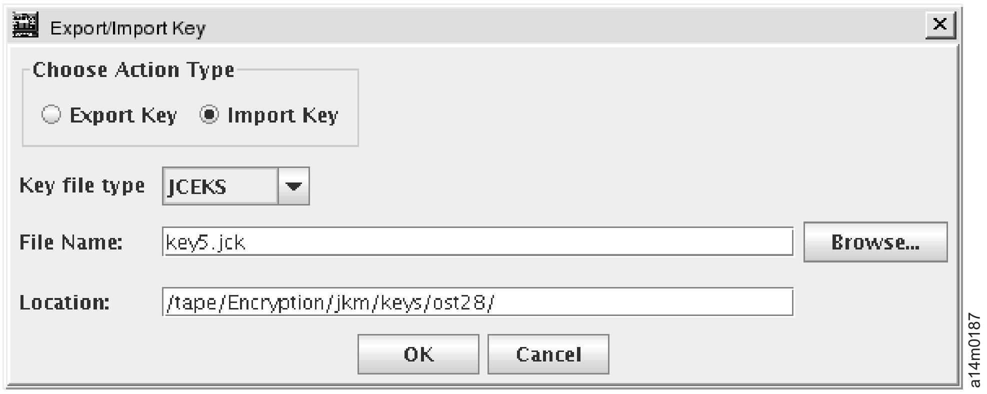
The requested action has successfully completed!

Figure 31. ikeyman Key Management window showing two new certificates

To import certificates that were created in another instance of ikeyman, click **Export/Import** on the right.

10. In the "Export/Import Key" window (Figure 32), select the following:
    - the **Import Key** button
    - Key file type **JCEKS**
    - Specify the name of the keystore from which to import keys (keys5.jck in our example) and its location.

    Select **OK**.


Export/Import Key

Choose Action Type

Export Key [✓ Import Key]

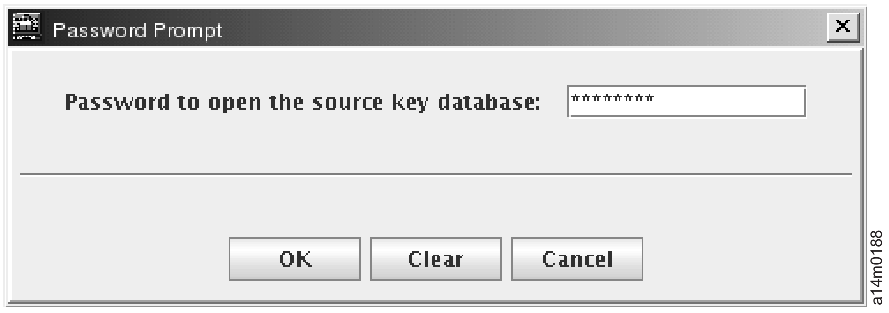
Key file type: JCEKS

File Name: key5.jck

Location: /tape/Encryption/jkm/keys/ost28/

Figure 32. Export/Import Key window

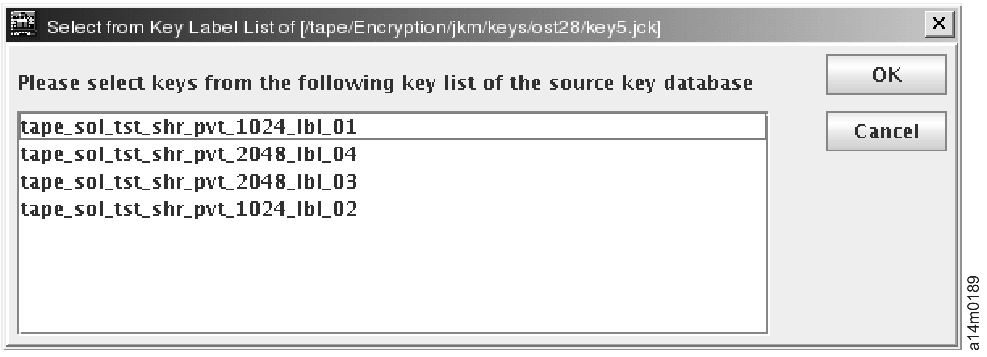
11. The "Password" window opens (Figure 33). Enter the required password and Click **OK**.


Password Prompt

Password to open the source key database: *********

Figure 33. Import Key Password window

12. A list of key labels in the specified keystore is displayed (Figure 33 on page 92).


Select from Key Label List of [/tape/Encryption/jkm/keys/ost28/key5.jck]
OK
Cancel
a14m0189

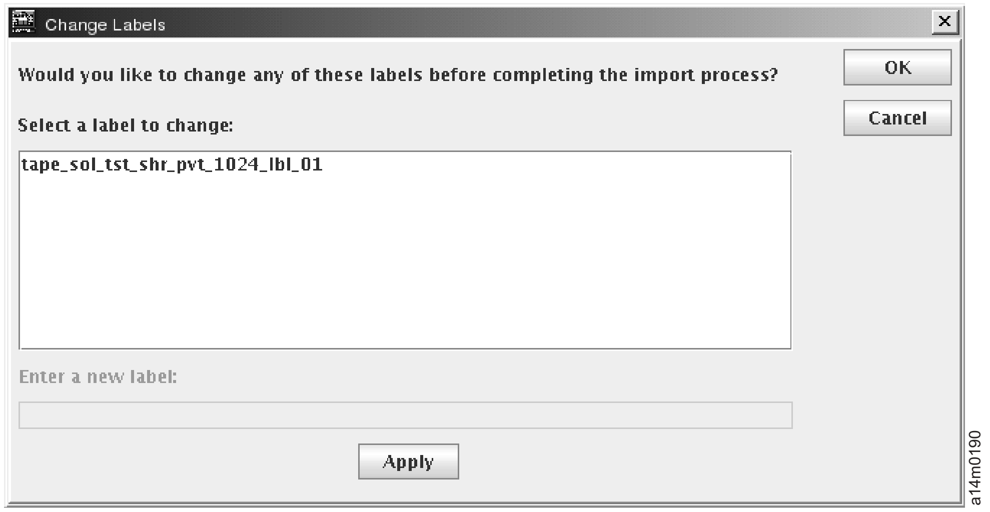
Please select keys from the following key list of the source key database

tape_sol_tst_shr_pvt_1024_lbl_01
tape_sol_tst_shr_pvt_2048_lbl_04
tape_sol_tst_shr_pvt_2048_lbl_03
tape_sol_tst_shr_pvt_1024_lbl_02

Figure 34. Key label list

Select one or more for import. In our example, we selected the first one. Click **OK**.

13. The “Change Labels” window opens (Figure 33 on page 92). If you wish to change the key label, select it, enter a new label name in the field below, and click **Apply**.



Would you like to change any of these labels before completing the import process?

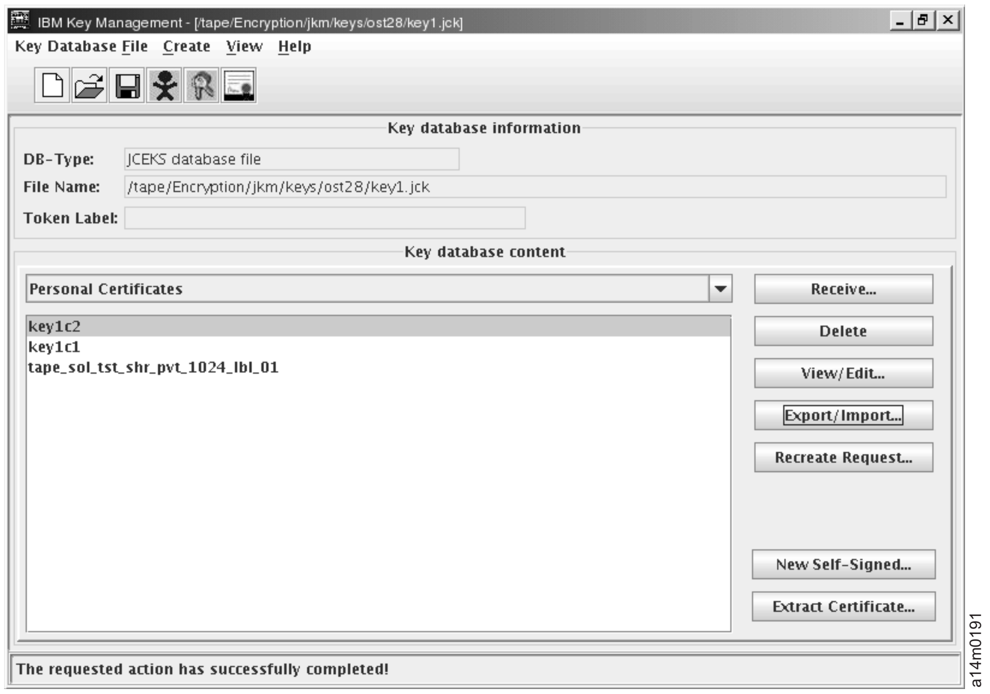
Select a label to change:

tape_sol_tst_shr_pvt_1024_lbl_01

Figure 35. Change Key Label window

Click **OK**.

14. The “IBM Key Management” window (Figure 36) displays the two certificates we created and the one we imported.



DB-Type: JCEKS database file
File Name: /tape/Encryption/jkm/keys/ost28/key1.jck
Token Label:

Personal Certificates

```
key1c2
key1c1
tape_sol_tst_shr_pvt_1024_lbl_01
```

The requested action has successfully completed!

Figure 36. ikeyman Key Management window showing three certificates

15. Click **Key Database > Close** and exit ikeyman.

Generating Keys and Aliases for Encryption on LTO 4

Keytool is the preferred utility for managing keys, certificates, and aliases. It enables you to generate, import, and export your encryption data keys and store them in a keystore.

Note: On Solaris and HP-UX, issuing keytool does not load the necessary IBM classes. The command you must use when running on Solaris and HP-UX is `java com.ibm.crypto.tools.KeyTool` with the same parameters.

Each data key in the keystore is accessed through a unique alias. An alias is a string of characters, such as `123456tape`. In JCEKS and most other keystores, `123456Tape` would be equivalent to `123456tape` and allow access to the same entry in the keystore. Check the documentation for your keystore type to determine whether it is case-sensitive. When you use the **keytool -genseckey** command to generate a data key, you specify a corresponding alias in the same command. The alias enables you to identify the correct key, in the correct key group and keystore, for use in writing and reading encrypted data on LTO 4 tape.

Note: Individual aliases and alias ranges must be unique. This is enforced when keys are generated on a given keystore/Encryption Key Manager instance. However, in a multiple Encryption Key Manager/Keystore environment, you should use a naming convention that maintains uniqueness across multiple instances in the event it becomes desirable to transport keys between instances while maintaining uniqueness of reference.

After generating keys and aliases, be sure to update the symmetricKeySet property in the KeyManagerConfig.properties file to specify the new alias, range of aliases, or key group GroupID, the filename under which the symmetric keys are stored, and the filename where key groups are defined. (See "Creating and Managing Key Groups" on page 100 for details.) Only those keys named in the symmetricKeySet will be validated (checked for an existing alias and a symmetric key of the proper size and algorithm). If an invalid key is specified in this property, the key manager does not start and an audit record is created.

The keytool utility also provides for the import and export of data keys to and from other keystores. An overview of each task follows. You can issue the **keytool -ekmhelp** to display all the key manager-related parameters covered in the following discussions.

If You are Not Using Keytool

If you use a utility other than keytool to generate keys and aliases, you cannot generate ranges of keys compatible with the Encryption Key Manager. To generate individual keys compatible with the Encryption Key Manager, be sure to specify aliases using one of the following formats:

- 12 printable characters or less (for example, `abcdefghijk`)
- 3 printable characters, followed by two zeros, followed by 16 hexadecimal digits (for example, `ABC000000000000000001`) for a total of exactly 21 characters

Generating Data Keys and Aliases Using Keytool -genseckey

The Keytool utility generates aliases and symmetric keys for encryption on LTO Ultrium 4 Tape Drives using LTO 4 tape. Use the **keytool -genseckey** command to generate one or more secret keys and store them in a specified keystore. **keytool -genseckey** takes the following parameters:

```
-genseckey   [-v] [-protected]
             [-alias <alias> | aliasrange <aliasRange>] [-keypass <keypass>]
             [-keyalg <keyalg>] [-keysize <keysize>]
             [-keystore <keystore>] [-storepass <storepass>]
             [-storetype <storetype>] [-providerName <name>]
             [-providerClass <provider_class_name> [-providerArg <arg>] ...
             [-providerPath <pathlist>]
```

These parameters are of particular importance when generating data keys for Encryption Key Manager to serve to the LTO 4 drives for tape encryption:

-alias
Specify an *alias* value for a single data key with up to 12 printable characters (for example, `abcfrg` or `key123tape`).

-aliasrange
When generating multiple data keys, *aliasrange* is specified as a 3-character alphabetic prefix followed by lower and upper limits for a series of 16-character (hexadecimal) strings with leading zeroes filled in automatically to construct aliases 21-characters in length. For example, specifying `key1-a` would yield a series of aliases from `KEY000000000000000001` through `KEY00000000000000000A`. Specifying an *aliasrange* value of `xyz01-FF` would yield `XYZ000000000000000001` through `XYZ0000000000000000FF` , which would generate 255 symmetric keys.

-keypass
Specifies a password used to protect the data key. This password **must be identical** to the keystore password. If no password is specified, you are prompted for it. If you press **Enter** at the prompt, the key password is set to the same password as that used for the keystore. *keypass* must be at least six characters long.

Note: Once you have set the keystore password, **do not change** it unless it's security has been breached. See "Changing Keystore Passwords."

-keyalg
Specifies the alogrithm to be used to generate the data key. On an Open Systems platform, the key algorithm must be specified as `AES`. If the encrypted tape will be shared with systems on the z/OS platform and the zOSCompatibility property is set to true, the key algorithm should be specified as `DESede` to ensure compatibility. In addition, if generating symmetric keys on z/OS with keystore type other than JCEKS (i.e., JCE4758KS/JCECCAKS) then you must specify `DESede`.

-keysize
Specifies the size of the data key to be generated. On an Open Systems platform, the key size must be specified as `256`. If -keyalg is specified as `DESede` for z/OS compatibility, then -keysize should be allowed to default to 168.

These parameters are of particular importance when using a hardware cryptographic card as the keystore (keystore type = PKCS11ImplKS):

-providerClass
Specifies the PKCS11 provider class. This should always be `com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl`.

-providerArg
Specifies the configuration file used to initialize the PKCS11Impl provider and open a session with the hardware cryptographic card. For more information about the contents of this file, check the Java 5.0 documentation: http://www-128.ibm.com/developerworks/java/jdk/security/50/.

Examples of acceptable aliases that could be associated with symmetric keys are:

```
abc000000000000000001
abc00a0120fa000000001
```

Examples of aliases that would not be accepted by the key manager are:

```
abcefghij1234567 ? wrong length
abcg0000000000000001 ? prefix is longer than 3 characters
```

If an alias already exists in the keystore, keytool throws an exception and stops.

Changing Keystore Passwords

Note: Once you have set the keystore password, **do not change** it unless it's security has been breached. The passwords are obfuscated to eliminate any security exposure. Changing the keystore password requires that the password on every key in that keystore be changed individually using the following **keytool** command.

To change the keystore password enter:

```
keytool -keypasswd  -keypass old_passwd -new new_passwd -alias alias
        -keystore keystorename -storetype keystoretype
```

You must also edit KeyManagerConfig.properties to change the keystore password in every server configuration file property where it is specified using one of these methods:

- Delete the entire obfuscated password and allow the Encryption Key Manager to prompt on the next startup.
- Delete the entire obfuscated password and type the new password in the clear. It will be obfuscated on the next startup.

Importing Data Keys Using Keytool -importseckey

Use the keytool -importseckey command to import a secret key or a batch of secret keys from an import file. **keytool -importseckey** takes the following parameters:

```
-importseckey        [-v]
              [-keyalias <keyalias>] [-keypass <keypass>]
              [-keystore <keystore>] [-storepass <storepass>]
              [-storetype <storetype>] [-providerName <name>]
              [-importfile <importfile>] [-providerClass <provider_class_name>]
              [providerArg <arg>]
```

These parameters are of particular importance when importing data keys for the Encryption Key Manager to serve to the LTO 4 drives for tape encryption:

-keyalias
Specifies the alias of a private key in keystore to decrypt all the data keys in *importfile*.

-importfile
Specifies the file that contains the data keys to be imported.

These parameters are also of particular importance when using a hardware cryptographic card as the keystore (keystore type = PKCS11ImplKS):

-providerClass
Specifies the PKCS11 provider class. This should always be `com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl`.

-providerArg
Specifies the configuration file used to initialize the PKCS11Impl provider and open a session with the hardware cryptographic card.

Note: On Java 1.42, use of a hardware cryptographic card for the Encryption Key Manager keystore is only supported by JCE4758KS.

Exporting Data Keys Using Keytool -exportseckey

Use the keytool -exportseckey command to export a secret key or a batch of secret keys to an export file. **keytool -exportseckey** takes the following parameters:

```
-exportseckey        [-v]
              [-alias <alias> | aliasrange <aliasRange>] [-keyalias <keyalias>]
              [-keystore <keystore>] [-storepass <storepass>]
              [-storetype <storetype>] [-providerName <name>]
              [-exportfile <exportfile>] [-providerClass <provider_class_name>]
              [providerArg <arg>]
```

These parameters are of particular importance when exporting data keys for Encryption Key Manager to serve to the LTO 4 drives for tape encryption:

-alias
Specify an *alias* value for a single data key with up to 12 printable characters (for example, `abcfrg` or `key123tape`).

-aliasrange
When exporting multiple data keys, *aliasrange* is specified as a 3-character alphabetic prefix followed by lower and upper limits for a series of 16-character (hexadecimal) strings with leading zeroes filled in automatically to construct aliases 21-characters in length. For example, specifying `key1-a` would yield a series of aliases from `KEY000000000000000001` through `KEY00000000000000000A`. Specifying an *aliasrange* value of `xyz01-FF` would yield `XYZ000000000000000001` through `XYZ0000000000000000FF`

-exportfile
Specifies the file to store the data keys when they are exported.

-keyalias
Specifies the alias of a public key in keystore to encrypt all the data keys. Ensure that the keystore where the symmetric (data) keys will be imported contains the corresponding private key.

The following parameters are of importance if using the IBMPKCS11ImplKS type for your keystore (keystore type = PKCS11ImplKS):

-providerClass
Specifies the PKCS11 provider class when using a hardware cryptographic device. This should always be `com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl`.

-providerArg
Specifies the configuration file used to initialize the PKCS11Impl provider and open a session with the hardware cryptographic card.

Note: On Java 1.42, use of a hardware cryptographic card for the Encryption Key Manager keystore is only supported by JCE4758KS.

keytool -exportseckey takes a slightly different option in SDK 1.4.2. It requires that -keypass *keypass* be specified. This is because the keystore requires -keypass (*keypassword*) to retrieve a secret key, indicating that all the secret keys share the same key password.

Sample Alias and Symmetric Key Setup for LTO 4 Encryption Using a JCEKS Keystore

Invoke the **KeyTool** with the `-aliasrange` option.

Note that on an Open Systems platform, key algorithm (-keyalg) must be specified as AES and key size (–keysize) must be specified as `256`, as follows:

```
/bin/keytool –genseckey –v –aliasrange AES01-FF –keyalg AES –keysize 256
–keypass password -storetype jceks –keystore path/filename.jceks
```

or

If the encrypted tape will be shared with systems on the z/OS platform and zOSCompatibility property is set to true, or your Encryption Key Manager keystore is of JCE4758KS/JCECCAKS keystore type, then key algorithm (-keyalg) must be specified as `DESede`.

```
keytool –genseckey –v –aliasrange AES01-FF –keyalg DESede
–keypass password -storetype jceks –keystore path/filename.jceks
```

These KeyTool invocations generate 255 sequential aliases in the range AES000000000000000001 through AES0000000000000000FF and associated AES 256-bit symmetric keys. Either can be repeated cumulatively as many times as necessary to setup the full number of ranged and standalone key aliases that are desired for robust key manager operation. For example, to generate an additional alias and symmetric key for LTO 4 on an Open Systems platform:

```
/bin/keytool –genseckey –v –alias abcfrg –keyalg AES –keysize 256
–keypass password -storetype jceks –keystore path/filename.jceks
```

This invocation adds standalone alias `abcfrg` cumulatively to the named keystore, which already contains 255 aliases from the Open Systems platform invocation above yielding 256 symmetric keys in the jceks file named in –keystore option.

Update the symmetricKeySet property in the KeyManagerConfig.properties file to add the following line to match any or all of the alias ranges used above, and the filename under which the symmetric keys were stored. Note that the Encryption Key Manager may not start if an invalid alias is specified. Other causes for validation check failure may include incorrect bit size (for AES keysize MUST be 256) or an invalid algorithm for the platform. On z/OS -keyalg must be DESede (3-DES). On Open Systems, -keyalg must be AES and -keysize must be 256. The filename specified in the **config.keystore.file** should match the name specified in the –keystore <filename> in the KeyTool invocation:

symmetricKeySet = AES01-FF,abcfrg
config.keystore.file = <filename>.jceks

Only those keys named in the symmetricKeySet will be validated (checked for an existing alias and a symmetric key of the proper size and algorithm). If an invalid key is specified in this property, the Encryption Key Manager will not start and an audit record will be created.

Sample Alias and Symmetric Key Setup for LTO 4 Encryption Using a PKCS11ImplKS Keystore

This is example is very similar to the JCEKS example above with the exception that a hardware keystore is involved.

Invoke the **KeyTool** with the `-aliasrange` option.

```
/bin/keytool –genseckey –v –aliasrange AES01-FF –keyalg AES –keysize 256
–keypass password -storetype PKSC11ImplKS
–keystore path/filename_of_vendor's_PKCS11_interface_to_the_hardware
-providerClass com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
-providerArg path/filename_of_vendor's_PKCS11configuration_file
```

This KeyTool invocation generates 255 sequential aliases in the range AES000000000000000001 through AES0000000000000000FF and associated AES 256-bit symmetric keys.

Update the symmetricKeySet property in the KeyManagerConfig.properties file to add the following line to match any or all of the alias ranges used above, and the filename under which the symmetric keys were stored. Note that the Encryption Key Manager may not start if an invalid alias is specified. Other causes for validation check failure may include incorrect bit size (for AES keysize MUST be 256) or an invalid algorithm for the platform. On z/OS -keyalg must be DESede (3-DES). On an Open Systems platform, -keyalg must be AES and -keysize must be 256. The filename specified in the config.keystore.file should match the name specified in the –keystore <filename> in the KeyTool invocation:

symmetricKeySet = AES01-FF,abcfrg
config.keystore.file = *filename_of_vendor's_PKCS11_interface_to_the_hardware*
  for example, config.keystore.file = /opt/nfast/toolkits/pkcs11/libcknfast.so

Only those keys named in the symmetricKeySet are validated (checked for an existing alias and a symmetric key of the proper size and algorithm). If an invalid key is specified in this property, the Encryption Key Manager does not start and an audit record is created.

Creating and Managing Key Groups

The Encryption Key Manager gives you the ability to organize your symmetric keys for LTO 4 encryption into key groups. In this way, you can group keys according to the type of data they encrypt, the users who have access to them, or by any other meaningful characteristic. Once a key group is created, you can associate it with a specific tape drive using the -symrec keyword in the **adddrive** command. See "adddrive" on page 161 for syntax.

In order to build a key group, you must define it in the KeyGroups.xml file. If you are creating the configuration file manually, the location of the KeyGroups.xml file is specified in the configuration properties file as follows:

```
config.keygroup.xml.file = FILE:KeyGroups.xml
```

If this parameter is not specified, then the default behavior is to use the KeyGroups.xml file from the the Encryption Key Manager launching location's working directory. If this file does not exist, an empty KeyGroups.xml file is created. On subsequent starts of the Encryption Key Manager Server, the following message may appear in the **native_stderr.log**: `[Fatal Error] :-1:-1: Premature end of file.` This is an error in parsing the empty KeyGroups.xml file and it does not prevent the Encryption Key Manager Server from starting unless the Encryption Key Manager Server has been configured to use keygroups.

Key groups are built using the following CLI client commands (see "CLI Commands" on page 161 for syntax):

Using CLI Commands to Define Key Groups

The Encryption Key Manager has a key group feature that allows you to group sets of keys.

Once the Encryption Key Manager application is installed and configured (keystore and keys generated) and the Encryption Key Manager server is started, log in to into the server using the client and follow these steps:

1. Run the **createkeygroup** command.

This command creates the initial key group object in the KeyGroups.xml file. Run this only once.

Syntax: **createkeygroup -password** *password*

-password
The *password* that is used to encrypt the keystore's password in the KeyGroups.xml file for later retrieval. The keystore encrypts the key group's key, which in turn encrypts each individual key group alias password. Therefore no key in the KeyGroups.xml file is in the clear.

Example: createkeygroup -password a75xynrd

2. Run the **addkeygroup** command.

   This command creates an instance of a key group with a unique Group ID in the KeyGroups.xml.

   Syntax: **addkeygroup -groupID** *groupname*

   -groupID
   The unique *groupname* used to identify the group in the KeyGroups.xml file.

   Example: addkeygroup -groupID keygroup1

3. Run the **addkeygroupalias** command.

   This command creates a new alias for an existing key alias in your keystore for addition to a specific key group ID.

   Syntax: **addkeygroupalias -alias** *aliasname* **-groupID** *groupname*

   -alias
   The new *aliasname* for the key. This must be the full key name, that is, Key00 must be entered as key000000000000000000.

   -groupID
   The unique *groupname* used to identify the group in the KeyGroups.xml file.

   Example: addkeygroupalias -alias key000000000000000000 -groupID keygroup1

   Note: When using this CLI command, you can only add one key at time. This command must be run for every individual key that needs to be added to the key group.

4. Associate a key group with a new or existing tape drive.

   a. Run the **moddrive** command to associate a key group with an existing tape drive.

      This command modifies tape drive information in the drive table.

      Syntax: **moddrive -drivename** *drivename* **-symrec** *alias*

      -drivename
      drivename specifies the serial number of the tape drive.

      -symrec
      Specifies an *alias* (of the symmetric key) or a key group name for the tape drive.

      Example: moddrive -drivename 000123456789 -symrec keygroup1

   b. Run the **adddrive** command to add a tape drive to the drive table and associate it with a key group.

      This command allows you to add a drive and associate it with a specific key group.

      Syntax: **adddrive -drivename** *drivename* **-symrec** *alias*

-drivename
drivename specifies the 12-digit serial number of the drive to be added.

-symrec
Specifies an *alias* (of the symmetric key) or a groupID for the tape drive.

Example: adddrive -drivename 000123456789 -symrec keygroup1

To specify a key group as default for use when no alias is defined for a tape drive, set the symmetrickeySet property of the configuration properties file to the GroupID of the key group you wish to use. For example,

```
symmetricKeySet = keygroup1
```

The GroupID must match an existing key group ID in the KeyGroups.xml file. If not, the Encryption Key Manager Server will not start. The Encryption Key Manager tracks key usage within a key group. When you specify a valid GroupID, the Encryption Key Manager records which key was last used and then selects a random key from within the specified key group.

Copying Keys From One Key Group to Another

Run **addaliastogroup** command.

This command copies a specific alias from an existing (source) key group to a new (target) key group.

Syntax: **addaliastogroup -aliasID** *aliasname* **-sourceGroupID** *groupname* **-targetGroupID** *groupname*

-aliasID
The *aliasname* for the key to be added.

-sourceGroupID
The unique *groupname* used to identify the group from which the alias is to be copied.

-targetGroupID
The unique *groupname* used to identify the group to which the alias is to be added.

Example: addaliastogroup -aliasID aliasname -sourceGroupID keygroup1 -targetGroupID keygroup2

Note: Key is available in both key groups.

Installing on i5/OS

Install the IBM Software Developer Kit (i5/OS)

On i5/OS V5R3

1. Refer to this Software Knowledge Base document for any updates to the install processing: http://www-912.ibm.com/8625680A007CA5C6/1AC66549A21402188625680B0002037E/99ECAE271619B371862571CC0006652D.
2. Install 5722JV1 *BASE and 5722JV1 Option 7
3. Install 5722AC3
4. Install the latest Java PTFs (PTF Group SF99269)
5. Install PTF 5722JV1 SI24671 - This PTF installs the unrestricted policy files.
6. Install PTF 5722SS1 SI26705 - This PTF installs the Encryption Key Manager code, the default configuration properties file, and the script file.

   Note: PTF 5722SS1 SI26705 will install the 10242006 version of the Encryption Key Manager application. No more PTFs will be provided for V5R3 to update the Encryption Key Manager application. You can download a more recent version of the Encryption Key Manager application from this URL (http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504). The Encryption Key Manager application (**IBMKeyManagementServer.jar**) must be downloaded to the **/QIBM/ProdData/OS400/Java400/ext/** directory, replacing the version of the application that is currently in the directory.

   The strEKM script shipped with this PTF will not support any Encryption Key Manager versions after the 10242006 version . If a future version of the code is downloaded, add the following symbolic links:

   ```
   ln -s /QIBM/ProdData/OS400/Java400/ext/IBMKeyManagementServer.jar
      /QIBM/UserData/Java400/ext/IBMKeyManagementServer.jar
   ln -s /QIBM/ProdData/OS400/Java400/ext/ibmkeycert.jar
      /QIBM/UserData/Java400/ext/ibmkeycert.jar
   ```

   To start the Encryption Key Manager server or command line interface, use the java command as it is documented elsewhere in this manual.
7. Install PTF 5722JV1 SI28159 - This PTF installs the updates to KeyTool to allow the creation of symmetric keys.
8. Edit the **/QIBM/ProdData/Java400/jdk15/lib/security/java.security** file to add these providers (the number specified will depend on which numbers are already specified in the file):

   ```
   security.provider.6=com.ibm.jsse2.IBMJSSEProvider2
   security.provider.7=com.ibm.i5os.jsse.JSSEProvide
   ```

On i5/OS V5R4

1. Refer to this Software Knowledge Base document for any updates to the install processing: http://www-912.ibm.com/8625680A007CA5C6/1AC66549A21402188625680B0002037E/99ECAE271619B371862571CC0006652D.
2. Install 5722JV1 *BASE and 5722JV1 Option 8
3. Install the latest Java PTFs (PTF Group SF99291)
4. Install SR5 (PTF 5722JV1 SI27562) and all of its requisites

   Note: PTF 5722JV1 SI27562 will install the 05032007 version of the Encryption Key Manager application. You can download a more recent version of the Encryption Key Manager application from this URL (http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504). The Encryption Key Manager application (**IBMKeyManagementServer.jar**) must be downloaded to the **/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/ext/** directory, replacing the version of the application that is currently in the directory.

.

5. Install PTF 5722SS1 SI25094 - This PTF will install the default configuration properties file and the script file.

   Note: The strEKM script shipped with this PTF will not support any Encryption Key Manager versions after the 05032007 version . If a later version of the code is used, set the JAVA_HOME environment variable to /QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit:

   ```
   ADDENVVAR ENVVAR(JAVA_HOME)
   VALUE('/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit')
   ```

   Then start the Then start the EKM server or command line interface using the java command as it is documented elsewhere in this manual.
6. Edit the **/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/security/java.security** file to add these providers (the number specified will be dependent on which numbers are already specified in the file):

   ```
   security.provider.6=com.ibm.jsse2.IBMJSSEProvider2
   security.provider.7=com.ibm.i5os.jsse.JSSEProvider
   ```

On IBM i V6R1

1. Refer to this Software Knowledge Base document for any updates to the install processing: http://www-912.ibm.com/8625680A007CA5C6/1AC66549A21402188625680B0002037E/99ECAE271619B371862571CC0006652D.
2. Install 5761JV1 *BASE and 5761JV1 Option 8
3. Install the latest Java PTFs (PTF Group SF99562)
4. Edit the **/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/security/java.security** file to add these providers (the number specified will be dependent on which numbers are already specified in the file):

   ```
   security.provider.6=com.ibm.jsse2.IBMJSSEProvider2
   security.provider.7=com.ibm.i5os.jsse.JSSEProvider
   ```

Create and Manage Keystores on i5/OS

Before configuring and launching the Encryption Key Manager, you must create at least one new keystore. The types of encrypting tape drives that you will be supporting determines the type of keystore that you should create.

- If you are supporting only TS1120 or TS1130 tape drives, then you can create either a JCEKS or IBMi5OSKeystore keystore.
- If you are supporting only LTO 4 tape drives, or you have one tape library that supports both LTO 4 and TS1120 or TS1130 tape drives, then you must create a JCEKS keystore.
- If you have separate tape libraries for the LTO 4 and TS1120 or TS1130 tape drives, then you can setup two Encryption Key Manager servers for each of the tape libraries. One Encryption Key Manager server can run using an IBMi5OSKeystore for use with the TS1120 or TS1130 tape drive and the other server can run using a JCEKS for use with the LTO 4 tape drive. The two Encryption Key Manager servers can run on the same system as long as they listen on different ports.

You can use Digital Certificate Manager (DCM) to manage IBMi5OSKeystore keystores. Refer to "Using Digital Certificate Manager to Create and Manage Keystores on i5/OS" for details. Use **keytool** to manage a JCEKS.

The keytool interface is the only way to create symmetric keys for use with LTO 4 tape drives. Refer to "Using Keytool on i5/OS."

Using Keytool on i5/OS

Be sure to use the correct version of keytool to be able to do symmetric key processing. The keytool script on i5/OS points to the **sun.security.tools.KeyTool** version. This will not have the updates necessary for doing symmetric key processing. To invoke the correct version, use the appropriate version of this java command:

For V5R3:
`/QIBM/ProdData/Java400/jdk15/bin/java com.ibm.crypto.tools.KeyTool ....`

For V5R4:
`/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/bin/java com.ibm.crypto.tools.KeyTool ....` You can also set the JAVA_HOME environment variable to `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit` and then invoke by issuing **keytool**.

Note: If your current java version is already set to the correct version, then you can just enter `java com.ibm.crypto.tools.KeyTool ...`

Refer to "Generating Keys and Aliases for Encryption on LTO 4" on page 94 for details on using keytool to generate symmetric keys for use with LTO 4 tape drives.

If you are currently using an IBMi5OSKeystore keystore and need to change to a JCEKS keystore, you can use DCM to export the certificates needed by the Encryption Key Manager to files. You can then use keytool or ikeyman to import the certificate into the JCEKS keystore.

Using Digital Certificate Manager to Create and Manage Keystores on i5/OS

Before configuring and launching the Encryption Key Manager, you must create at least one new keystore and at least one self-signed certificate.

To create a keystore, follow the tasks under "Create Keyring/Keystore Instance." To create a self-signed certificate, follow the tasks under "Create Self-Signed Certificate (for Internal Use)" on page 114, which includes creating a Local CA certificate and a certificate signed by the Local CA certificate. The tasks under "Generate Public/Private Key Pair" on page 108 only need be performed to create external certificates used to share encrypted tapes outside of your company.

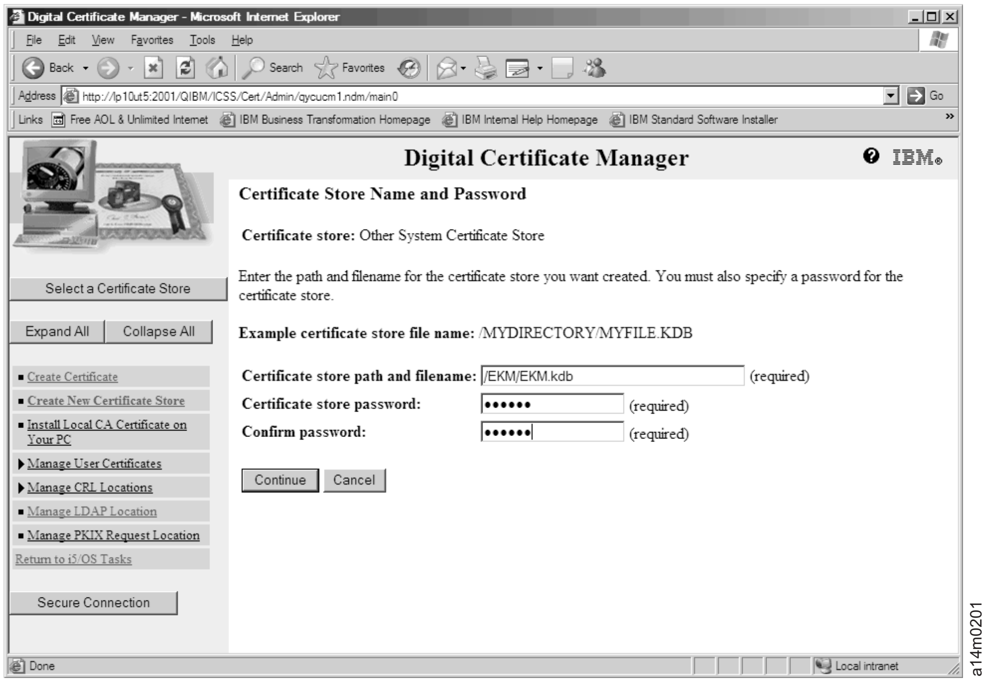
Visit http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp for information about installing Digital Certificate Manager (DCM).

The following tasks assume that DCM is started and the home window opens.

Create Keyring/Keystore Instance

1. Select **Create New Certificate Store** from the menu on the left.
2. In the "Create a New Certificate Store" window, click the **Other System Certificate Store** button and click **Continue**.

3. In the “Create a Certificate in New Certificate Store” window, accept the default value of No, and click **Continue**.
4. In the “Certificate Store Name and Password” window (Figure 37), specify a certificate store path and file name and click **Create**. This file name is also specified in your Encryption Key Manager configuration file.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Continue
Cancel
Done
Local intranet
a14m0201

Digital Certificate Manager

Certificate Store Name and Password

Certificate store: Other System Certificate Store

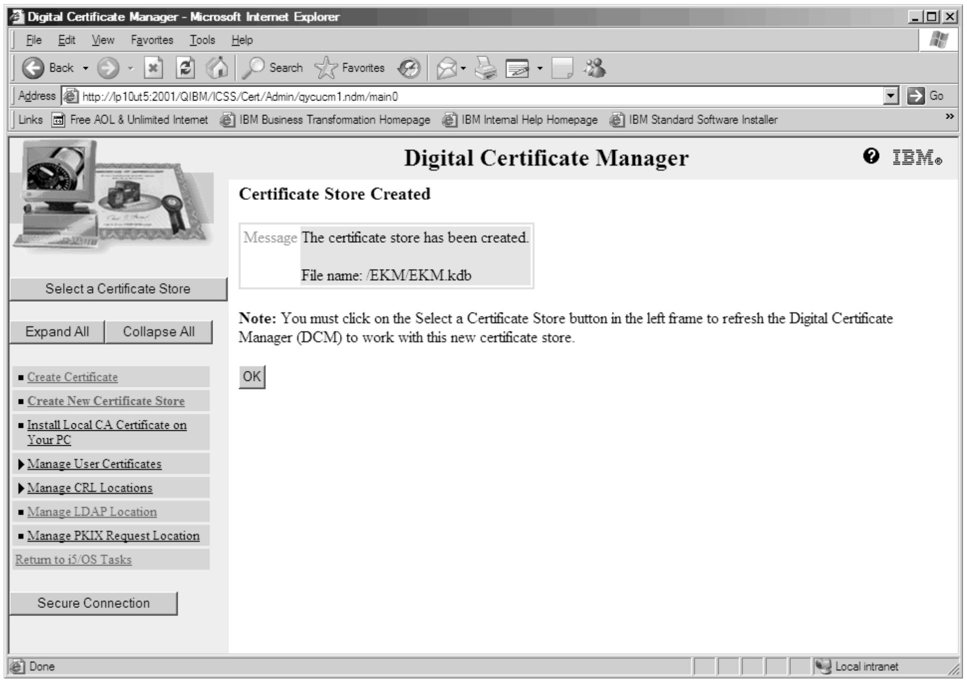
Enter the path and filename for the certificate store you want created. You must also specify a password for the certificate store.

Example certificate store file name: /MYDIRECTORY/MYFILE.KDB

Certificate store path and filename: /EKM/EKM.kdb (required)
Certificate store password: •••••• (required)
Confirm password: •••••• (required)

Figure 37. Certificate Store Name and Password window

5. The “Certificate Store Created” window opens. Select the **Select a Certificate Store** menu item to access the new keystore you just created.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
OK
Done
Local intranet
a14m0202

Certificate Store Created

Message The certificate store has been created.

File name: /EKM/EKM.kdb

Note: You must click on the Select a Certificate Store button in the left frame to refresh the Digital Certificate Manager (DCM) to work with this new certificate store.

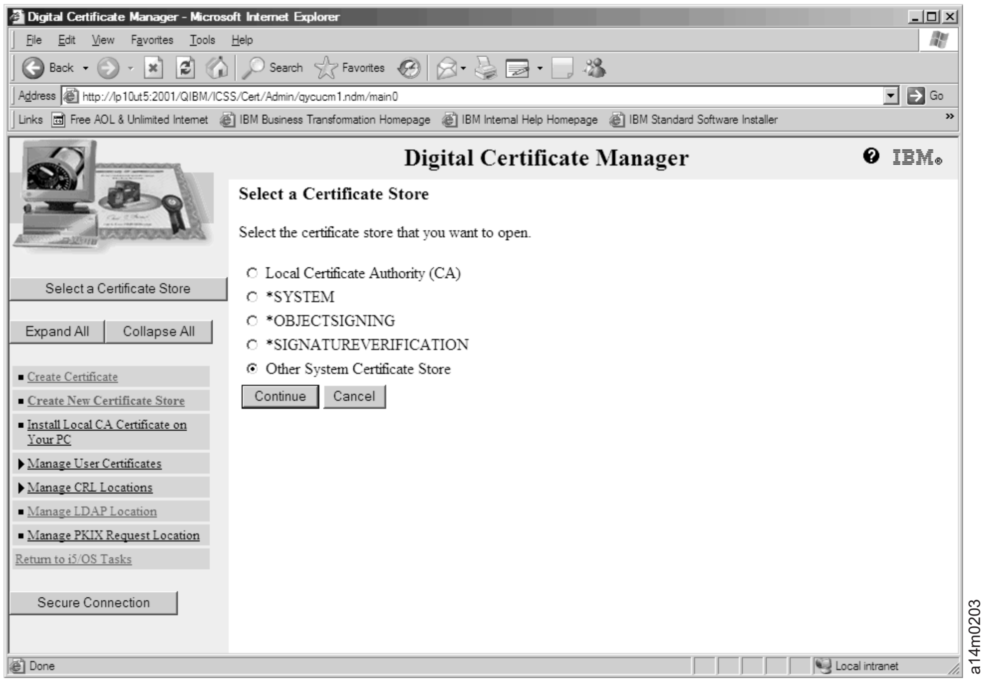
Figure 38. Certificate Store Created window

Generate Public/Private Key Pair

Perform these steps for as many public/private key pairs as needed. If you have multiple organizational identities within your company that need their own CA-signed certificates, create a public/private key pair for each.

These steps create a public/private key pair as well as a certificate request.

1. Select the **Select Certificate Store** menu item. In the Select Certificate Store window (Figure 39), click **Other System Certificate Store**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM.
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Continue
Cancel
Done
Local intranet
a14m0203

Select a Certificate Store

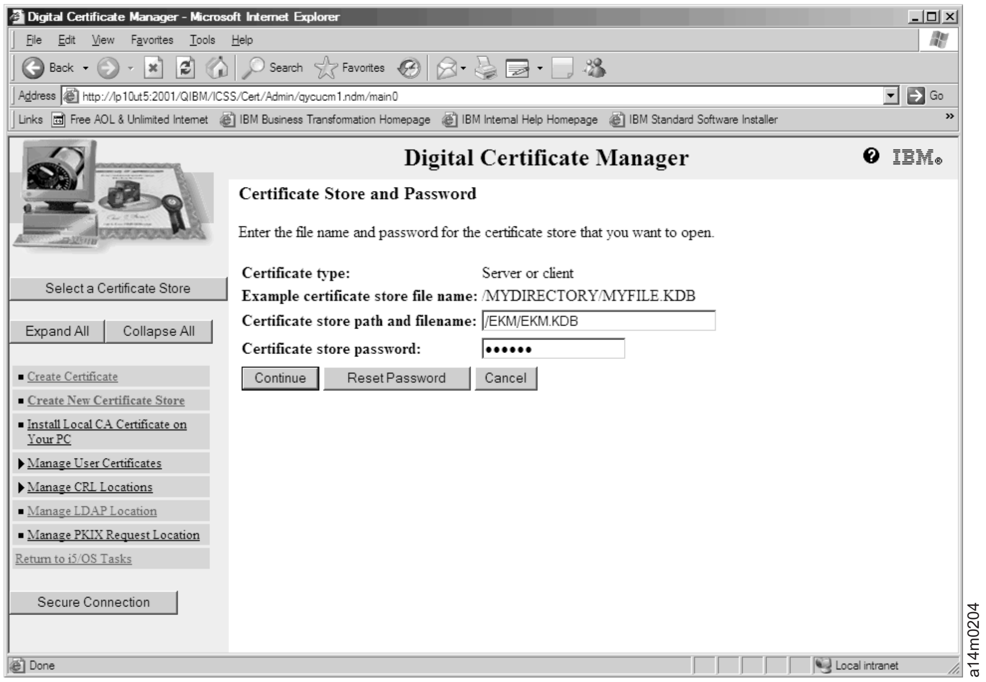
Select the certificate store that you want to open.

- Local Certificate Authority (CA)
- *SYSTEM
- *OBJECTSIGNING
- *SIGNATUREVERIFICATION
- [✓ Other System Certificate Store]

Figure 39. Select Certificate Store window

Select **Continue**.

2. The “Certificate Store and Password” window opens (Figure 40). Specify the path and filename you entered in “Create Keyring/Keystore Instance” on page 105.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Continue
Reset Password
Cancel
Done
Local intranet
a14m0204

Digital Certificate Manager

Certificate Store and Password

Enter the file name and password for the certificate store that you want to open.

Certificate type:	Server or client
Example certificate store file name:	/MYDIRECTORY/MYFILE.KDB
Certificate store path and filename:	/EKM/EKM.KDB
Certificate store password:	••••••

Figure 40. Certificate Store and Password window

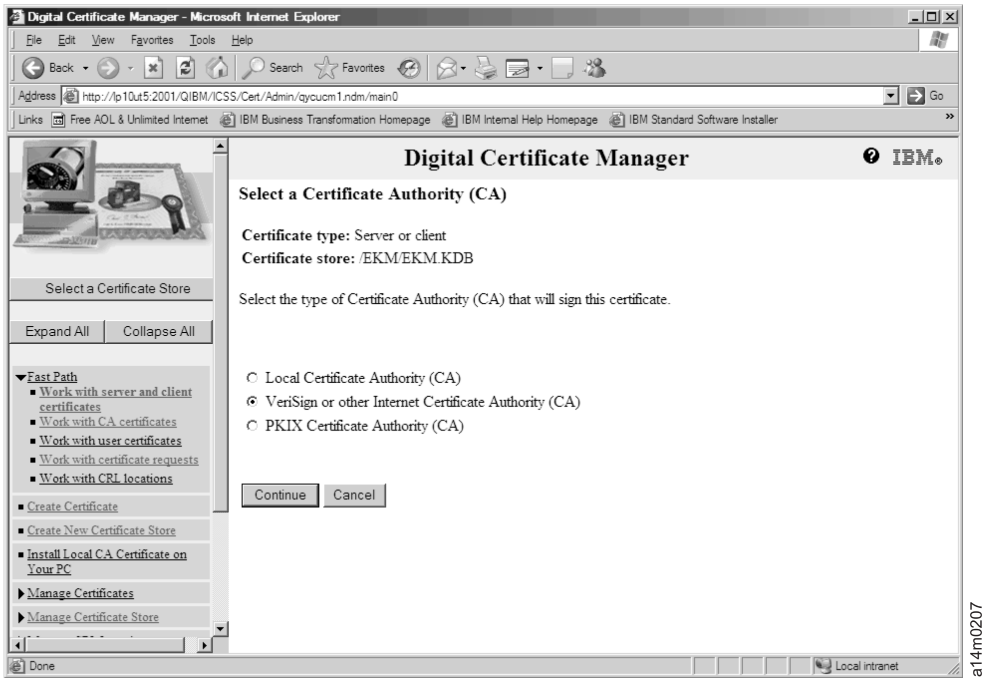
click **Continue**.

The “Current Certificate Store” window opens verifying your certificate store selection.

After you select a certificate store, you can use the **Work with Server and Client Certificates** option of the **Fast Path** menu group to perform all of the tasks, or use the various **Manage Certificates** menu group options to perform individual tasks.

3. Select **Create Certificate** menu item.

4. The “Select Certificate Authority” window opens (Figure 41).


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0207

Digital Certificate Manager

Select a Certificate Authority (CA)

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Select the type of Certificate Authority (CA) that will sign this certificate.

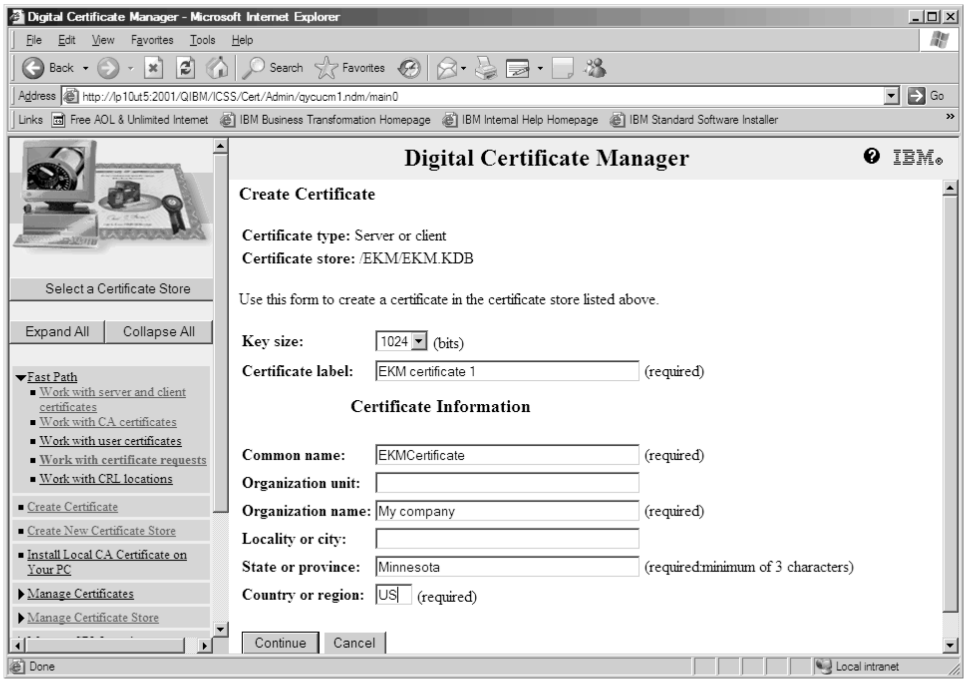
Local Certificate Authority (CA) [✓ VeriSign or other Internet Certificate Authority (CA)] PKIX Certificate Authority (CA)

Continue Cancel

Figure 41. Certificate Request Created window

Click the **VeriSign or other Internet Certificate Authority (CA)** button for the CA to sign the certificate and click **Continue**.

5. In the “Create Certificate” window (Figure 42 on page 111), specify a Key size of **2048** and a Certificate label value that will correspond to the alias1 key label value in your Encryption Key Manager configuration file. Fill in the other required values as appropriate.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0208

Digital Certificate Manager

Create Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Use this form to create a certificate in the certificate store listed above.

Key size: 1024 (bits)
Certificate label: EKM certificate 1 (required)

Certificate Information

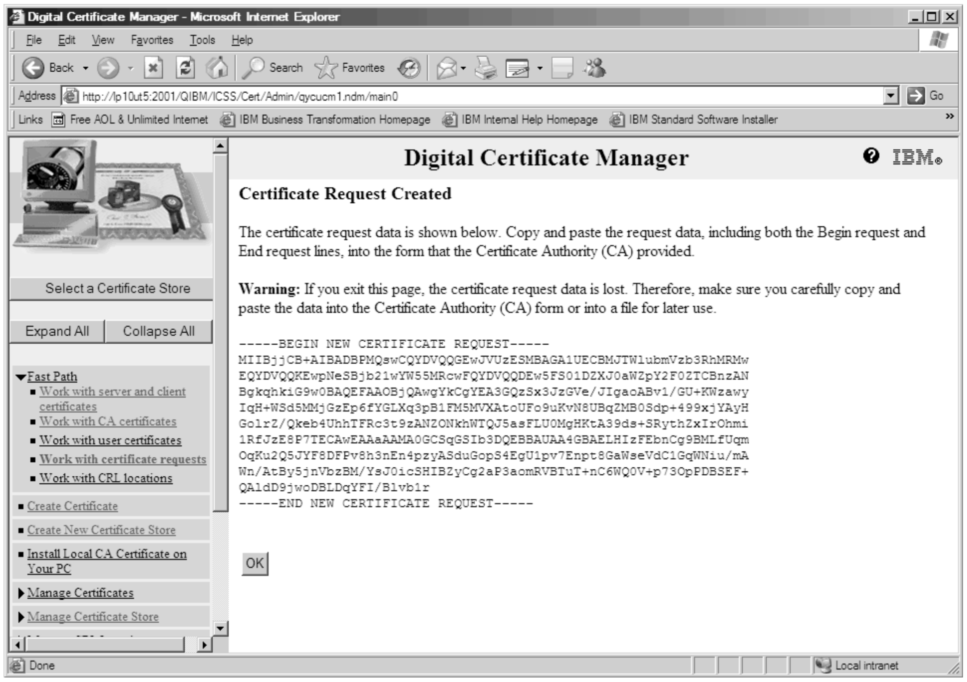
Common name: EKMCertificate (required)
Organization unit:
Organization name: My company (required)
Locality or city:
State or province: Minnesota (required:minimum of 3 characters)
Country or region: US (required)

Continue Cancel

Figure 42. Create Certificate window

click **Continue**.

6. The Certificate Request Created window opens (Figure 43).
Copy the contents of the certificate request and paste into the certificate request form provided by the CA. Once the signed certificate is received from the CA, copy it into a file on the System i system. Click **OK**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0209

Digital Certificate Manager

Certificate Request Created

The certificate request data is shown below. Copy and paste the request data, including both the Begin request and End request lines, into the form that the Certificate Authority (CA) provided.

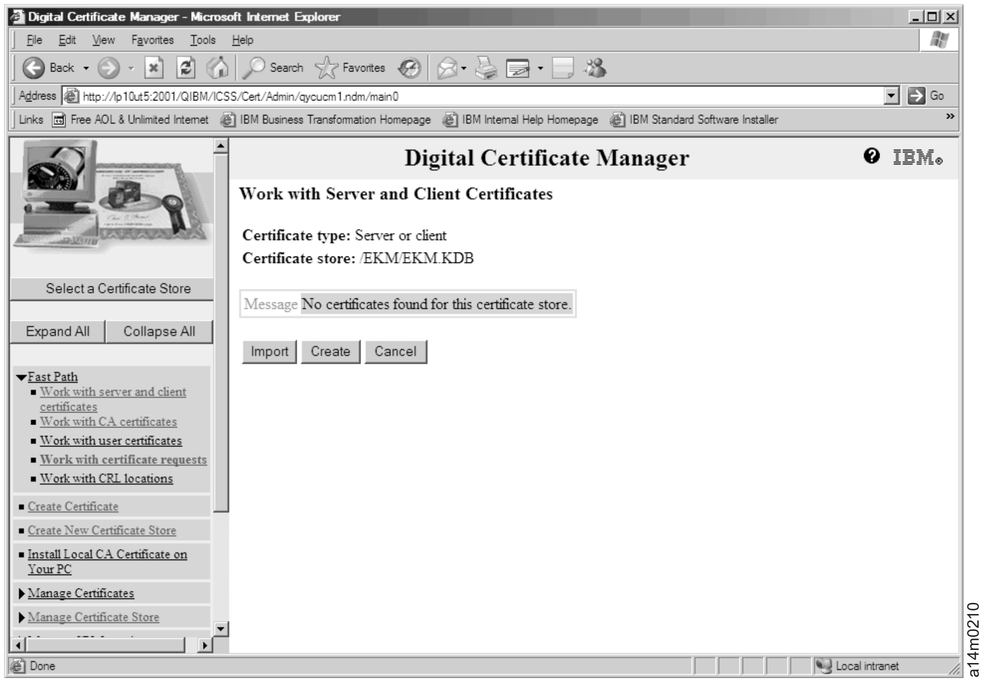
Warning: If you exit this page, the certificate request data is lost. Therefore, make sure you carefully copy and paste the data into the Certificate Authority (CA) form or into a file for later use.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBjjCB+AIBADBPMQswCQYDVQQGEwJVUzESMBAGA1UECBMJTWlubmVzb3RhMRMw
EQYDVQQKEwpNeSBjb21wYW55MRcwFQYDVQQDEw5FS01DZXJ0aWZpY2F0ZTCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA3GQzSx3JzGVe/JIgaoABv1/GU+KWzawy
IqH+WSd5MMjGzEp6fYGLXq3pB1FM5MVXAtoUFo9uKvN8UBqZMB0Sdp+499xjYAyH
GolrZ/Qkeb4UhhTFRc3t9zANZONkhWTQJ5asFLU0MgHKtA39ds+SRythZxIrOhmi
1RfJzE8P7TECAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBAELHIzFEbnCg9BMLfUqm
OqKu2Q5JYF8DFPv8h3nEn4pzyASduGopS4EgU1pv7Enpt8GaWseVdC1GqWNiu/mA
Wn/AtBy5jnVbzBM/YsJ0icSHIBZyCg2aP3aomRVBTuT+nC6WQ0V+p73OpPDBSEF+
QAldD9jwoDBLDqYFI/Blvb1r
-----END NEW CERTIFICATE REQUEST-----
```

OK

Figure 43. Certificate Request Created window

7. Select the Work with Server and Client Certificates option. The Work with Server and Client Certificates window opens (Figure 44).


Digital Certificate Manager - Microsoft Internet Explorer
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0210

Digital Certificate Manager

Work with Server and Client Certificates

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

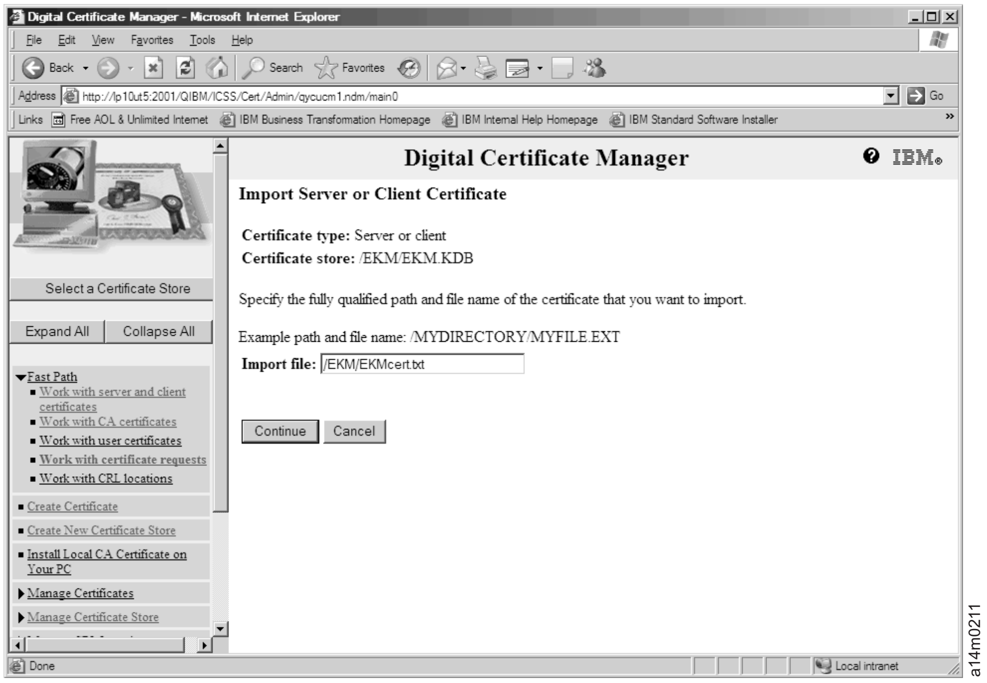
Message No certificates found for this certificate store.

Import Create Cancel

Figure 44. Work with Server and Client Certificates window

Click **Import** to receive the signed certificate.

8. The Import Server or Client Certificate window opens (Figure 45). Specify the name of the file into which you copied the signed certificate.


Digital Certificate Manager - Microsoft Internet Explorer
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0211

Digital Certificate Manager

Import Server or Client Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Specify the fully qualified path and file name of the certificate that you want to import.

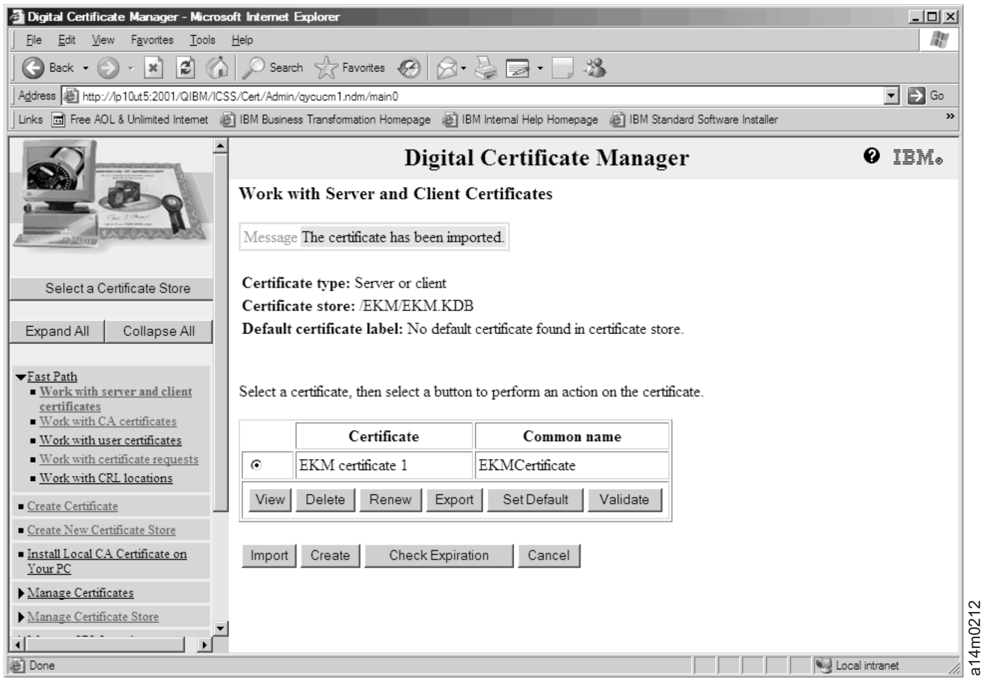
Example path and file name: /MYDIRECTORY/MYFILE.EXT
Import file: /EKM/EKMcert.txt

Continue Cancel

Figure 45. Import Server or Client Certificate window

Click **Continue**.

9. The Work with Server and Client Certificates window opens (Figure 46).


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Digital Certificate Manager
IBM
Done
Local intranet
a14m0212

Work with Server and Client Certificates

Message The certificate has been imported.

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Default certificate label: No default certificate found in certificate store.

Select a certificate, then select a button to perform an action on the certificate.

	Certificate	Common name
●	EKM certificate 1	EKMCertificate

View Delete Renew Export Set Default Validate

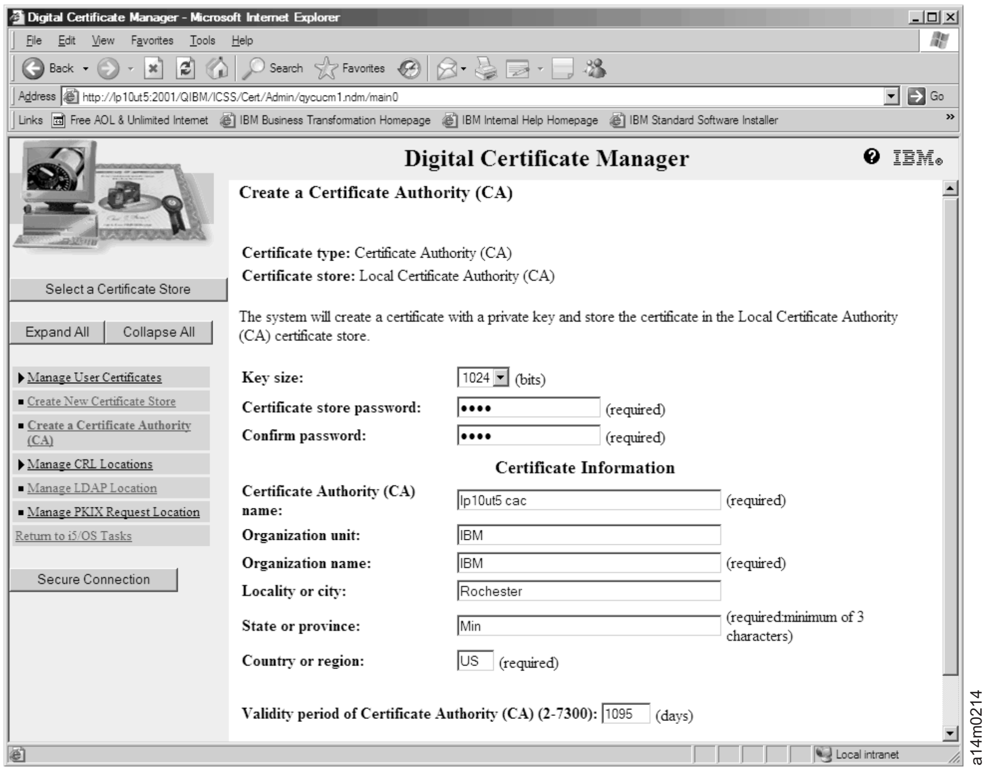
Import Create Check Expiration Cancel

Figure 46. Work with Server and Client Certificates

Create Self-Signed Certificate (for Internal Use)

Self-signed certificate key pairs can be used instead of CA-signed certificates for internal use only. Although DCM does not create self-signed certificates, it does create a *local CA*-signed certificate for internal use that serves the same purpose.

1. Click **Create a Local CA Certificate** menu item.
2. In the "Create Certificate Authority (CA)" window (Figure 47), specify a Key size of **2048** and the keystore password. Fill in the required certificate information.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM
Select a Certificate Store
Expand All
Collapse All
Manage User Certificates
Create New Certificate Store
Create a Certificate Authority (CA)
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Local intranet
a14m0214

Create a Certificate Authority (CA)

Certificate type: Certificate Authority (CA)
Certificate store: Local Certificate Authority (CA)

The system will create a certificate with a private key and store the certificate in the Local Certificate Authority (CA) certificate store.

Key size:	1024	(bits)
Certificate store password:	••••	(required)
Confirm password:	••••	(required)

Certificate Information

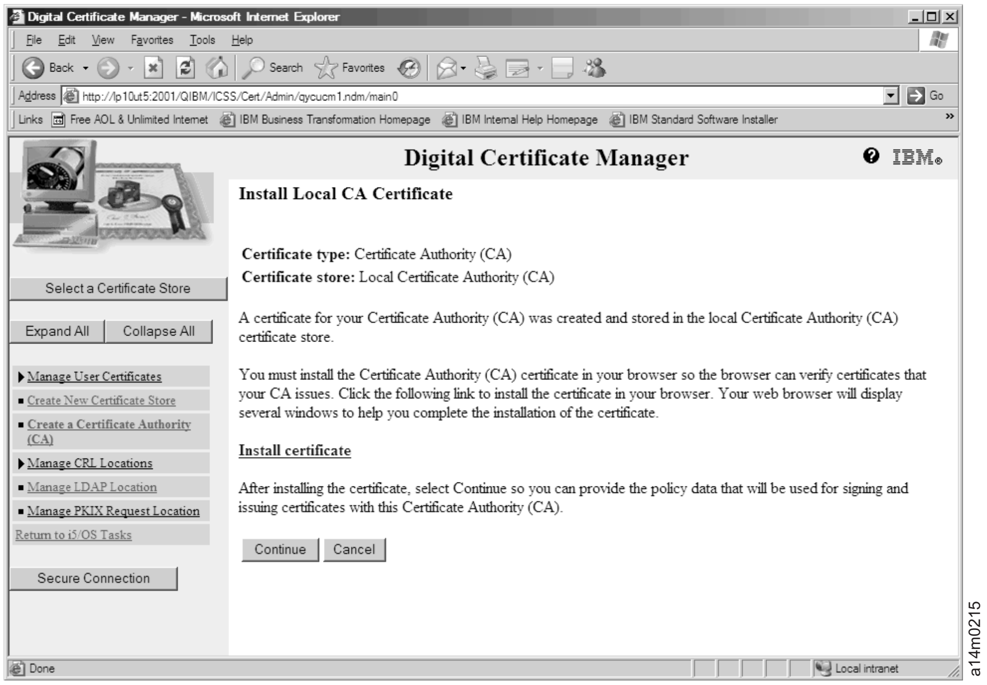
Certificate Authority (CA) name:	lp10ut5 cac	(required)
Organization unit:	IBM	
Organization name:	IBM	(required)
Locality or city:	Rochester	
State or province:	Min	(required:minimum of 3 characters)
Country or region:	US	(required)

Validity period of Certificate Authority (CA) (2-7300): 1095 (days)

Figure 47. Create a Local CA Certificate

Click **Continue**.

3. In the “Install Local CA Certificate” window (Figure 48), click **Install certificate** to install on your browser.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM
Select a Certificate Store
Expand All
Collapse All
Manage User Certificates
Create New Certificate Store
Create a Certificate Authority (CA)
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Install certificate
Continue
Cancel
Done
Local intranet
a14m0215

Install Local CA Certificate

Certificate type: Certificate Authority (CA)
Certificate store: Local Certificate Authority (CA)

A certificate for your Certificate Authority (CA) was created and stored in the local Certificate Authority (CA) certificate store.

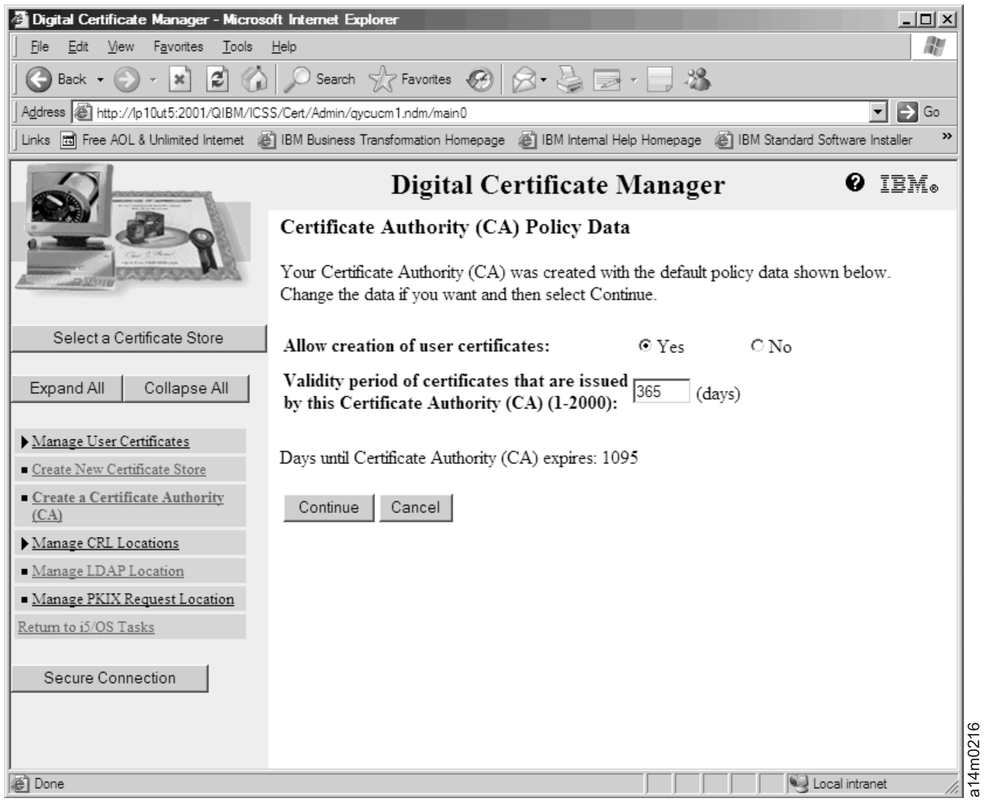
You must install the Certificate Authority (CA) certificate in your browser so the browser can verify certificates that your CA issues. Click the following link to install the certificate in your browser. Your web browser will display several windows to help you complete the installation of the certificate.

After installing the certificate, select Continue so you can provide the policy data that will be used for signing and issuing certificates with this Certificate Authority (CA).

Figure 48. Install Local CA Certificate window

Click **Continue**.

4. The “Certificate Authority (CA) Policy Data” window opens (Figure 49). Click the **Yes** button to allow creation of user certificates and specify period of issue.
   Click **Continue**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM
Select a Certificate Store
Expand All
Collapse All
Manage User Certificates
Create New Certificate Store
Create a Certificate Authority (CA)
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Continue
Cancel
Done
Local intranet
a14m0216

Certificate Authority (CA) Policy Data

Your Certificate Authority (CA) was created with the default policy data shown below. Change the data if you want and then select Continue.

Allow creation of user certificates: [✓ Yes] No

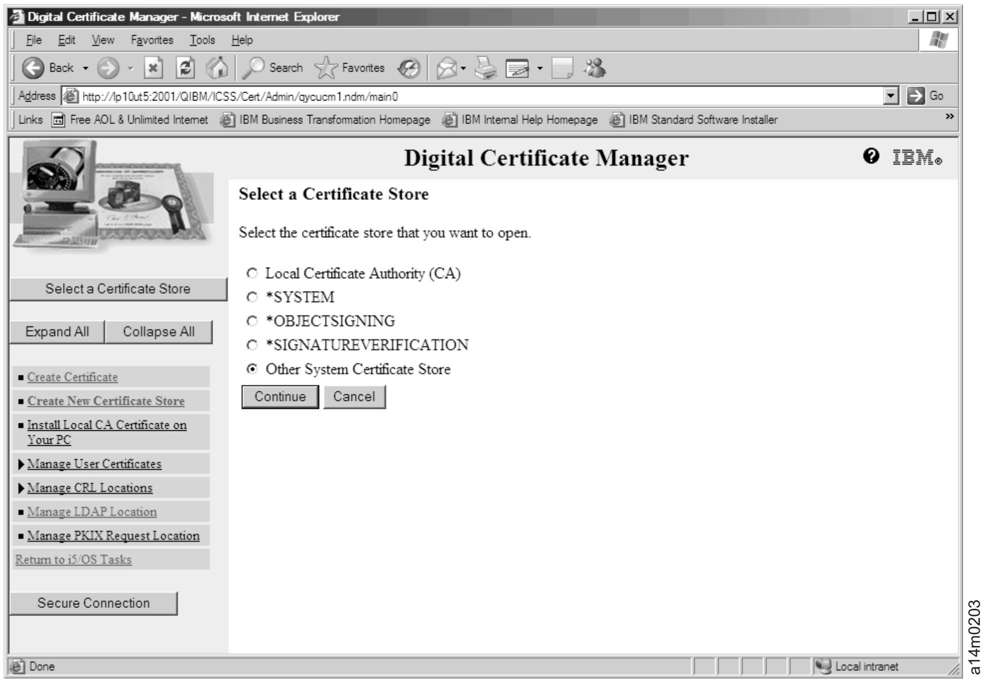
Validity period of certificates that are issued by this Certificate Authority (CA) (1-2000): 365 (days)

Days until Certificate Authority (CA) expires: 1095

Figure 49. Certificate Authority (CA) Policy Data window

5. Select the **Select Certificate Store** menu item.

6. In the “Select Certificate Store” window (Figure 50), Click the button for **Other System Certificate Store**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Digital Certificate Manager
IBM
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Done
Local intranet
a14m0203

Select a Certificate Store

Select the certificate store that you want to open.

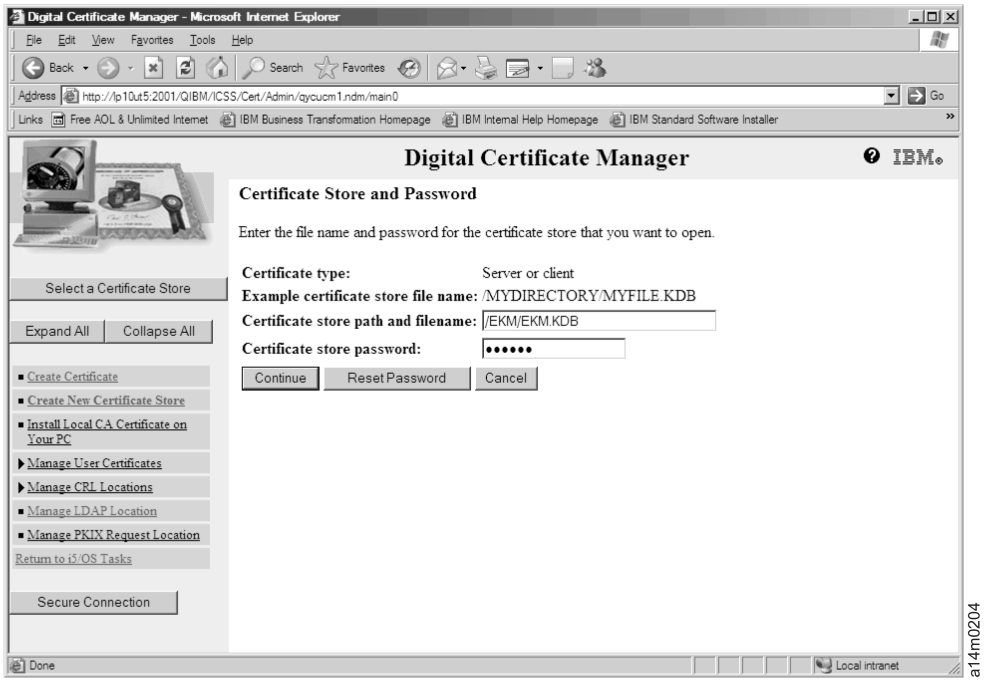
- Local Certificate Authority (CA)
- *SYSTEM
- *OBJECTSIGNING
- *SIGNATUREVERIFICATION
- [✓ Other System Certificate Store]

Continue Cancel

Figure 50. Select Certificate Store window

Click **Continue**.

7. The “Certificate Store and Password” window opens (Figure 40 on page 109). Specify the path and filename you entered in “Create Keyring/Keystore Instance” on page 105.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Digital Certificate Manager
Select a Certificate Store
Expand All
Collapse All
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage User Certificates
Manage CRL Locations
Manage LDAP Location
Manage PKIX Request Location
Return to i5/OS Tasks
Secure Connection
Continue
Reset Password
Cancel
Done
Local intranet
a14m0204

Certificate Store and Password

Enter the file name and password for the certificate store that you want to open.

Certificate type:	Server or client
Example certificate store file name:	/MYDIRECTORY/MYFILE.KDB
Certificate store path and filename:	/EKM/EKM.KDB
Certificate store password:	••••••

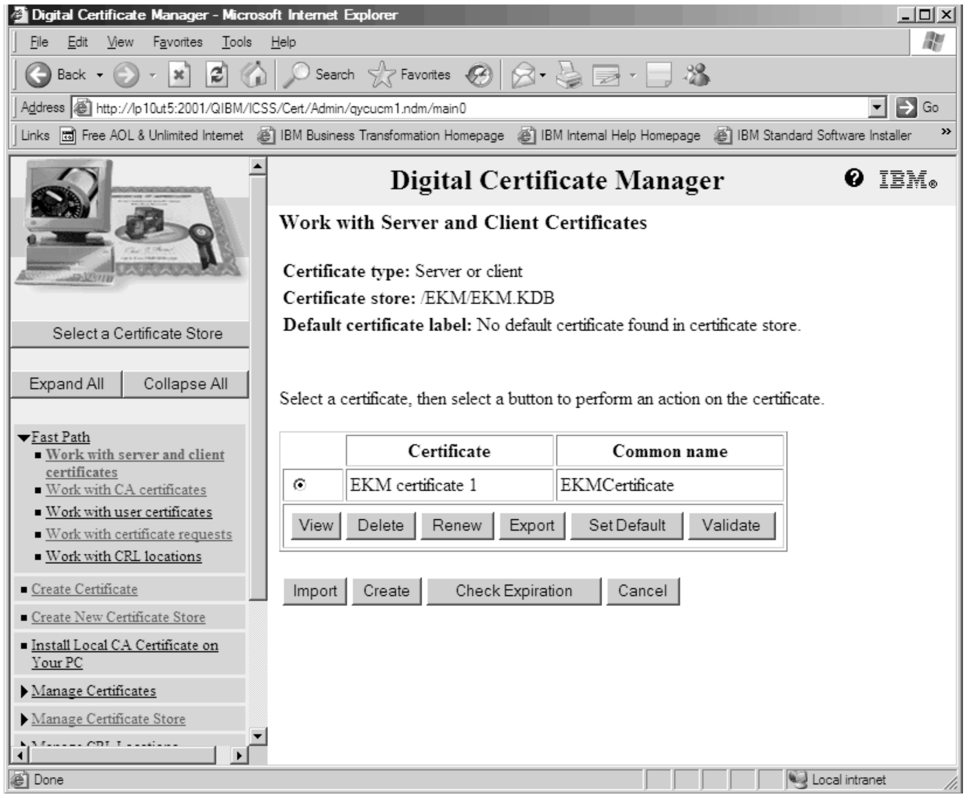
Figure 51. Certificate Store and Password window

Click **Continue**.

The “Current Certificate Store” window opens, verifying your certificate store selection.

After you select a certificate store, you can use the **Work with Server and Client Certificates** option of the **Fast Path** menu group to perform all of the tasks, or use the various **Manage Certificates** menu group options to perform individual tasks.

8. The “Work with Server and Client Certificates” window opens (Figure 52).


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0217

Digital Certificate Manager

Work with Server and Client Certificates

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Default certificate label: No default certificate found in certificate store.

Select a certificate, then select a button to perform an action on the certificate.

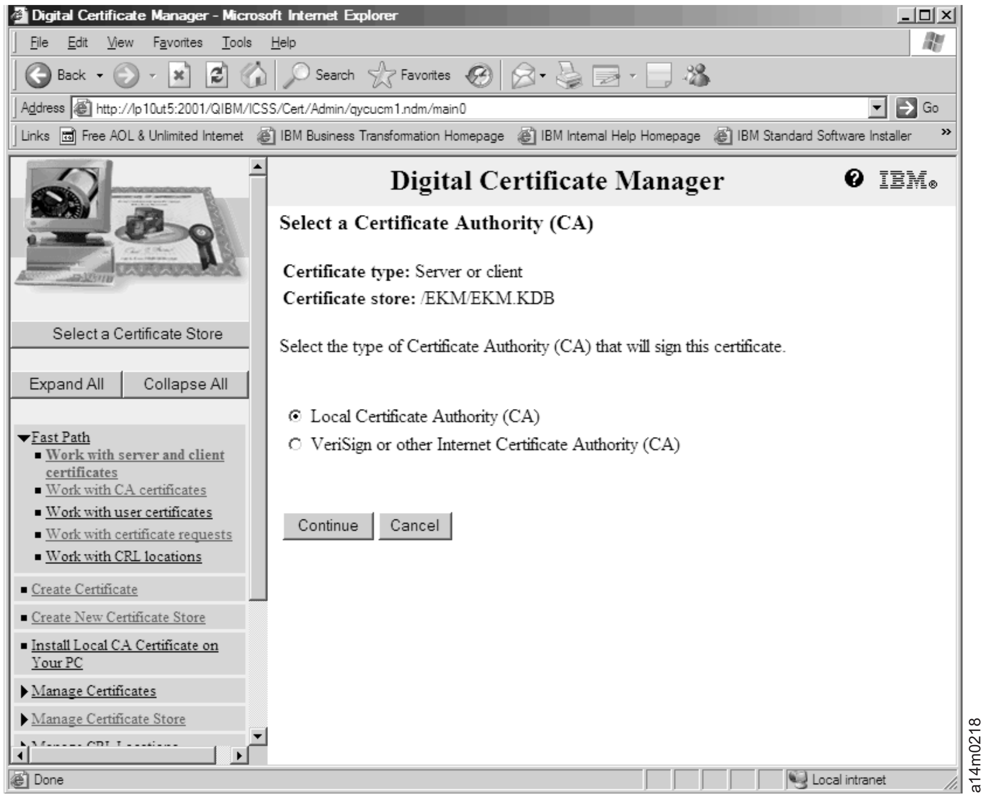
	Certificate	Common name
●	EKM certificate 1	EKMCertificate

View Delete Renew Export Set Default Validate
Import Create Check Expiration Cancel

Figure 52. Work with Server and Client Certificates window

Click **Create**.

9. The “Select a Certificate of Authority (CA)” window opens (Figure 53). Select the button for Local Certificate Authority (CA).


Digital Certificate Manager - Microsoft Internet Explorer

Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0

Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer

Digital Certificate Manager

Select a Certificate Authority (CA)

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Select the type of Certificate Authority (CA) that will sign this certificate.

[✓ Local Certificate Authority (CA)]
VeriSign or other Internet Certificate Authority (CA)

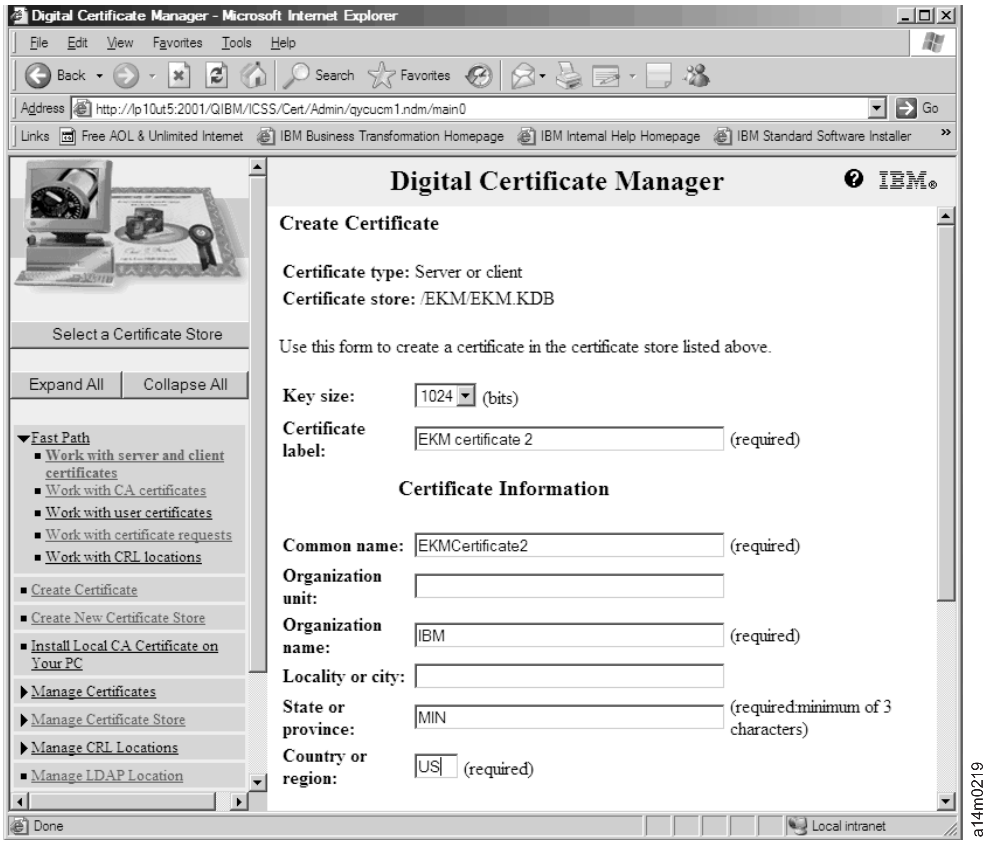
Continue Cancel

File Edit View Favorites Tools Help
Back Search Favorites
Select a Certificate Store
Expand All Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0218

Figure 53. Select a Certificate of Authority (CA) window

Select **Continue** .

10. In the “Create Certificate” window (Figure 54), specify a Key size of **2048** and a Certificate label value that will correspond to the alias2 value in your Encryption Key Manager configuration file. Fill in the other required values as appropriate.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Manage CRL Locations
Manage LDAP Location
Done
Local intranet
Digital Certificate Manager
IBM
a14m0219

Create Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Use this form to create a certificate in the certificate store listed above.

Key size:	1024	(bits)
Certificate label:	EKM certificate 2	(required)

Certificate Information

Common name:	EKMCertificate2	(required)
Organization unit:		
Organization name:	IBM	(required)
Locality or city:		
State or province:	MIN	(required:minimum of 3 characters)
Country or region:	US	(required)

Figure 54. Create Certificate window

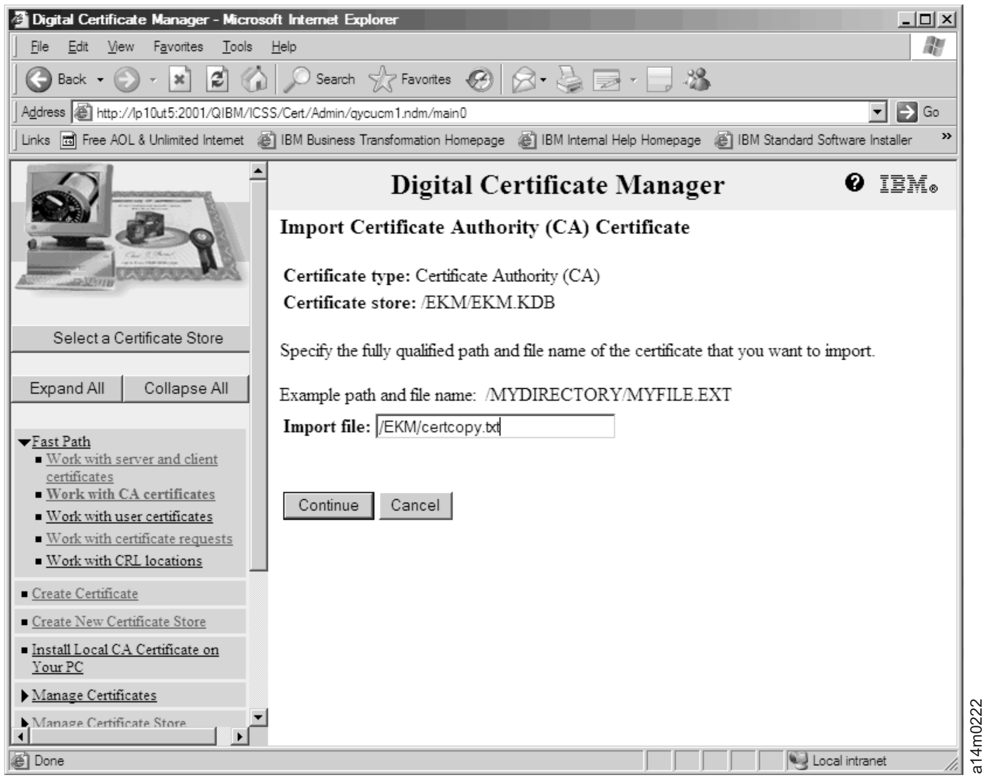
Click **Continue**.

The “Work with Server and Client Certificates” window opens showing the newly created certificate in the list.

Receive Certificate

Perform these steps to create a second key label (alias2) to be used for EEDK generation when receiving a certificate from an outside organization. Since this certificate will not have a private key it must be imported as a CA certificate through DCM.

1. Select the **Work with CA certificates** menu item and click **Import** when the "Work with CA certificates" window opens.
2. In the "Import Certificate Authority (CA) Certificate" window (Figure 55), specify the certificate to be imported.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0222

Digital Certificate Manager

Import Certificate Authority (CA) Certificate

Certificate type: Certificate Authority (CA)
Certificate store: /EKM/EKM.KDB

Specify the fully qualified path and file name of the certificate that you want to import.

Example path and file name: /MYDIRECTORY/MYFILE.EXT

Import file: /EKM/certcopy.txt

Continue Cancel

Figure 55. Import Certificate Authority window

Click **Continue**.

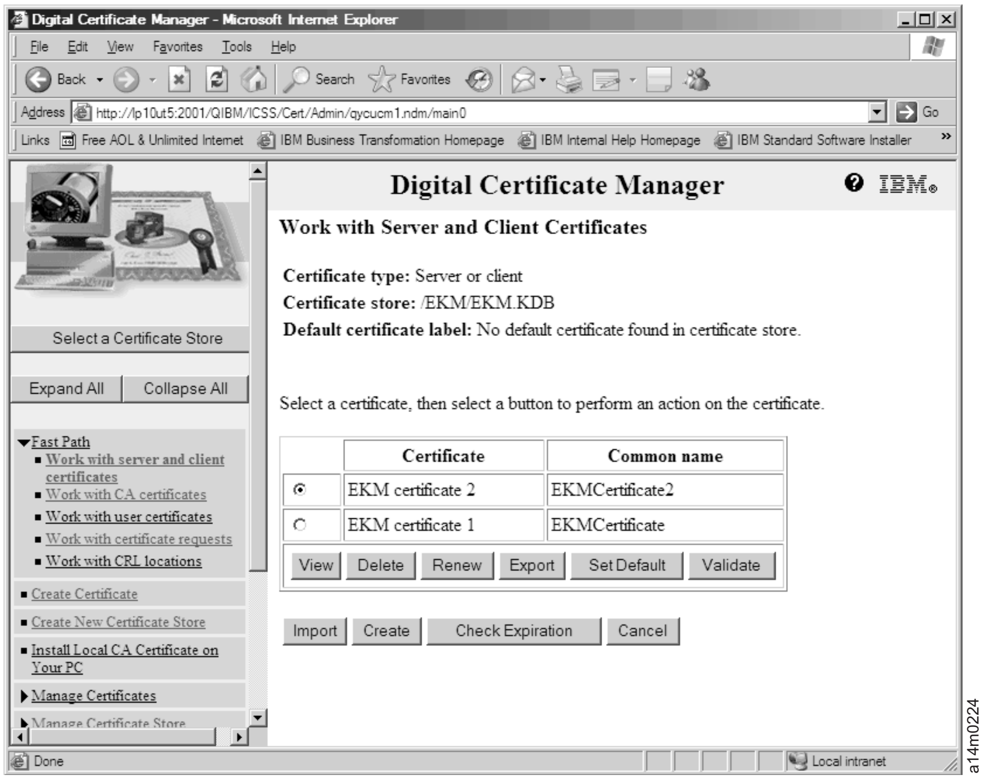
3. Enter password when prompted. Click **Continue**.

   The "Work with CA Certificates" window reopens, showing the imported certificate.

Export Private Key and Certificate

Perform these steps when copying or moving a key, a key and certificate, or only a certificate, from a source keystore.

1. Select the **Work with Server and Client Certificates** menu item.
2. On the “Work with Server and Client Certificates” window, (Figure 56). select the desired certificate and Click **Export**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0224

Digital Certificate Manager

Work with Server and Client Certificates

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Default certificate label: No default certificate found in certificate store.

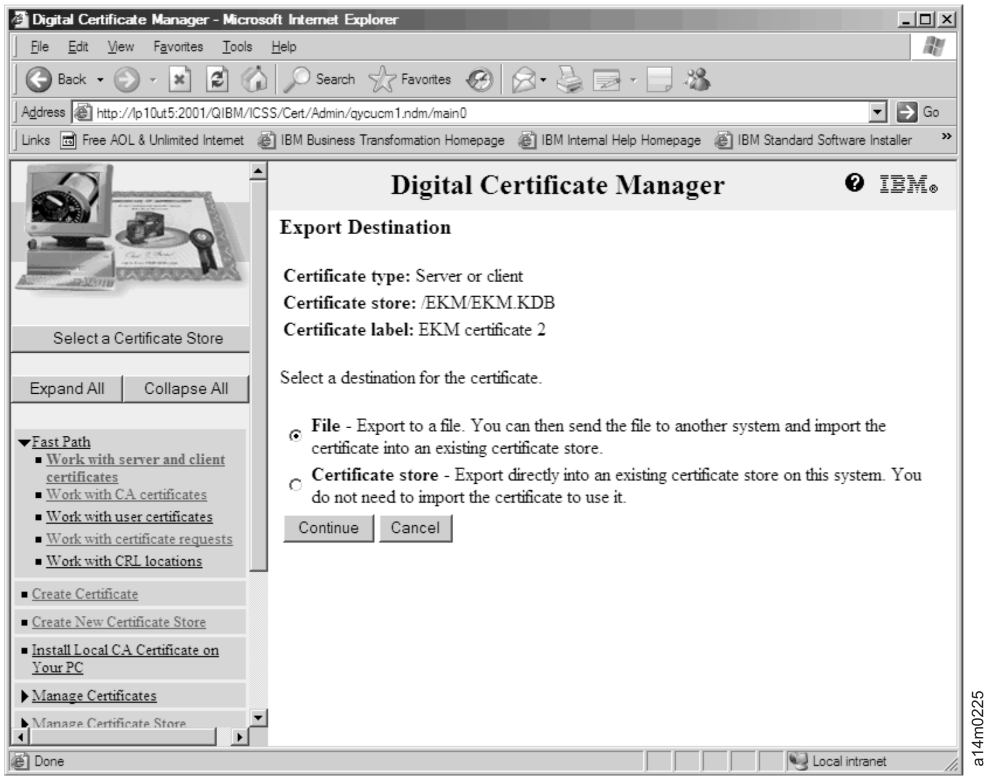
Select a certificate, then select a button to perform an action on the certificate.

	Certificate	Common name
●	EKM certificate 2	EKMCertificate2
○	EKM certificate 1	EKMCertificate

View Delete Renew Export Set Default Validate
Import Create Check Expiration Cancel

Figure 56. Work with Server and Client Certificates window

3. In the “Export Destination” window (Figure 57), Click the button to export the certificate to a file.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0 Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0225

Digital Certificate Manager

IBM

Export Destination

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Certificate label: EKM certificate 2

Select a destination for the certificate.

[✓] **File** - Export to a file. You can then send the file to another system and import the certificate into an existing certificate store.

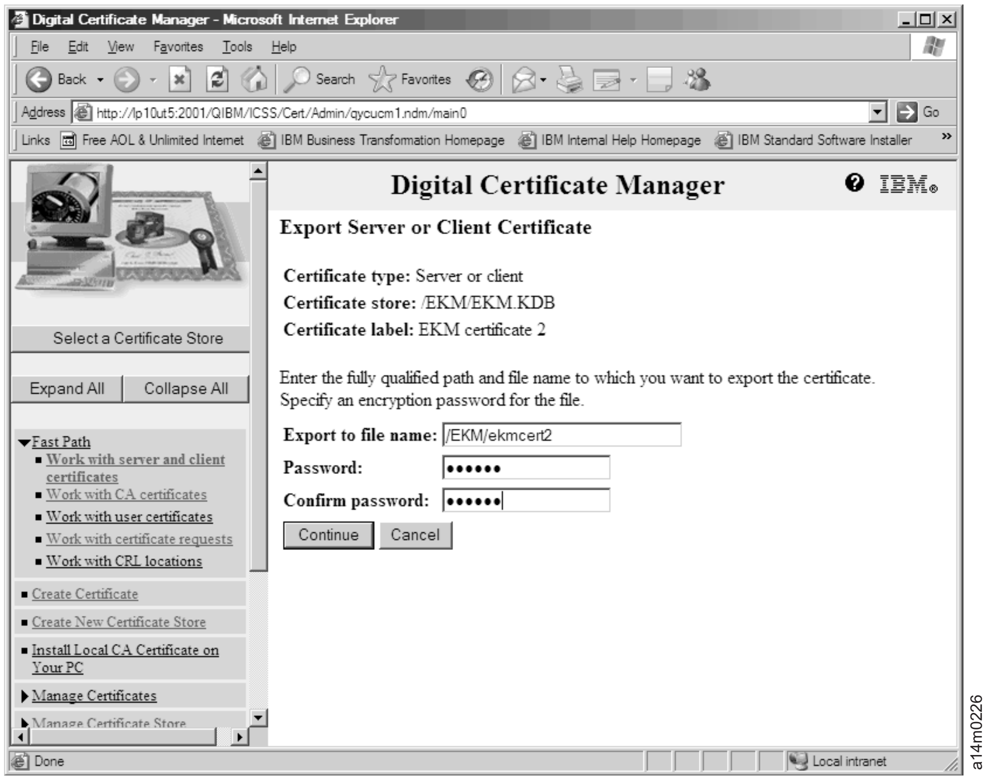
Certificate store - Export directly into an existing certificate store on this system. You do not need to import the certificate to use it.

Continue Cancel

Figure 57. Export Destination window

Click **Continue**.

4. In the “Export Server or Client Certificate” window (Figure 58), specify a file name and enter the password.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0226

Digital Certificate Manager

Export Server or Client Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Certificate label: EKM certificate 2

Enter the fully qualified path and file name to which you want to export the certificate. Specify an encryption password for the file.

Export to file name: /EKM/ekmcert2
Password: ••••••
Confirm password: ••••••

Continue Cancel

Figure 58. Export Server or Client Certificate window

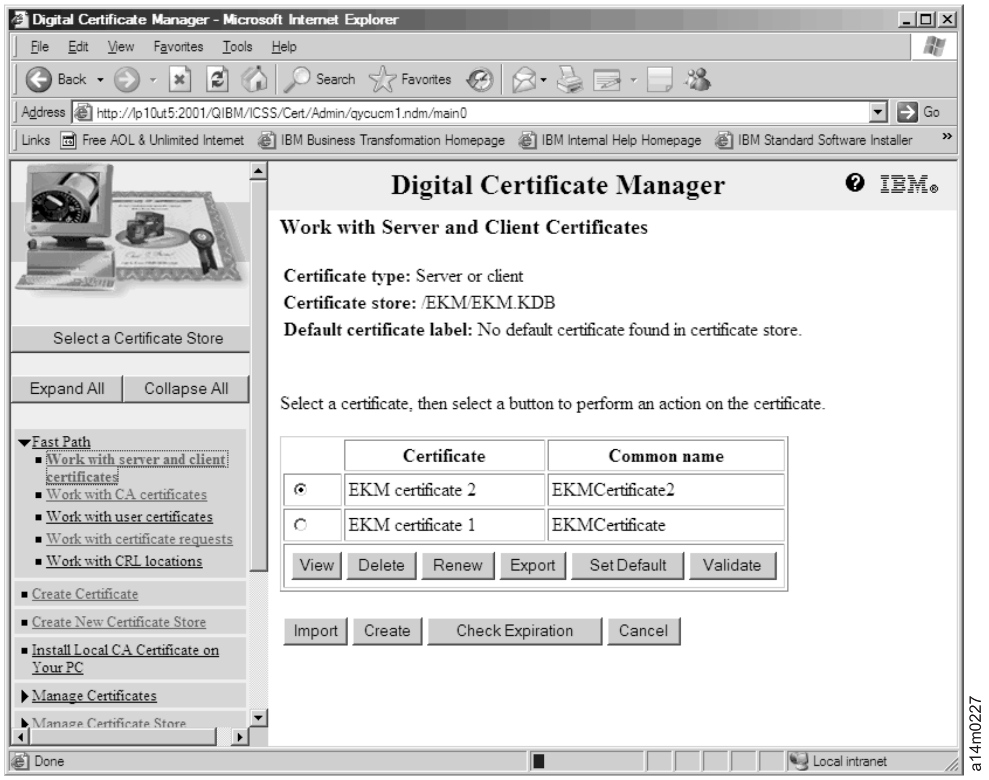
Click **Continue**.

The Certificate Exported window opens.

Import Private Key and Certificate

Perform these steps when copying or moving a key, a key and certificate, or only a certificate into a target keystore.

1. Click the **Work with Server and Client Certificates** menu item.
2. In the "Work with Server and Client Certificates" window, (Figure 59, select the desired certificate and Click **Import**.


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Digital Certificate Manager
IBM
View Delete Renew Export Set Default Validate
Import Create Check Expiration Cancel
Done
Local intranet
a14m0227

Work with Server and Client Certificates

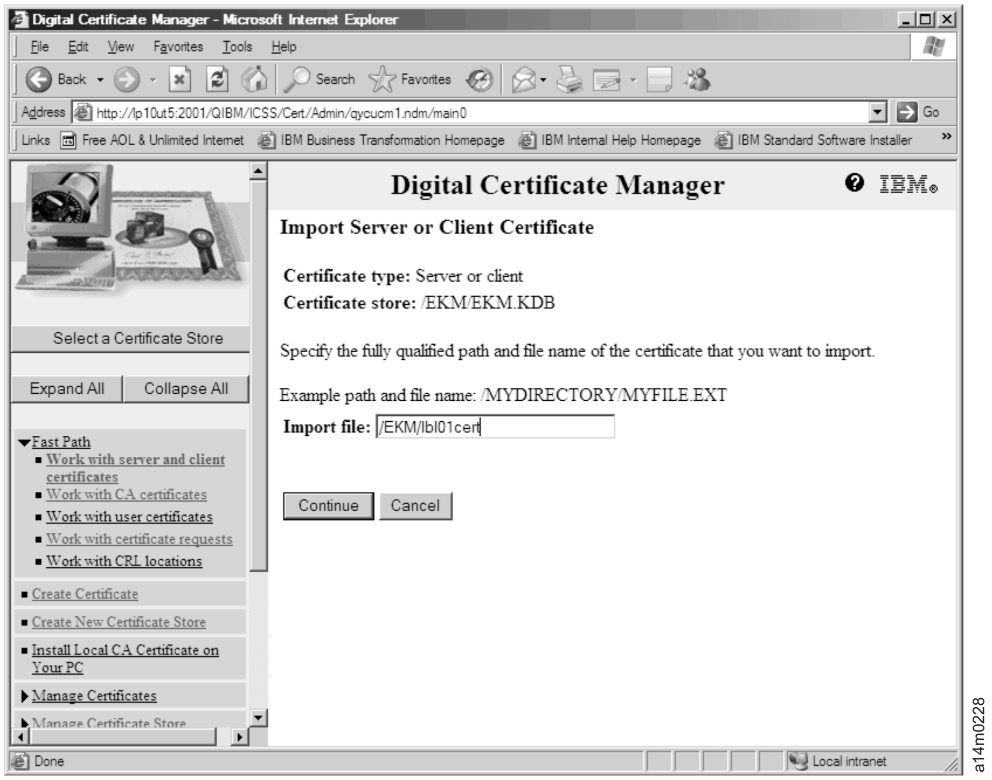
Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Default certificate label: No default certificate found in certificate store.

Select a certificate, then select a button to perform an action on the certificate.

	Certificate	Common name
●	EKM certificate 2	EKMCertificate2
○	EKM certificate 1	EKMCertificate

Figure 59. Work with Server and Client Certificates window

3. The "Import Server or Client Certificate" window opens (Figure 60). Specify the name of the certificate file to be imported.
   Click **Continue**.



Digital Certificate Manager

Import Server or Client Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Specify the fully qualified path and file name of the certificate that you want to import.

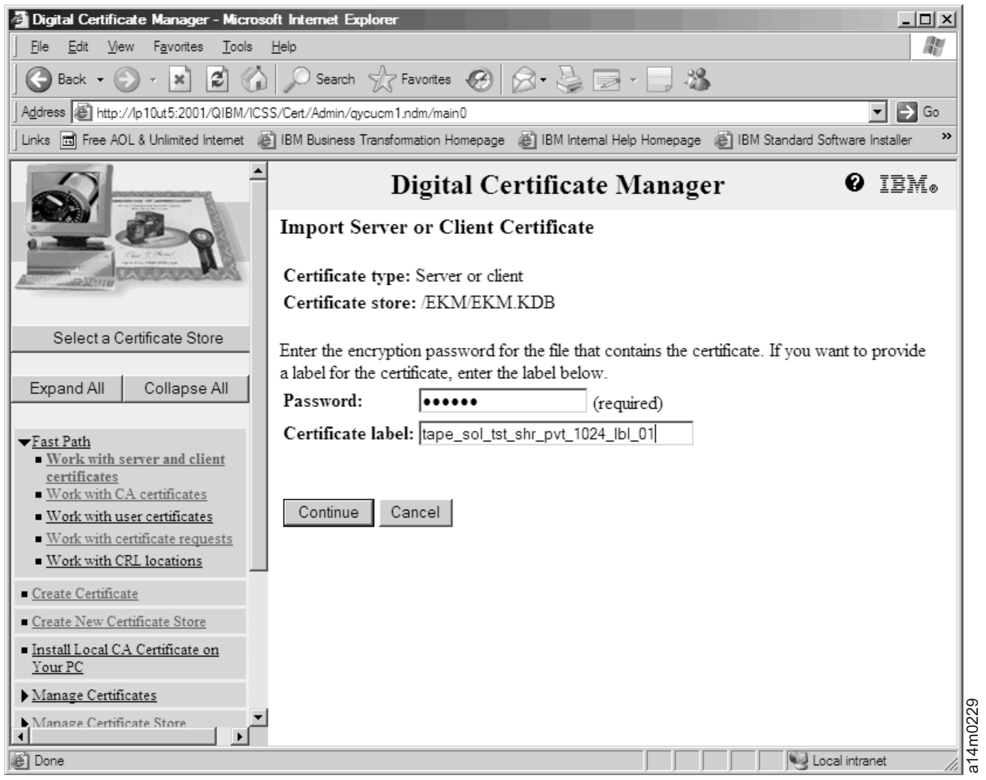
Example path and file name: /MYDIRECTORY/MYFILE.EXT

Import file: /EKM/lbl01cert

Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
IBM
Continue
Cancel
Done
Local intranet
a14m0228

Figure 60. Import Server or Client Certificate window

4. Specify the password when prompted (Figure 61 on page 128).



Digital Certificate Manager - Microsoft Internet Explorer

http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0

Digital Certificate Manager

Import Server or Client Certificate

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB

Enter the encryption password for the file that contains the certificate. If you want to provide a label for the certificate, enter the label below.

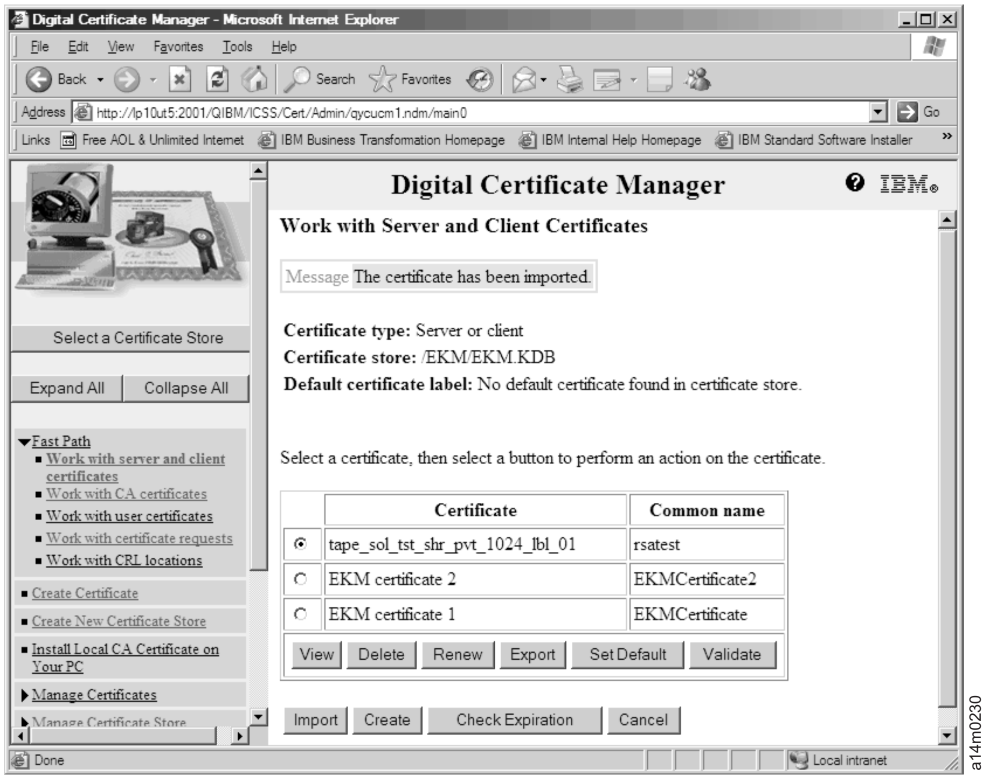
Password: •••••• (required)
Certificate label: tape_sol_tst_shr_pvt_1024_lbl_01

Continue
Cancel
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Done
Local intranet
a14m0229

Figure 61. Import Certificate Password window

Click **Continue**.

5. The "Work with Server and Client Certificates" window reopens, showing the imported certificate (Figure 62).


Digital Certificate Manager - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://lp10ut5:2001/QIBM/ICSS/Cert/Admin/qycucm1.ndm/main0
Go
Links Free AOL & Unlimited Internet IBM Business Transformation Homepage IBM Internal Help Homepage IBM Standard Software Installer
Select a Certificate Store
Expand All
Collapse All
Fast Path
Work with server and client certificates
Work with CA certificates
Work with user certificates
Work with certificate requests
Work with CRL locations
Create Certificate
Create New Certificate Store
Install Local CA Certificate on Your PC
Manage Certificates
Manage Certificate Store
Digital Certificate Manager
IBM
Done
Local intranet
a14m0230

Work with Server and Client Certificates

Message The certificate has been imported.

Certificate type: Server or client
Certificate store: /EKM/EKM.KDB
Default certificate label: No default certificate found in certificate store.

Select a certificate, then select a button to perform an action on the certificate.

	Certificate	Common name
●	tape_sol_tst_shr_pvt_1024_lbl_01	rsatest
○	EKM certificate 2	EKMCertificate2
○	EKM certificate 1	EKMCertificate

View Delete Renew Export Set Default Validate
Import Create Check Expiration Cancel

Figure 62. Work with Server and Client Certificates window showing imported certificate

Chapter 4. Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK

You should always install the latest Encryption Key Manager regardless of the level packaged with your SDK. See http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 for latest available version. To check the version you currently have installed, follow one of the procedures in "Determining Which Version of Encryption Key Manager is Installed."

Determining Which Version of Encryption Key Manager is Installed

How to check the Encryption Key Manager JAR version if you have the Encryption Key Manager installed and running

If you already have the Encryption Key Manager running you can execute the **version** command to find out the level of Encryption Key Manager. (See "CLI Commands" on page 161.)

How to check the Encryption Key Manager JAR version if you do not yet have the Encryption Key Manager running

Each Java JAR file contains a manifest file which has the version information for the jar. In order to view the manifest file, you will first need to extract it by following these instructions:

1. From any command window or shell, verify that java is in your PATH by running the **java -version** command. For more information, see the instructions in "Verify the Java Version" on page 46.
2. Go to your java installations lib/ext directory:

   ```
   cd JAVA_HOME/lib/ext
   ```

   If you list the contents of the directory you should see the IBMKeyManagementServer.jar file there.
3. List the contents of the IBMKeyManagementServer.jar file. This will produce a large listing but you should see a file called MANIFEST.MF

   ```
   jar -tvf IBMKeyManagementServer.jar
   234 Wed Aug 16 10:48:04 EDT 2006 META-INF/MANIFEST.MF
   ```
4. Extract the Manifest file:

   ```
   jar -xvf IBMKeyManagementServer.jar META-INF/MANIFEST.MF
   extracted: META-INF/MANIFEST.MF
   ```
5. If on z/OS only, you will need to convert the extracted manifest file so that it can be read:

   ```
   iconv -f iso8859-1 -t ibm-1047 META-INF/MANIFEST.MF > converted.MANIFEST.MF
   ```
6. Read the manifest file and find the Build Level, which reflects the Encryption Key Manager version (in this example, it is the minimum of 20070914 for Encryption Key Manager 2.1). The manifest file can be displayed by using the **cat** command. Compare the Build Level to the latest one displayed at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

   ```
   cat META-INF/MANIFEST.MF
   For z/OS cat converted.MANIFEST.MF

   Manifest-Version: 1.0
   Ant-Version: Apache Ant 1.5.3
   Created-By: 1.4.2 (IBM Corporation)
   ```

```
Implementation-Version: 1.5
Implementation-Title: Key Management Server
Implementation-Vendor: IBM Corporation
Build-Level: -20070914
```

Downloading the Latest Version Encryption Key Manager

You can acquire a new version of Encryption Key Manager in two ways:

- By downloading the latest IBMKeyManagementServer JAR file.
- By installing the latest IBM Java SDK which includes the IBMKeyManagementServer.jar file. Please note that installing the latest IBM Java SDK **does not ensure** you will get the latest available IBMKeyManagementServer.jar file. See Table 21.

Both are available for download at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504.

Downloading the latest IBMKeyManagementServer JAR file:
When downloading the latest IBMKeyManagementServer.jar from the website, look for "EKM Downloads" and click the link for **IBM Storage Encryption ftp site**. Download the IBMKeyManagementServer.jar file from the directory called **latest**. Use this new version to replace the version currently located in the java_home/lib/ext directory of your SDK 14.2, SDK5.0 or SDK 6.0 JVM.

When upgrading, always check these upgrade notes in the latest edition of this publication to ensure that a newer version of Encryption Key Manager does not require any additional configuration settings due to new features or functions.

Upgrade note: When upgrading from an Encryption Key Manager build earlier than 20070503 you must define the Audit.metadata.file.name property in the KeyManagerConfig.properties file. This is the name of the XML file where metadata is saved. This property is required to start versions of Encryption Key Manager with build date 20070503 (when metadata support was added) and later. See Chapter 9, "Using Metadata," on page 199 for more information.

Installing the latest IBM Java SDK:
Because the IBMKeymanagementServer.jar file is shipped with the IBM Java SDK, you may choose to upgrade by downloading the latest IBM Java SDK available. Please note, however, that installing the latest IBM Java SDK **does not ensure** you will get the latest IBMKeyManagementServer.jar file available. See Table 21 to learn which version of Encryption Key Manager was packaged with your installed version of IBM Java SDK. If your version of IBM Java SDK is not listed, refer to http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 for the latest information. You should always install the latest Encryption Key Manager regardless of the level packaged with your SDK.

Table 21. Encryption Key Manager versions shipped with Java SDKs

IBM Java SDK (availability date)	Included this Encryption Key Manager version build
SDK 1.4.2 SR6 (8/24/2006)	8/16/2006 - Pre-GA version - unsupported, must be downloaded from website.
SDK 1.4.2 SR7 (11/24/2006)	8/16/2006 - Pre-GA version - unsupported, must be downloaded from website.
SDK 1.4.2 SR8 (3/29/2007)	1.0 - 20061024

Table 21. Encryption Key Manager versions shipped with Java SDKs (continued)

IBM Java SDK (availability date)	Included this Encryption Key Manager version build
SDK 1.4.2 SR9 (7/8/2007)	2.0 - 20070503
SDK 1.4.2 SR9a (8/13/2007)	2.0 - 20070503
SDK 1.4.2 SR10 (1/22/2008)	2.1 - 20070924
SDK 1.4.2 SR11 (5/17/2008)	2.1 - 20080325
SDK 5.0 SR3 (10/2/2006)	8/16/2006 - Pre-GA version - unsupported.
SDK 5.0 SR4 (2/1/2007)	1.0 - 20061024
SDK 5.0 SR5 (5/11/2007)	1.0 - 20061024
SDK 5.0 SR5a (8/1/2007)	1.0 - 20061024
SDK 5.0 SR6b (10/25/2007)	Pre-GA version of 2.1 (8/31/2007) unsupported, must be downloaded from website.
SDK 5.0 SR7 (3/15/2008)	Pre-release service for 2.1 (3/6/2008), unsupported, must be downloaded from website.
SDK 5.0 SR8 (7/2008)	2-1 20080530
SDK 6.0 (11/23/2007)	2.1 - 20070924
SDK 6.0 SR1 (4/16/2008)	2.1 - 20080325

Chapter 5. Configuring the Encryption Key Manager

Configuration Strategies

Some configuration settings in the KeyManagerConfig.properties file provide shortcuts that may have effects you should know about.

Automatically Update Tape Drive Table

The Encryption Key Manager provides a variable in the configuration file (drive.acceptUnknownDrives) that, when set to a value of `true`, automatically populates the tape drive table when a new tape drive contacts the Encryption Key Manager. This eliminates the need to use the **adddrive** command for each tape drive or Library. In this mode, the 12-digit serial number for each of these devices need not be entered using the CLI client commands. The new drives undergo the normal public/private key cryptography exchange to verify the identity of the tape device. Once this verification is complete, the new device is able to read existing tapes based on the EEDKs or Key IDs stored on them (assuming the corresponding key information is found in the configured keystore).

Note: The Encryption Key Manager server should be refreshed using the "refresh" on page 166 command after drives are added automatically to ensure that they are stored in the drive table.

For TS1120 or TS1130 drives, this capability allows you to set the default certificate alias (or key label) and alternate certificate alias (drive.default.alias1, drive.default.alias2) for encryption on newly added devices. For LTO 4 drives, you can set the default symmetric key pool (symmetricKeySet) for encryption on newly added devices. In other words, you can have the Encryption Key Manager fully configure the device with associated key material when the device makes contact. If you choose not to do this when the device is added to the drive table, you can do so after the tape drive has been added to the tape drive table, using the **moddrive** command.

In addition to relieving the administrator of the need to enter the 12-digit serial number for each of the tape drives the Encryption Key Manager will service, it also allows a default environment for large systems configurations.

It should be noted that such convenience comes at the price of slightly reduced security. Since the devices are added automatically and could be associated with a certificate alias (able to write a tape with that certificate alias), the added security check that the administrator would perform when adding the devices manually is skipped. It is important that you evaluate the advantages and disadvantages of this option to determine if automatically adding the tape drive information to the drive table, and implicitly granting that new device access to the certificate information, is an acceptable security risk.

Note: The drive.acceptUnknownDrives property is set to `false` by default. Thus, the Encryption Key Manager will not add new drives to the drive table automatically. Choose the mode you wish to operate in and change the configuration accordingly. See Appendix B for details.

Global Default Alias (Key Label) for TS1120 and TS1130 Tape Drive Writes

An option is available to set an Encryption Key Manager-wide default for the primary alias and secondary aliases. These are used to wrap the data encrypting key when writing to a TS1120 or TS1130 Tape Drive that is not supplied with key labels. The values in these two variables are used when a tape device in the drive table does not have a specific definition as to what alias to use when writing a tape. When the Encryption Key Manager encounters a device that does not have a specific entry in the drive table specifying an alias (or key label) and alternate alias, the Encryption Key Manager would then use the drive.default.alias1 and drive.default.alias2 variables if they are set. For more information on these configuration variables please refer to Appendix B.

Notes:

When the tape drive is not supplied with key labels, and the acceptUnknownDrives setting is true, and a previously unknown tape drive communicates with the Encryption Key Manager, the global default alias(es) must be set in the KeyManagerConfig.properties file. Otherwise, the newly accepted drive will not be able to write tapes. (that is, drive.default.alias1 and 2 must be defined with alias/key labels for this to work correctly).

On the TS3500, the default alias (either defined in the properties file or the drive alias) is used to verify the Encryption Key Manager operation by means of a three-part test. If at least one set of aliases is not defined, the third part of the test fails, even though the Encryption Key Manager is otherwise functional. See *IBM System Storage TS3500 Tape Library Operator Guide* for instructions on performing the test.

Synchronizing Data Between Two Key Manager Servers

The drive table and configuration properties file can be synchronized between two Encryption Key Manager servers. This can be done manually by using the CLI client **sync** command or automatically by setting four properties in the KeyManagerConfig.properties file.

Notes

Neither synchronization method acts on the keystore or key groups XML file. They must be copied manually.

If your environment uses Shared HFS function for Unix Systems Services AND you use shared directories for debug and error logs, use of the **sync -all** or **sync -config** command is not recommended as this will change the log settings on synchronized systems to use the same directories. Only the **sync -drivetable** command should be used for this type of configuration.

The automatic synchronization function is only enabled when a valid IP address is specified in the sync.ipaddress property of the KeyManagerConfig.properties file. See "Automatic Synchronization" on page 137.

Manual Synchronization

The manual method involves executing the CLI client **sync** command. The syntax is as follows:

sync {**-all** | **-config** | **-drivetab**} **-ipaddr** *ip_addr* :*sslport* [**-merge** | **-rewrite**]

This command sends the configuration file properties or the drive table information or both from the source (or sending) server to the destination (or receiving) server specified by the **–ipaddr** parameter. The receiving Encryption Key Manager server must be up and running.

Required fields

-all
Send both the configuration properties file and the drive table information to the server specified by **-ipaddr**.

-config
Send only the configuration properties file to the server specified by **-ipaddr**.

-drivetab
Send only the drive table information to the server specified by **-ipaddr**.

-ipaddr
ip_addr:sslport specifies the address and ssl port of the receiving server. The *sslport* should match the value specified for "TransportListener.ssl.port" in the KeyManagerConfig.properties file of the receiving server.

Optional fields

-merge
Merge (add) new drive table data with current data on receiving server. (The configuration file is always a rewrite.) This is the default.

-rewrite
Replace the current data on the receiving server with new data.

Automatic Synchronization

The drive table and properties file can be sent from a primary key manager server to a secondary server automatically. The secondary server must be running for synchronization of the data to occur. To automatically synchronize the data from the primary to the secondary, the following four properties in the primary server KeyManagerConfig.properties file must be specified. There are no changes required to the secondary or receiving server properties file.

sync.ipaddress
Specifies the address and ssl port of the receiving server, for example,

```
sync.ipaddress = backupekm.server.ibm.com:1443
```

If this property is unspecified or specified incorrectly, automatic synchronization is disabled.

sync.action
Merge or rewrite the existing data in the receiving server Valid values are **merge** (default) and **rewrite**. Synchronizing the configuration properties always results in a rewrite.

sync.timeinhours
How often the data should be sent. The value is specified in whole numbers (hours). The time interval begins when the server is started, that is, the synchronization will occur after the server has been running for the specified number of hours. The default is 24.

sync.type
Which data should be sent. Valid values are **drivetab** (default), **config**, and **all**.

If You are Using Hardware Cryptography

Review the following to ensure that your environment meets all requirements.

- **On Open Systems platforms**
  1. If you want a completely secure environment in which all keys reside on the hardware device, be sure that you have updated your PKCS11 configuration file to make sure that all private key and symmetric keys are created and marked as ″sensitive″.
  2. Only the SDK 5.0 JVMs are supported when using hardware cryptography with the Encryption Key Manager. This allows access to full 256-bit AES keys (if supported by your hardware device).
  3. Ensure that you have the corresponding hardware configuration file for your hardware device (SDK 5.0 JVM only). Otherwise the Encryption Key Manager will not start.
  4. The keystore type must be PKCS11ImplKS.
- **On z/OS systems**
  1. Use one of the following keystore types:
     JCE4758KS (SDK 1.4.2)
     JCECCAKS (SDK 5)
     JCE4758RACFKS (SDK 1.4.2)
     JCECCARACFKS (SDK 5)
  2. If you want the RSA key to be completely secure and not visible in the clear, be sure to create your RSA keys in the ICSF PKDS using either the RACDCERT PCICC option or **hwkeytool** with the **-hardwaretype PKDS** flag.
  3. If you want the data encryption key to be completely secure and not visible in the clear be sure that you have changed the key manager configuration to set the requireHardwareProtectionForSymmetricKeys property to true.
  4. Java at the SDK 5 SR3 or greater level is the recommended environment because it contains additional AES support.
  5. Ensure that the IBMJCE4758 (SDK 1.4.2) or IBMJCECCA (SDK 5.0) provider is installed in your java.security provider list. Refer to “Installing on z/OS” on page 45 for more information.
- **On Linux for System z**
  1. The keystore type must be PKCS11ImplKS (see “On distributed system platforms” at the top of this topic for more information).
- **On i5/OS systems**
  1. The IBMi5OSKeyStore option does not support hardware cryptography.

If You are Not Using Hardware Cryptography

Review the following to ensure that your environment will meet all requirements.

- Use a keystore type appropriate for your Operating System.
- Ensure IBM JDK 1.42 SR8 or later, or IBM JDK 5.0 SR5 is installed. For z/OS, the minimum Java level is 1.4.2 SR6 or 5.0 SR3 and is only supported using the 31-bit versions of the SDKs.
- Obtain latest version of IBMKeyManagementServer.jar from the IBM website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 (or visit http://www.ibm.com/servers/storage/support/tape/ts1120/downloading.html and click **downloads** and look for **IBM Encryption Key Manager for the Java platform**).

- Obtain a list of all the aliases (or key labels) for the RSA keys that you want to use. See your keystore documentation for more info on how to get this information.
- Obtain a list of all the type Drive Serial Numbers you will need to register. (Optional when setting `drive.acceptUnknownDrives = true` for automatic addition of tape drives to tape drive table.)
- Edit the KeyManagerConfig.properties file, as shown in "Configuration Basics," to customize the entries appropriate for your installation.

Configuration Basics

This procedure contains the minimum steps necessary to configure the Encryption Key Manager. Appendix A includes examples of how to edit the configuration file. See Appendix B for sample configuration files, and Appendix C for the complete configuration file.

1. Use a key management tool specific to the keystore type you have chosen to create a new keystore. (See "Which Keystore is Right for You" on page 37.) When creating the keystore, take note of the path and filename as well as the names given to the certificates and keys. This information will be used in later steps.

   If you want a completely secure environment in which all keys reside on the hardware device, be sure that you have updated your PKCS11 configuration file to such that that all private key and symmetric keys are created and marked as "sensitive". This is done by adding CKA_SENSITVE=true to the configuration file for private keys. For symmetric keys you need to add both CKA_SENSITVE=true and CKA_EXTRACTABLE=true attributes.
2. Create a keystore if none exists. Add or import the certificates and keys that will be used with your tape drives to this new keystore. (See "Generating Keys and Aliases for Encryption on LTO 4" on page 94.) Take note of the names given to the certificates and keys. This information will be used in later steps.
3. Edit the **KeyManagerConfig.properties** to update the following values.

   Note: The Encryption Key Manager must not be running when you edit the **KeyManagerConfig.properties** file. If you have previously started the Encryption Key Manager server, you must exit it or any changes you make will not be saved.

   a. **Audit.handler.file.directory –** specify a location where audit logs are to be stored.
   b. **Audit.metadata.file.name –** specify a fully qualified path and filename for the metadata XML file.
   c. **config.drivetable.file.url –** specify a location for information about drives that are known to the Encryption Key Manager. This file is not required before starting the server or CLI client. If it does not exist, it will be created during shutdown of the Encryption Key Manager server.
   d. **TransportListener.ssl.keystore.name –** specify the path and filename of the keystore created in step 1.
   e. **TransportListener.ssl.truststore.name** - specify the path and filename of the keystore created in step 1.
   f. **Admin.ssl.keystore.name** - specify the path and filename of the keystore created in step 1.
   g. **Admin.ssl.truststore.name** - specify the path and filename of the keystore created in step 1.

h. **config.keystore.file** - specify the path and filename of the keystore created in step 1.

i. **drive.acceptUnknownDrives** - specify `true` or `false`. A value of true allows new tape drives that contact the Encryption Key Manager to be automatically added to the drive table. The default is false.

4. The following optional password entries may be added or omitted. If these entries are not specified in **KeyManagerConfig.properties**, the Encryption Key Manager will prompt for the keystore password during the startup of the server.

   Note: For i5/OS, the passwords must be specified in the properties file or on the strEKM script call at this time.

   a. **Admin.ssl.keystore.password** - specify the password of the keystore created in step 1.

   b. **config.keystore.password** - specify the password of the keystore created in step 1.

   c. **TransportListener.ssl.keystore.password** - specify the password of the keystore created in step 1.

   When added to the **KeyManagerConfig.properties** file, the Encryption Key Manager obfuscates these passwords for additional security. Passwords that are obfuscated in display cannot be read or copied.

5. Optionally set the **Server.authMechanism** property to a value of `LocalOS` if CLI client authentication is to be done against the local operating system registry. If unspecified (or set to `EKM`) the default is to have the CLI client user login to the key manager server using usr/passwd as EKMAdmin/changeME. (This password can be changed with the **chgpasswd** command.)

   When the **Server.authMechanism** property is set to LocalOS, additional setup is required for Linux, AIX, Solaris & HP platforms. Refer to the readme file on the Encryption Key Manager website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 for more details. "Authenticating CLI Client Users" on page 159 contains more information.

6. Save the changes to **KeyManagerConfig.properties**.

7. Start the Encryption Key Manager server:

```
java com.ibm.keymanager.EKMLaunch KeymanagerConfig.properties
```

   Note: For Encryption Key Manager versions later than 05032007, the Encryption Key Manager server is started with the EKMLaunch command instead of the KMSAdminCmd. In Encryption Key Manager versions with a build date of 05032007 and earlier, the EKMLaunch command is not recognized. Instead use the KMSAdminCmd, for example:/u/ekmserv/:> `java com.ibm.keymanager.KMSAdminCmd`.

   See "Starting and Stopping the Key Manager Server" on page 157 for details.

8. Start the CLI client:

```
java com.ibm.keymanager.KMSAdminCmd CLI_configfile_name -i
```

   See "The Command Line Interface Client" on page 159 for details.

9. Configure a drive by entering the following at the # prompt:

```
adddrive -drivename drive_name -rec1 cert_name -rec2 cert_name
```

   For example:

```
# adddrive -drivename 000001365054 -rec1 key1c1 -rec2 key1c2
```

followed by

```
# listdrives -drivename 000001365054
```

returns

```
Entry Key:  SerialNumber = 000001365054

Entry Key:  AliasTwo = key1c2

Entry Key:  AliasOne = key1c1
 Deleted : false
 Updated : true
 TimeStamp : Sun Jul 03 17:34:44 MST 2007
```

10. Enter the **listdrives** command at the # prompt to ensure the drive was successfully added.

Configuring for LTO 4 Encryption

The management of encryption keys is expected to be performed using existing keystore management utilities and manual synchronization (i.e. extract/export, send, receive, import/insert) of the keys into the key groups and keystores used by the set of Encryption Key Managers employed. Note that with this feature/capability, the names (key IDs, key aliases/labels, key group IDs) of the symmetric keys will be much more apparent to the Encryption Key Manager administrators. The key IDs are not meant to be private or sensitive information.

The expected administrative steps are:

1. Create or import a certificate and private key for key manager-to-key manager communications. See the appropriate topic for your operating environment in Chapter 3, "Installing the Encryption Key Manager and Keystores," on page 45.
2. Generate a set of symmetric encryption keys. See "Generating Keys and Aliases for Encryption on LTO 4" on page 94
3. Create key groups, if desired, and populate with key aliases. See "Creating and Managing Key Groups" on page 100.
4. For EACH key manager, store these keys into the keystore. This is implicit in the invocation of KeyTool with the **–storetype** `jceks` and **–keystore** <*filename*> options. After a suitable number of standalone (-alias) and/or ranged (-aliasrange) invocations have been issued with KeyTool using the above two options, the named keystore file from the command can be specified as a value for the `config.keystore.file` environment variable. The Encryption Key Manager supports formats other than JCEKS as well, see "Which Keystore is Right for You" on page 37.

   Note: If the zOSCompatibility property is set to `true`, then **–keyAlg** must be set to `DESede` (3-DES) with an implicit effective bit length of 168 that need not and should not be specified. Otherwise, **-keyalg** must be `AES` and **-keysize** must be `256`.
5. For each key manager, change the Encryption Key Manager configuration to set the key groups or key aliases and/or ranges to refer to the newly created keys. This is done using the configuration file property, `symmetricKeySet`. These aliases can be set up to match the aliases set up from KeyTool from steps 1 and 2 above. All ranged and standalone aliases can be specified all at once, delimited by commas.
6. After all key manager configurations have been updated, for each Encryption Key Manager, restart the Encryption Key Manager to incorporate the configuration changes.

Configuring on z/OS

z/OS-Specific Configuration Properties

The Encryption Key Manager on z/OS can take advantage of the z/OS hardware cryptography provided by z/OS ICSF to improve the security characteristics of the data encryption key generated by the Encryption Key Manager on z/OS. The following configuration properties, requireHardwareProtectionForSymmetricKeys and zOSCompatibility, should be considered when running the Encryption Key Manager on the z/OS platform.

In particular the **requireHardwareProtectionForSymmetricKeys** and **zOSCompatibility** configuration properties implement enhanced symmetric key handling in support of an Encryption Key Manager running on z/OS, which ensures that tape data encryption keys can be generated, wrapped, and rewrapped under multiple RSA keys utilizing z/OS ICSF services and residing in hardware-protected locations. The Encryption Key Manager may be configured with these properties so that keys that are sent or received from the tape drive and the keys used to encrypt data will not appear in a unencrypted form in z/OS host storage (system memory, expanded storage, paging space). z/OS ICSF services and zSeries hardware cryptography can be used to secure the RSA key management of symmetric keys. At present z/OS ICSF does not currently handle (generate or store) 256-bit AES data keys in a manner that would prevent these keys from appearing in an unencrypted form in host storage.

The Encryption Key Manager, when configured with the zOSCompatibility property set to true, uses the configured JCE cryptographic provider and causes a 168-bit TDES key to be generated in lieu of a 256-bit AES key. This key, wrapped using an RSA key which is protected by hardware cryptographic services, is then provided to the tape drive device. The tape drive device continues to use 256-bit AES-GCM encryption, using the 168-bit key to build a 256-bit AES key that is subsequently used for data encryption and decryption performed within the device. When the Encryption Key Manager is running on z/OS and **requireHardwareProtectionForSymmetricKeys** is set to true, this key is always encrypted in z/OS host storage. The following tables provide additional information on these Encryption Key Manager configuration properties.

requireHardwareProtectionForSymmetricKeys configuration property

Table 22. requireHardwareProtectionForSymmetricKeys property

Value	Applies to	Description and usage
true | false	Writing and reading tapes on the z/OS platform only when using an Encryption Key Manager started with any of the jce4758/jcecca provider based keystores.	If `true`, the data encryption key used with the JCE4758KS, JCE4758RACFKS, JCECCA, or JCECCARACFKS keystore will be protected by z/OS cryptographic hardware. This means that data encryption key generated for encryption and decryption will only appear in host storage in an encrypted form that is protected by a hardware resident master key. This option is specific to z/OS and only affects the z/OS jce4758/jcecca provider keystore types listed above. It has no effect on other keystore types, and no effect on other supported Encryption Key Manager platforms (other than z/OS).

zOSCompatibility configuration property

Table 23. zOSCompatibility property

Values	Applies to	Description and usage
true	Encrypting/Writing to Tapes on all Platforms – by specifying true, you are allowing the tape being encrypted to be read by an Encryption Key Manager running on the z/OS Platform.	Use this option if plan to exchange tapes between z/OS and non-z/OS systems. **Note:** If Encryption Key Manager configuration file parameter requireHardwareProtectionForSymmetricKeys is set to true, the zOSCompatibility parameter must be set to true to enable the keys generated by the Encryption Key Manager to always be encrypted in host storage. This option affects ALL keystores that may be configured to the Encryption Key Manager on all platforms that are supported by the Encryption Key Manager.
false	Encrypting/Writing to Tapes on all Platforms – by specifying false, the encrypted tape can only be read by an Encryption Key Manager running on the z/OS platform if the Encryption Key Manager is running with SDK 5.0 (it cannot be decrypted by an Encryption Key Manager running with SDK 1.4.2). In addition, you cannot use the z/OS encrypted key support as provided by ICSF and zSeries hardware assisted cryptography.	

Table 24 and Table 25 summarize the effect of two Encryption Key Manager configuration file parameters, requireHardwareProtectionforSymmetricKeys and zOSCompatability, when the Encryption Key Manager is deployed in a Java SDK 1.4.2 or a Java SDK 5.0 environment.

Table 24. z/OS Encryption Key Manager Configuration and Tape Writing Operations

		zOSCompatibility	
		True	**False (default)**
requireHardwareProtection ForSymmetricKeys	**True**	Tape encryption data key that is generated is protected by the ICSF Master key. This tape can be read by z/OS and non-z/OS platforms who also have zOSCompatibility flag set to true.	Unsupported due to cryptographic capability restrictions.
	False	Tape encryption data key that is generated is protected by a key that can appear in host storage and is NOT protected by the ICSF Master key. This tape can be read by z/OS and non-z/OS platforms who also have zOSCompatibility flag set to true.	Tape encryption data key that is generated is protected by a key that can appear in host storage and is NOT protected by the ICSF Master key. This tape can be read by non-z/OS platforms regardless of the zOSCompatibility flag setting. This tape can only be read by z/OS platforms that are running their Encryption Key Manager with java SDK 5.0 and have zOSCompatibility set to false. This is not supported in Java 1.4.2 due to cryptographic restrictions.

Table 25. z/OS Encryption Key Manager Configuration and Tape Reading Operations

		zOSCompatibility	
		True	**False (default)**
requireHardwareProtection ForSymmetricKeys	**True**	Encryption Key Manager returns the data key needed to decrypt the tape. The data key is protected by the ICSF Master Key.	Unsupported due to cryptographic capability restrictions.
	False (default)	Encryption Key Manager returns the data key needed to decrypt the tape. The data key is protected by a key that can appear in host storage (i.e., it is not protected by the ICSF Master key).	Encryption Key Manager returns the data key needed to decrypt the tape. The data key is protected by a key that can appear in host storage (i.e., it is not protected by the ICSF Master key). This is not supported in Java 1.4.2 due to cryptographic restrictions.

Create Encryption Key Manager Configuration File

Create the Encryption Key Manager configuration file in /&SYSNAME/etc and customize accordingly for your installation

Audit.handler.file.directory
Modify this to a location where you want the Encryption Key Manager to store the audit logs.

Audit.metadata.file.name
Specify a filename for the metadata XML file.

zOSCompatibility
Use this option if you intend to exchange tapes between z/OS and non-z/OS systems. If the Encryption Key Manager configuration file parameter requireHardwareProtectionForSymmetricKeys is set to true this value must be set to true also. This applies to ALL supported platforms.

config.drivetable.file.url
Specify a location for information about drives that are known to the Encryption Key Manager. This file is not required to exist before starting the Encryption Key Manager server or Encryption Key Manager Admin console. If it does not exist, it will be created during shutdown of the Encryption Key Manager server of Encryption Key Manager Admin Console.

Admin.ssl.keystore.name
Admin.ssl.truststore.name
config.keystore.file
TransportListener.ssl.keystore.name
TransportListener.ssl.truststore.name
Specify the path and filename of the keystore created previously.

requireHardwareProtectionForSymmetricKeys
This option gives the user the ability to define if the data encryption key used with the JCE4758KS, JCE4758RACFKS, JCECCA, or JCECCARACFKS keystores are to be protected by z/OS cryptographic hardware. This means that keys generated and used by the Encryption Key Manager will only appear in host storage in an encrypted form that is protected by a hardware resident master key. This option applies to Encryption Key Manager instances on z/OS

drive.acceptUnknownDrives
Specify true or false. A value of true allows new tape drives that contact the Encryption Key Manager to be automatically added to the drive table. The default is false. If you specify true for this value then we suggest that you also set drive.default.alias1 and drive.default.alias2 to the certificate alias(es)/key label(s) created in above.

The following example illustrates an Encryption Key Manager configuration file using the JCE4758RACFKS that has been customized for a z/OS system that is using shared HFS where systemname = JA0.

```
Admin.ssl.keystore.name = safkeyring://EKMSERV/EKMRing
Admin.ssl.truststore.name = safkeyring://EKMSERV/EKMRing
Audit.event.outcome = success,failure
Audit.event.outcome.do = success,failure
Audit.event.types = all
Audit.event.types.backup = authentication,authorization,data synchronization,
  runtime,audit management,authorization terminate,configuration management,
  resource management,none
Audit.eventQueue.max = 0
Audit.handler.file.directory = /ekmlogs/JA0/audit
Audit.handler.file.name = kms_audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /keymanager/metafile.xml
config.drivetable.file.url = FILE:/ekmetc/JA0/filedrive.table
config.keystore.file = safkeyring://EKMSERV/EKMRing
config.keystore.password = password
config.keystore.provider = IBMJCE4758
config.keystore.type = JCE4758RACFKS
debug = none
debug.output = simple_file
debug.output.file = /ekmlogs/JA0/debug
drive.acceptUnknownDrives = true
drive.default.alias1 = EKMServer
drive.default.alias2 = EKMServer
fips = Off
requireHardwareProtectionForSymmetricKeys = true
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = safkeyring://EKMSERV/EKMServer
TransportListener.ssl.keystore.password = password
TransportListener.ssl.keystore.type = JCE4758RACFKS
TransportListener.ssl.port = 1443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = safkeyring://EKMSERV/EKMServer
TransportListener.ssl.truststore.type = JCE4758RACFKS
TransportListener.tcp.port = 3801
zOSCompatibility = true
```

Figure 63. Encryption Key Manager Configuration File: KeyManagerConfig.properties.jce4758racfks

z/OS Java Levels

Ensure that a level of Java for z/OS is installed that contains the JZOS launcher code. For SDK 1.4.2 this is SR 6 or higher. For SDK 5.0 this is SR 3 or higher. Complete installation instructions are contained on the Java for z/OS website at http://www-03.ibm.com/servers/eserver/zseries/software/java.

Note About z/OS Configuration Steps for z/OS In-band Encrypted Tape Drive

Depending on whether you are using z/OS in-band key management, there will be additional configuration steps that you will need to perform. One of these steps may be the identification of the port that the z/OS IOS subsystem (proxy support) will be using to communicate with the Encryption Key Manager. The specification of this port will need to be specified in the Encryption Key Manager configuration and the appropriate parmlib member that is in use by the z/OS IOS component. For additional information on setting up the z/OS IOS subsystem, including
| specification of the appropriate port ID, refer to *z/OS DFSMS Software Support for*
| *IBM System Storage TS1130 and TS1120 Tape Drives (3592)*, SC26-7514.

Migrating from SDK 1.4.2 to SDK 5.0 on z/OS with the IBM Hardware Crypto Provider

The IBM hardware crypto provider IBMJCE4758 in SDK 1.4.2 is renamed to IBMJCECCA in SDK 5.0. After migrating from SDK 1.4.2 to SDK 5.0, all occurrences of IBMJCE4758 should be replaced by IBMJCECCA in all the following locations.

All Instances of Keystores

JCERACF4758KS must be replaced with JCERACFCCAKS

JCE4758KS must be replaced with JCECCAKS

java.security File in the Path JAVA_HOME/lib/security

Change the IBM hardware crypto provider name from `security.provider.x=com.ibm.crypto.hdwrCCA.provider.IBMJCE4758` to `security.provider.x=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA`.

If using the Encryption Key Manager, Update Configuration File EKM2ENV

# Set JZOS specific options

export ZZZZ="KeyManagerConfig.properties"

Go to the above mentioned line in the file EKM2ENV and make sure you point to the appropriate KeyManagerConfig.properties file. The properties file depends on the type of the keystore you are using. If you are modifying the file KeyManagerConfig.properties, then the following entries should be changed:

```
TransportListener.ssl.truststore.name =
TransportListener.ssl.truststore.type =
TransportListener.ssl.keystore.name =
TransportListener.ssl.keystore.type =
TransportListener.ssl.keystore.password =

config.keystore.provider =
config.keystore.type =
config.keystore.file =
config.keystore.password =

Admin.ssl.truststore.name =
Admin.ssl.keystore.name =
Admin.ssl.keystore.password =
```

If you are already using JCERACF4758KS/ JCE4758KS, it must be replaced with JCERACFCCAKS/ JCECCAKS accordingly.

If Using JZOS, Update the PROC JVMPRC50 to Use the Program JVMLDM50 Instead of JVMLDM14

To do this, simply update the line `VERSION='14'` to `VERSION='50'`.

Ensure the LIBPATH, CLASSPATH and PATH are All Set Properly for SDK 5.0

Check these variables for the correct value for java5.

Configuring on AIX and other Operating Systems

Note to Windows Users: Windows does not accept commands with directory paths that contain blanks. When entering commands it may be necessary to specify the short name generated for such directories, for example `progra~1` instead of `Program Files`. To list directory short names, issue the **dir /x** command.

1. Download the latest IBMKeyManagementServer.jar and KeyManagerConfig.properties from the website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 (or visit http://www.ibm.com/servers/storage/support/tape/ts1120/downloading.html and click **downloads** and look for **IBM Encryption Key Manager for the Java platform**) into a directory of your choice. In our example, we used the following directory:

```
(iot@ost28.storage.tucson.ibm.com)/tape/Encryption/jkm/ost28 # ls -al
total 396304
drwxr-sr-x   3 root     sys            512 Apr 30 00:12 .
drwxr-s---  30 root     sys           1536 Apr 29 18:39 ..
-rw-r-----   1 root     sys         271949 Apr 30 00:12 IBMKeyManagementServer.jar
-rw-r--r--   1 root     sys           2310 Apr 29 22:02 KeyManagerConfig.properties
```

2. Copy the IBMKeyManagementServer.jar file to the same directory.
   a. For a UNIX-based system, copy to `/tape/Encryption/jkm/ost28`
   b. For Windows-based system, copy to `C:\"Program Files"\IBM\Java50\jre\lib\ext\`
3. Edit the **KeyManagerConfig.properties** file. Please note that the current design of the server is very strict. Do not use Windows to edit the file for a UNIX machine because of ^M. If you use Windows, edit the file with gvim/vim.

   Note to Windows Users: The Java SDK uses forward slashes, even when running on Windows. When specifying paths in the **KeyManagerConfig.properties** file, be sure to use forward slashes. When specifying a fully-qualified path name in the command window, use back slashes in the normal manner for Windows.

   a. Find and modify the following properties in the **KeyManagerConfig.properties** file:

      Admin.ssl.keystore.name
      Modify this to the path and filename specified when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

      ```
      Admin.ssl.keystore.name = /tape/Encryption/jkm/keys/ost28/key1.jks
      ```

      Admin.ssl.truststore.name
      Modify this to the path and filename specified when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

      ```
      Admin.ssl.truststore.name = /tape/Encryption/jkm/keys/ost28/key1.jks
      ```

      Audit.handler.file.directory
      Modify this to a location where you want to store the audit logs. This directory MUST exist before starting the server or CLI client.

      ```
      Audit.handler.file.directory = /tape/Encryption/jkm/ost28/auditlogs/
      ```

      Audit.metadata.file.name
      Specify a filename for the metadata XML file.

      ```
      Audit.metadata.file.name = /keymanager/metafile.xml
      ```

TransportListener.ssl.keystore.name
Modify this to the path and filename of the keystore created previously.

```
TransportListener.ssl.keystore.name = /tape/Encryption/jkm/keys/ost28/key1.jks
```

TransportListener.ssl.keystore.password
Modify this to the password specified when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

```
TransportListener.ssl.keystore.password =
password
```

TransportListener.ssl.truststore.name
Modify this to the path and filename specified when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

```
TransportListener.ssl.truststore.name = /tape/Encryption/jkm/keys/ost28/key1.jks
```

config.drivetable.file.url
Modify this to a path and filename where you want to store the information about known drives. This directory MUST exist before starting the server or CLI client. Please note that the "FILE://" must be before the path

```
config.drivetable.file.url = FILE:///tape/Encryption/jkm/ost28/drvinfo.txt
```

config.keystore.file
Modify this to the path and filename specified when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

```
config.keystore.file = /tape/Encryption/jkm/keys/ost28/key1.jks
```

debug.output.file
Modify this to the path and filename of a debug file you would like to create.

```
debug.output.file = /tape/Encryption/jkm/ost28/debug.log
```

b. Add the following properties to your **KeyManagerConfig.properties** file:

config.keystore.password
Specify the password defined when creating the keystore in "Create a Keystore and Certificates (Unix-based and Windows Operating Systems)" on page 87.

```
config.keystore.password = password
```

TransportListener.tcp.timeout
Specify a number in minutes.

```
TransportListener.tcp.timeout = 120
```

drive.acceptUnknownDrives
Set this property to `true` to automatically add tape drives to the drive table when they contact the Encryption Key Manager. See "Configuration Strategies" on page 135 for more information.

```
drive.acceptUnknownDrives = true
```

symmetricKeySet
Specify one value for *GroupID* or one or more values for *keyAliasList* to establish default symmetric keys for use when no alias is defined for tape drive. See "Generating Keys and Aliases for Encryption on LTO 4" on page 94 and "Creating and Managing Key Groups" on page 100 for more information.

c. Save the changes to the **KeyManagerConfig.properties** file.

Note: The order of lines in the **KeyManagerConfig.properties** file may change after the Encryption Key Manager rewrites the file while running.

4. Start the Encryption Key Manager server by entering the **EKMLaunch** command from any shell or command window:

   ```
   java com.ibm.keymanager.EKMLaunch KeymanagerConfig.properties
   ```

   This starts the server in the background. On Windows, the server can run as a Windows Service. See "Starting and Stopping the Key Manager Server" on page 157 for more information.

5. Start the CLI Client.

   a. On a UNIX-based system, enter the following at the prompt:

      ```
      ost28> java com.ibm.keymanager.KMSAdminCmd CLI_configfile_name -i
      Keystore password:          *********
      # login -ekmuser EKMAdmin -ekmpassword changeME
      User successfully logged in.
      ```

      See "The Command Line Interface Client" on page 159 for more information.

   b. On a Windows-based system, enter the following at the prompt:

      ```
      C:\WinEKM>C:\progra~1\IBM\Java50\jre\bin\java com.ibm.keymanager.KMSAdminCmd
      CLI_configfile_name -i
      Keystore password:          *********
      # login -ekmuser EKMAdmin -ekmpassword changeME
      User successfully logged in.
      ```

6. Before using the Encryption Key Manager, you must add any tape drives you wish to use for encryption to the drive table. If you did not add `drive.acceptUnknownDrives = true` to the **KeyManagerConfig.properties** file, specify drives manually, entering the following command in the CLI client for each drive:

   ```
   adddrive -drivename  drivename -rec1 alias1 -rec2 alias2
   ```

   where -rec1 and -rec2 are optional. They are the default keys assigned to your drive in the event that the key requested for use with a cartridges loaded in your drive is not present in the keystore. *alias1* and *alias2* are two different self signed certificates or key labels in the keystore.

   In our example, we used the following command:

   ```
   # adddrive -drivename 000001365054 -rec1 key1c1 -rec2 key1c2
   # listdrives -drivename 000001365054
   -----------------------------------------------

   Entry Key:  SerialNumber = 000001365054

   Entry Key:  AliasTwo = key1c2

   Entry Key:  AliasOne = key1c1
    Deleted : false
    Updated : true
    TimeStamp : Sun Apr 29 17:34:44 MST 2007

   -----------------------------------------------
   ```

7. To stop the key manager server enter stopekm:

   ```
   # stopekm
   Stopping the KMS admin service...
   # status
   Server is not started
   #
   ```

Configuring on i5/OS

1. Install the latest Encryption Key Manager code. See "Install the IBM Software Developer Kit (i5/OS)" on page 103.
2. Copy of the **KeyManagerConfig.properties** file into another directory so that it is not overwritten when installing the next release. For example,

   ```
   CPY OBJ('/QIBM/ProdData/OS400/Java400/ext/KeyManagerConfig.properties') TODIR('/EKM')
   ```

   Note: A default configuration file can also be downloaded from this URL (http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504)
3. Edit the copied version of the **KeyManagerConfig.properties** file. Please note that the current design of the server is very strict. If there are extra spaces or line returns after the filenames/passwords, you will have a problem with next sped. Ensure that for the properties you edit, the last character on the line is not a space. Do not use Windows to edit the file for a UNIX machine because of ^M. If you use Windows, edit the file with **gvim/vim**.
   a. Find and modify the following properties in the **KeyManagerConfig.properties** file:

      Audit.handler.file.directory
      Modify this to a location where you want the Encryption Key Managerto store the audit logs. This directory MUST exist before starting the Encryption Key Manager server or Admin client. For example, `Audit.handler.file.directory = /EKM/auditlogs/`

      Audit.metadata.file.name
      Specify a filename for the metadata XML file. For example, `Audit.metadata.file.name = /EKM/metadata/metafile.xml`

      Admin.ssl.keystore.name
      Admin.ssl.truststore.name
      config.keystore.file
      TransportListener.ssl.keystore.name
      TransportListener.ssl.truststore.name
      Modify these to the path and filename specified when creating the keystore in "Using Digital Certificate Manager to Create and Manage Keystores on i5/OS" on page 105. For example,

      ```
      Admin.ssl.keystore.name = /EKM/EKM.kdb
      Admin.ssl.truststore.name = /EKM/EKM.kdb
      config.keystore.file = /EKM/EKM.kdb
      TransportListener.ssl.keystore.name = /EKM/EKM.kdb
      TransportListener.ssl.truststore.name = /EKM/EKM.kdb
      ```

      Note: If using a JCEKS keystore to support LTO 4 tape drives, the keystore name would have a different extension. For example, `/EKM/EKM.jceks` instead of `/EKM/EKM.kdb`.

      Admin.ssl.keystore.password
      config.keystore.password
      TransportListener.ssl.keystore.password
      Modify these to the password specified when creating the keystore in "Using Digital Certificate Manager to Create and Manage Keystores on i5/OS" on page 105.

      ```
      Admin.ssl.keystore.password = kspwd
      config.keystore.password = kspwd
      TransportListener.ssl.keystore.password = kspwd
      ```

      config.drivetable.file.url
      Modify this to a path and filename where you want the Encryption Key Manager to store the information on the drives that are known to the

Encryption Key Manager. This directory MUST exist before starting the Encryption Key Manager server or Admin client. Please note that the ″FILE://″ must be before the path, for example,

```
config.drivetable.file.url = FILE:///EKM/drives/drivetable
```

debug.output.file
Modify this to the path and filename of a debug file you would like to create. For example,

```
debug.output.file = /EKM/debug.log
```

drive.acceptUnknownDrives
Specify **true** or **false**. A value of true allows new tape drives that contact the Encryption Key Manager to be automatically added to the drive table. The default configuration file is shipped with a value of true. If you leave the value as true, then we suggest that you also add the drive.default.alias1 and drive.default.alias2 attributes, specifying the certificate label(s) specified when creating the keystore in "Using Digital Certificate Manager to Create and Manage Keystores on i5/OS" on page 105. For example:

```
drive.default.alias1 = EKM certificate 1
drive.default.alias2 = EKM certificate 2
```

symmetricKeySet
Add this attribute to specify the symmetric key set when supporting LTO 4 tape drives. For example,

```
symmetricKeySet = ekm01-10
```

If using a JCEKS keystore, then the following attributes will need to be changed from the values specified in the default configuration file:

```
config.keystore.provider = IBMJCE
TransportListener.ssl.truststore.type = jceks
TransportListener.ssl.keystore.type = jceks
config.keystore.type = jceks
```

If using an IBMi5OSKeyStore, and the file was downloaded from the Encryption Key Manager website, then the following attributes will need to be changed from the values specified in the default configuration file:

```
config.keystore.provider = IBMi5OSJSSEProvider
config.keystore.type = IBMi5OSKeyStore
TransportListener.ssl.keystore.type = IBMi5OSKeyStore
TransportListener.ssl.truststore.type = IBMi5OSKeyStore
```

b. Save the changes to the **KeyManagerConfig.properties** file.

4. Start the Encryption Key Manager CLI Client.

If you are using the 05032007 version of Encryption Key Manager, or an earlier version, use the **strEKM** script that is provided, as shown in step 4a.

| If you are using a version of Encryption Key Manager that is more recent than
| the 05032007 version, the **strEKM** script cannot be used. Instead use the
| **KMSAdminCmd** command shown in step 4b on page 154.

| **Note:** V6R1 ships with a later version of Encryption Key Manager.

a. This is the help for the **strEKM** script:

strEKM [-h] [-cmd |-server] [-propfile *properties_file_pathname*]**[-kspwd** *password*]

-h Shows the help for the command.

-cmd|-server
Indicates if the Encryption Key Manager command prompt is to be started or the Encryption Key Manager server (starts Encryption Key

Manager command prompt and issues the **startekm** command). If not specified, **-cmd** is assumed (optional).

-propfile
: Pathname to the properties file to be used by the Encryption Key Manager. If not specified, then the shipped default pathname of **/QIBM/ProdData/OS400/Java400/ext/KeyManagerConfig.properties** is used (optional).

-kspwd
: The password to the keystore specified in the KeyManagerConfig.properties file. This value is only used if the -server option is used. It is also not used if the keystore password is already specified in the properties file

To start the Encryption Key Manager Admin Console, do the following:

Call the QSH command

From within QSH, use the **strEKM** script to start the Encryption Key Manager command prompt, specifying the pathname to the previously modified properties file:

```
===>  strEKM -propfile /EKM/KeyManagerConfig.properties
Apr 30, 2007 5:26:59 PM com.ibm.keymanger.config.ConfigImpl get
FINER: ENTRY
Apr 30, 2007 5:27:00 PM com.ibm.keymanger.config.ConfigImpl get
ALL: debug.output = simple_file
Apr 30, 2007 5:27:00 PM com.ibm.keymanger.config.ConfigImpl get
FINER: RETURN
#
```

b. If using a version of Encryption Key Manager Encryption Key Manager that is more recent than the 05032007, to start the Encryption Key Manager CLI client, call the QSH command:

```
java com.ibm.keymanager.KMSAdminCmd CLI_configfile_name -i
```

| **Note:** On i5/OS V5R4 only, ensure that you have set the JAVA_HOME environment variable to `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit`

5. Before using the Encryption Key Manager, you must add any tape drives you wish to use for encryption to the Encryption Key Manager drive table. If you did not set up the **KeyManagerConfig.properties** file to automatically add new drives, then you need to add the drives manually. For example, enter the following command for each drive (note that if there are spaces in the certificate label, the quotation marks are required):

```
adddrive -drivename 000001234567 -rec1 "EKM certificate 1"  -rec2 "EKM certificate 2"
```

6. Start the Encryption Key Manager server. To start the server in a batch job, issue one of the following commands. With these commands, the server will run in the QUSRWRK subsystem. To make it easier to find and manage the Encryption Key Manager server job, you may want to create a separate subsystem in which to run the server instead of using QUSRWRK. Also, these commands will send stdout and stderr messages to the specified log files. These will aid in the diagnosis of runtime errors in the server.
   a. If using the 05032007 version of Encryption Key Manager, or an earlier version:

```
SBMJOB CMD(QSH CMD('strEKM -server -propfile /EKM/KeyManagerConfig.properties 1> /EKM/stdout.log
      2> /EKM/stderr.log')) JOBQ(QSYS/QUSRNOMAX)
```

   b. If using a version of Encryption Key Manager that is more recent than the 05032007 version, issue the **QSH** command. Then start the Encryption Key Manager :

```
java com.ibm.keymanager.EKMLaunch KeyManagerConfig.properties
```

Note: On i5/OS V5R4 only, ensure that you have set the JAVA_HOME environment variable to `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit` prior to running the **QSH** command.

7. To stop the Encryption Key Manager server when using the 05032007 version, you must end the submitted job. Be sure to use OPTION(*CNTRLD) when ending the Encryption Key Manager server job.

   To stop the Encryption Key Manager server when using a more recent version, follow the procedure in "Starting and Stopping the Key Manager Server" on page 157.

Configuring the Encryption Key Manager to Use Two Keystores

The default configuration instructions for the Encryption Key Manager suggest using the same keystore for storing certificates and keys for the TS1120 or TS1130 Tape Drives and for SSL operations. The JSSE provider chooses the certificates at random from the keystore to be used for SSL handshakes, and fails when an expired certificate is used. As certificates in the keystore age, Encryption Key Manager SSL operations may risk failure due to expired certificates. The risk of failed SSL operations can be reduced by configuring the Encryption Key Manager to use two separate keystores, one for storing the keys and certificates for the TS1120 or TS1130 drives, and the other to hold keys and certificates for SSL operations. When the certificates in the SSL keystore expire, they can be deleted and recreated without affecting the Encryption Key Manager's ability to serve keys to TS1120 or TS1130 Tape Drives.

The following steps show how to configure the Encryption Key Manager to use two keystores. These steps can be used regardless whether the Encryption Key Manager is uninstalled or unused for any period of time. These instructions assume that both keystores are to be JCEKS keystores. Alter the keystore type as appropriate for your environment.

1. Create a new keystore and self-signed certificate for Encryption Key Manager SSL operations using **keytool**. This **keytool** command creates a new JCEKS keystore called SSLKeystore.jck and populates it with a certificate and private key with an alias of sslcert. This certificate is valid for five years and should have a sufficiently long life to prevent the need for frequent recreation.

   ```
   keytool –keystore SSLKeystore.jck –storetype jceks –genkey –alias sslcert –keyalg RSA
         –keysize 2048 –validity 1825
   ```

   This command prompts you for information it uses to create a distinguished name in the certificate. The prompts, with sample responses, look similar to these:

   ```
   What is your first and last name? [Unknown]: sslcert
   What is the name of your organizational unit? [Unknown]: EKM
   What is the name of your organization? [Unknown]: IBM
   What is the name of your City or Locality? [Unknown]: Austin
   What is the name of your State or Province? [Unknown]: TX
   What is the two-letter country code for this unit? [Unknown]: US
    Is CN=sslcert, OU=EKM, O=IBM, L=Austin, ST=TX, C=US correct?(type "yes" or "no"):
   ```

   Type `yes` and press Enter.

   This command prompts you for a password to access the keystore. Please note the keystore password entered here as it will be needed later when starting the Encryption Key Manager. When prompted for a *key* password, just press Enter. Do not type in a new or different password.

On Solaris and HP-UX, issuing the **keytool** command as shown above does not load the necessary IBM classes. Instead, use **java com.ibm.crypto.tools.KeyTool** with the same parameters, as in:

```
java com.ibm.crypto.tools.KeyTool –keystore SSLStore.jck -storetype jceks
      -genkey -alias sslcert -keyAlg RSA -keysize 2048 –validity 1825
```

2. Create a new keystore and self-signed certificate for use in tape encryption operations for TS1120 or TS1130 Tape Drives. Skip this step if the Encryption Key Manager has been previously configured and already has a keystore used to hold the keys and certificates used for encryption with TS1120 or TS1130 Tape Drives.

   This **keytool** command creates a new JCEKS keystore called EKMKeys.jck and populates it with a certificate and private key with an alias of ekmcert. This certificate is valid for one year.

```
keytool -keystore EKMKeys.jck -storetype jceks -genkey -alias ekmcert -keyAlg RSA
      -keysize 2048 –validity 365
```

   This command prompts you for information it uses to create a distinguished name in the certificate. The prompts, with sample responses, look similar to these:

```
What is your first and last name? [Unknown]: ekmcert
What is the name of your organizational unit? [Unknown]: EKM
What is the name of your organization? [Unknown]: IBM
What is the name of your City or Locality? [Unknown]: Austin
What is the name of your State or Province? [Unknown]: TX
What is the two-letter country code for this unit? [Unknown]: US
 Is CN=ekmcert, OU=EKM, O=IBM, L=Austin, ST=TX, C=US correct?(type "yes" or "no"):
```

   Type **yes** and press Enter.

3. Edit the Encryption Key Manager server properties file (KeyManagerConfig.properties) to use the new SSLKeystore.jck keystore for its SSL operations. The following properties must be updated to point to the new keystore. The password properties will be obfuscated on the next start of the Encryption Key Manager server.

```
Admin.ssl.keystore.name = SSLKeystore.jck
Admin.ssl.keystore.password =  SSLKeystore.jck_password
Admin.ssl.truststore.name = SSLKeyStore.jcks
TransportListener.ssl.keystore.name = SSLKeystore.jck
TransportListener.ssl.keystore.password = SSLKeystore.jck_password
TransportListener.ssl.truststore.name = SSLKeystore.jck
```

4. Verify that other keystore properties in KeyManagerConfig.properties are still correct for tape operations and that they point to the tape encryption operations keystore. In this example EKMKeys.jck.

```
config.keystore.file = EKMKeys.jck
config.keystore.password = EKMKeys.jck_password
```

5. Edit the Encryption Key Manager CLI Client properties file to use the new SSLKeystore.jck when connecting to the Encryption Key Manager Server. The following properties must be modified. The password properties will be obfuscated on the next start of the Encryption Key Manager CLI Client.

```
TransportListener.ssl.keystore.name = SSLKeystore.jck
TransportListener.ssl.keystore.password = SSLKeystore.jck_password
TransportListener.ssl.truststore.name = SSLKeystore.jck
```

Chapter 6. Administering the Encryption Key Manager

Starting and Stopping the Key Manager Server

The Encryption Key Manager server is very easy to start and stop.

Starting and Stopping the Key Manager Server from the Command Prompt

To start the Encryption Key Manager server from any command window or shell (on platforms other than z/OS), enter:

```
java com.ibm.keymanager.EKMLaunch KeymanagerConfig.properties
```

Note: For Encryption Key Manager versions with a build date of 05032007 and earlier, the EKMLaunch command is not recognized. Instead use the KMSAdminCmd, for example:`/u/ekmserv/:> java com.ibm.keymanager.KMSAdminCmd`.

(For z/OS, refer to "Set Up and Run Encryption Key Manager in z/OS Production Mode" on page 72.) This launches the Encryption Key Manager server in the background. When started correctly, the Encryption Key Manager Java process can be displayed with the `ps -ef | grep java` command (AIX and other Unix platforms) or using the Windows Task Manager. When running as a Windows Service, it displays as LaunchEKMService.

To stop the server, issue the **stopekm** command using any of the methods described below in "The Command Line Interface Client" on page 159. Another method is to send a **sigterm** to the key manager process. This allows the server to shutdown and end cleanly. Do not send a **sigkill** to the key manager process. **sigkill** will not shut the process down cleanly. For example, on Linux systems, enter `kill -SIGTERM` *`pid`* or `kill -15` *`pid`*.

On Windows platforms, when Encryption Key Manager is started as a Windows Service, it can be stopped from the Control Panel.

Installing the Key Manager Server as a Window Service

On Windows, the Encryption Key Manager server can be installed as a Windows Service or can be started from a command line as described above. To run the server as a Windows Service, you must manually download the binaries for LaunchEKMService.exe from the Encryption Key Manager website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504. Please refer to the readme file on this website for instructions.

Set the following system variables:

- Create a system variable called JAVA_HOME:
  1. From the Start menu, select Control Panel
  2. Double click on **System**
  3. Click the **Advanced** tab
  4. Click **Environment Variables**
  5. Under the list of System Variables click **New**.

6. Specify JAVA_HOME as the variable name and enter the IBM JVM directory, for example C:\ibm-sdk1.4.2. Click **OK**.

- Edit the system PATH variable using this procedure. Setting the PATH variable from the command line will not work.
  1. From the Start menu, select Control Panel
  2. Double click on **System**
  3. Click the **Advanced** tab
  4. Click **Environment Variables**
  5. Scroll the list of System Variables for the Path variable and click **Edit**.
  6. Add the IBM JVM to the beginning of the Path variable. Click **OK**.
- Ensure the paths in your Encryption Key Manager Server Configuration properties file are fully qualified. All the paths listed below must have a fully qualified path (for example, they must be specified as ″c:\ekm\gui\EKMKeys.jck″ not ″gui\EKMKeys.jck″). Make the necessary changes to ensure your paths are correct.

  config.keygroup.xml.file

  Admin.ssl.keystore.name

  Audit.metadata.file.name

  TransportListener.ssl.truststore.name

  Audit.handler.file.directory

  config.keystore.file

  TransportListener.ssl.keystore.name

  Admin.ssl.truststore.name

  config.drivetable.file.url

Note: You must start the Encryption Key Manager Windows Service manually from a command prompt the first time it is used, by navigating to **Start > Programs > Accessories > Command Prompt**. From the command prompt, navigate to the temporary directory where the LaunchEKMService.exe was extracted.

The LaunchEKMService.exe command has the following options:

LaunchEKMService [-help | -i *config_file* **| -u]**

-help
Displays this usage information.

-i Installs the key manager Windows Service using the properties specified in *config_file*, which should contain full path names for all properties listed either as files or URLs. Once the service is installed, you can start and stop the Windows Service from Control Panel. The configuration file needs to be specified with this option. If the configuration file does not have all keystore passwords specified, you will be prompted for those. When the Windows Service is started, all the passwords will be obfuscated and stored in the configuration file so no password is stored in clear text in the configuration file after the first run.

-u Uninstalls the key manager Windows Service.

To install Encryption Key Manager as a Windows service, issue:

```
LaunchEKMService -i KeyManagerConfig.properties
```

Once Encryption Key Manager is installed as a windows service with the above command, it can be started and stopped from the Control Panel.

If you are attempting to put your keystores on networked drives, you must change the user under which the Encryption Key Manager Windows Service runs to a network user. By default, the Encryption Key Manager Windows Service is created to run under the LocalSystem use, which has no access to the network. Do the following to make this change:

1. Log in as Administrator or a user that is a member of the Administrator's group.
2. Open **Services** located in the Administrative Tools.
3. Right click **EKM Service** and select **Properties**.
4. Click **Log On** tab.
5. Select **This account**.
6. Enter user name or browse for user.
7. Enter user passwords in Password and Confirm Password: text fields.
8. Click Apply to apply changes.
9. Click **OK** to close EKM Service properties.

The Encryption Key Manager Windows Service should now start successfully

The Command Line Interface Client

Once the Encryption Key Manager server is started, you can issue CLI commands through the client interface locally or remotely. To issue CLI commands you must first start the CLI client.

Authenticating CLI Client Users

The Server.authMechanism property in the configuration file specifies the authentication mechanism to be used with local/remote clients. When the value is set to `EKM`, the CLI client user must login to the server using user/password as EKMAdmin/changeME. (This password can be changed with **chgpasswd** command. See "chgpasswd" on page 162.) The default setting for the Server.authMechanism property is `EKM`.

When the Server.authMechanism property value is specified as `LocalOS` in the KeyManagerConfig.properties file, client authentication is done against the local operating system registry. The CLI client user must login to the server with OS user/password. Note that only user/password allowed to login and submit commands to the server is the user ID under which the server is running and which also has superuser/root authority.

Be sure to shut down the Encryption Key Manager server before making any change to the KeyManagerConfig.properties file.

For local OS-based authentication on Linux and Unix-based platforms, additional steps are required:

1. Download EKMServicesAndSamples.jar on the server machine from the Encryption Key Manager website (http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504) and extract the contents into a temporary directory.
2. Locate the LocalOS directory in the download.

3. Copy the libjaasauth.so file from the JVM-JaasSetup directory appropriate to your platform to *java_home*/jre/bin.

For z/OS, i5/OS, and Windows platforms this file is not necessary.

A readme file on the Encryption Key Manager website provides more installation details.

Starting the Command Line Interface Client

Note: The TransportListener.ssl.port properties in both the Encryption Key Manager Server and the Encryption Key Manager CLI client properties files must be set to the same value or they will not communicate. See "Debugging Communication Problems Between the CLI Client and the Encryption Key Manager Server" on page 173 if problems occur.

The Encryption Key Manager CLI Client and Encryption Key Manager Server use SSL to secure their communications. When using the default JSSE configuration of no client authentication, the certificates in the TransportListener.ssl.keystore on the Encryption Key Manager Server must be present in the TransportListener.ssl.truststore. In this way the client knows it can trust the server. If the Encryption Key Manager CLI client is running on the same system as the Encryption Key Manager Server, then the same configuration properties file can be used. This allows the Encryption Key Manager CLI Client to use the same keystore/truststore configuration as the Encryption Key Manager Server. If they are not on the same system or if you would like to have the client use different keystores, you must export the certificates from the TransportListener.ssl.keystore specified in the Encryption Key Manager Server configuration properties file. These certificates must be imported into the truststore specified by the TransportListener.ssl.truststore in the Encryption Key Manager CLI properties file.

You can start the CLI client and issue CLI commands in three ways. Regardless which you choose, you must specify the name of a CLI configuration file. See Appendix B for details.

Interactively

To run the commands interactively from any command window or shell, enter:

```
java com.ibm.keymanager.KMSAdminCmd CLIconfiglfile_name -i
```

The # prompt appears. Before submitting any commands, you must log in the CLI client into the key manager server with the following command:

```
#login –ekmuser EKMAdmin –ekmpassword changeME
```

Once the CLI client is successfully logged into the key manager server, you can execute any CLI commands. Use the **quit** or **logout** command to shut down the CLI client when you are finished. By default, the Encryption Key Manager server closes the communication socket with an unused client after ten minutes. Any attempt to enter a command after that will result in the client exiting. To specify a longer timeout period for the Encryption Key Manager server-client socket, modify theTransportListener.ssl.timeout property in the KeyManagerConfig.properties file.

Using a command file

To submit a batch of commands in a file to the key manager server, create a file containing the commands you wish to issue, for example, *clifile*. The first command in this file must be the **login** command because the client is required to login before executing any commands. For example, clifile might contain the following:

```
login -ekmuser EKMAdmin -ekmpassword changeME
listdrives
```

Then to execute this command file, start the CLI client:

```
java com.ibm.keymanager.admin.KMSAdminCmd CLIconfiglfile_name –filename clifile
```

One command at a time

You can run a single command at a time by specifying the CLI userid_ID and password for each command. From any command window or shell, enter:

```
java com.ibm.keymanager.KMSAdminCmd ClientConfig.properties_name -listdrives
    -ekmuser EKMAdmin -ekmpassword changeME
```

(This password can be changed with **chgpasswd** command.) The command will execute and the client session will end.

CLI Commands

The Encryption Key Manager provides a command set that can be used to interact with the Encryption Key Manager server from a command-line interface client, which includes the following commands.

addaliastogroup

Copy a specific alias from an existing (source) key group to a new (target) key group. This is useful when you wish to add an alias that already exists in one key group to a different key group.

addaliastogroup -aliasID *aliasname* **-sourceGroupID** *groupname* **-targetGroupID** *groupname*

-aliasID
The *aliasname* for the key to be added.

-sourceGroupID
The unique *groupname* used to identify the group from which the alias is to be copied.

-targetGroupID
The unique *groupname* used to identify the group to which the alias is to be added.

Example: addaliastogroup -aliasID aliasname -sourceGroupID keygroup1 -targetGroupID keygroup2

adddrive

Add a new drive to key manager drive table. Refer to "Automatically Update Tape Drive Table" on page 135 to learn how to add tape drives to the drive table automatically. See "Encryption Keys and the TS1120 and TS1130 Tape Drives " on page 29 and "Encryption Keys and the LTO Ultrium 4 Tape Drive " on page 32 for information about alias requirements.

adddrive -drivename *drivename* [**-rec1** *alias*] [**-rec2** *alias*][**-symrec** *alias*]

-drivename
drivename specifies the 12-digit serial number of the drive to be added.

-rec1
Specifies the *alias* (or key label) of the drive's certificate.

-rec2
Specifies a second *alias* (or key label) of the drive's certificate.

-symrec
Specifies an *alias* (of the symmetric key) or a key group name for the tape drive.

Example: adddrive -drivename 000123456789 -rec1 alias1 -rec2 alias2

addkeygroup

Create an instance of a key group with a unique Group ID in the Key Group XML.

addkeygroup -groupID *groupname*

-groupID
The unique *groupname* used to identify the group in the KeyGroup XML file.

Example: addkeygroup -groupID keygroup1

addkeygroupalias

Create a new alias for an existing key alias in your keystore for addition to a specific key group ID.

addkeygroupalias -alias *aliasname* **-groupID** *groupname*

-alias
The new *aliasname* for the key.

-groupID
The unique *groupname* used to identify the group in the KeyGroup XML file.

Example: addkeygroupalias -alias aliasname -groupID keygroup1

chgpasswd

Change the CLI client's user (EKMAdmin) default password.

chgpasswd -new *password*

-new
The new *password* that replaces the previous password.

Example: chgpasswd -new ebw74jxr

createkeygroup

Create the initial key group object in the KeyGroups.xml file. Run only once.

createkeygroup -password *password*

-password
The *password* that is used to encrypt the keystore's password in the

KeyGroups.xml file for later retrieval. The keystore encrypts the key group's key, which in turn encrypts each individual key group alias password. Therefore no key in the KeyGroups.xml file is in the clear.

Example: createkeygroup -password password

deletedrive

Delete a drive from key manager drive table. Equivalent commands are **deldrive** and **removedrive**.

deletedrive -drivename *drivename*

-drivename
drivename specifies the serial number of the drive to be deleted.

Example: deletedrive -drivename 000123456789

delgroupalias

Delete a key alias from a key group.

delgroupalias -groupID *groupname* **-alias** *aliasname*

-groupID
The unique *groupname* used to identify the group in the KeyGroups.xml file.

-alias
The *aliasname* for the key alias to be removed.

Example: delgroupalias -groupID keygroup1 -alias aliasname

delkeygroup

Delete an entire key group.

delkeygroup -groupID *groupname*

-groupID
The unique *groupname* used to identify the group in the KeyGroups.xml file.

Example: delkeygroup -groupID keygroup1

exit

Exit CLI client and stop Encryption Key Manager server. Equivalent command is **quit**.

Example: exit

export

Export a drive table or Encryption Key Manager server configuration file to the specified URL.

export {**-drivetab** | **-config**} **-url** *urlname*

-drivetab
Export the drive table.

-config
Export the Encryption Key Manager server configuration file.

-url
urlname specifies the location where the file is to be written.

Example: export -drivetab -url FILE:///keymanager/data/export.table

help

Display command line interface command names and syntax. Equivalent command is **?**.

help

import

Import a drive table or configuration file from a specified URL.

import {**-merge**|**-rewrite**} {**-drivetab**|**-config**} **-url** *urlname*

-merge
Merge the new data with current data.

-rewrite
Replace the current data with new data.

-drivetab
Import the drive table.

-config
Import the configuration file.

-url
urlname specifies the location from which the new data is to be taken.

Example: import -merge -drivetab -url FILE:///keymanager/data/export.table

list

List certificates contained in keystore named by config.keystore.file property.

list [-cert |-key|-keysym][-alias *alias* **-verbose |-v]**

-cert
List certificates in the specified keystore.

-key
List all keys in the specified keystore.

-keysym
List symmetric keys in the specified keystore.

-alias
alias specifies a specific certificate to list.

-verbose|-v
Display more information about the certificate(s).

Examples:

`list -v` lists everything in the keystore.

`list -alias mycert -v` lists all available data for the mycert alias if it exists in the config.keystore.file keystore.

listcerts

List certificates contained in keystore named by config.keystore.file property.

listcerts [**-alias** *alias* **-verbose** |**-v**]

-alias
alias specifies a specific certificate to list.

-verbose|-v
Display more information about the certificate(s).

Example: listcerts -alias alias1 -v

listconfig

Lists the Encryption Key Manager server configuration properties in memory, reflecting the current contents of the KeyManagerConfig.properties file plus any updates made with the **modconfig** command.

listconfig

listdrives

List drives in drive table.

listdrives [**-drivename** *drivename*]

-drivename
drivename specifies the serial number of the tape drive to list.

-verbose|-v
Display more information about the tape drive(s).

Example: listdrives -drivename 000123456789

login

Sign on to a CLI client on the Encryption Key Manager server.

login -ekmuser *userID* **-ekmpassword** *password*

-ekmuser
Specify `EKMadmin` or a localOS user ID value for *userID*, depending on the type of authentication used (see "Authenticating CLI Client Users" on page 159).

-ekmpassword
Valid password for user ID.

Example: login -ekmuser EKMAdmin -ekmpassword changeME

logout

Logs off the current user. Equivalent command is **logoff**. These commands are only useful when the client session is enabled.

Example: logout

modconfig

Modify a property in the Encryption Key Manager server configuration properties file, KeyManagerConfig.properties. Equivalent command is **modifyconfig**.

modconfig {**-set** | **-unset**} **-property** *name* **-value** *value*

-set
: Set the specified property to the specified value.

-unset
: Remove the specified property.

-property
: *name* specifies the name of the target property.

-value
: *value* specifies the new value for the target property when **-set** is specified.

Example: modconfig -set -property sync.timeinhours -value 24

moddrive

Modify drive information in the drive table. Equivalent command is **modifydrive**.

moddrive -drivename *drivename* {**-rec1** [*alias*] | **-rec2** [*alias*]| **-symrec** [*alias*]}

-drivename
: *drivename* specifies the serial number of the tape drive.

-rec1
: Specifies the *alias* (or key label) of the drive's certificate.

-rec2
: Specifies a second *alias* (or key label) of the drive's certificate.

-symrec
: Specifies an *alias* (of the symmetric key) or a key group name for the tape drive.

Example: moddrive -drivename 000123456789 -rec1 newalias1

refresh

Tells the Encryption Key Manager to refresh the debug, audit, and drive table values with the latest configuration parameters.

Example: refresh

refreshks

Refreshes the keystore. Use this to reload the keystore specified in **config.keystore.file** if it was modified while the Encryption Key Manager server was running. Use this command only when needed as it may degrade performance.

Example: refreshks

status

Displays whether key manager server is started or stopped.

Example: status

stopekm

Stops the Encryption Key Manager server.

Example: stopekm

sync

Synchronizes the configuration file properties, or drive table information, or both on another Encryption Key Manager server with those on the key manager server issuing the command.

Notes

Neither synchronization method acts on the keystore or KeyGroups.xml file. These must be copied manually.

If your environment uses Shared HFS function for Unix Systems Services AND you use shared directories for debug and error logs, use of the **sync -all** or **sync -config** command is not recommended as this will change the log settings on synchronized systems to use the same directories. Only the **sync -drivetable** command should be used for this type of configuration.

sync {**-all** | **-config** | **-drivetab**} **-ipaddr** *ip_addr* :*ssl*:*port* [**<u>-merge</u>** | **-rewrite**]

-all
: Send both the configuration properties file and the drive table information to the Encryption Key Manager server specified by **-ipaddr**.

-config
: Send only the configuration properties file to the Encryption Key Manager server specified by **-ipaddr**.

-drivetab
: Send only the drive table information to the Encryption Key Manager server specified by **-ipaddr**.

-ipaddr
: *ip_addr:ssl:port* specifies the address and ssl port of the receiving Encryption Key Manager server. The *ssl:port* should match the value specified for "TransportListener.ssl.port" in the KeyManagerConfig.properties file of the receiving server.

-merge
Merge new drive table data with current data. (The configuration file is always a rewrite.) This is the default.

-rewrite
Replace the current data with new data.

Example: sync -drivetab -ipaddr remoteekm.ibm.com:443 -merge

version

Displays the version of the Encryption Key Manager server.

Example: version

Chapter 7. Problem Determination

You can enable debugging for an individual component, multiple components, or all components of the Encryption Key Manager.

When Running on z/OS

When running the Encryption Key Manager on z/OS, there are three places to look for errors:

The Encryption Key Manager audit log
Most error messages will appear in the audit log. The location and filename are set in the **Encryption Key Manager KeyManagerConfig.properties** file in the Audit.handler.file.directory and Audit.handler.file.name properties.

Standard Error (stderr)
When running the Encryption Key Manager as a started task using the Encryption Key Manager console wrappers, examine the Execution LOG for errors. When running the Encryption Key Manager in the foreground using USS/OMVS, these errors will appear where you have directed STDERR which may simply be the screen.

The Encryption Key Manager debug log
The location is set in the **Encryption Key Manager KeyManagerConfig.properties file** debug.output.file property and the data written to the file is controlled by the debug property. For space reasons, it is recommended that you initially set the property to debug = none. If an error is encountered while Encryption Key Manager is running you can turn debug on by submitting the **modconfig –set –property debug –value all** command. If you have run into a problem and did not get any debug information from the Encryption Key Manager audit log or Standard Error, it is recommended that you set debug=all.

The following is a list of errors, and their possible causes, you may see when running the Encryption Key Manager on z/OS:

Error 1

```
com.ibm.keymanager.j [Caused by java.security.PrivilegedActionException:
java.io.IOException: The private key of EKMSERVE is not a software or icsf
key. Error creating key entry because private key is not available.]
```

Possible causes: If you are using a RACF keystore type (i.e, JCE4758RACFKS or JCERACFKS):

- This error can occur if the Userid running the Encryption Key Manager is not the owner of the KeyRing/Private key. RACF only allows a private key to be retrieved by its owner.
- This error can occur when starting the Encryption Key Manager if your keyring has a public key (that does not contain a corresponding private key, such as a business partners key) AND that key was not connected as CERTAUTH (see directions in “Business Partner and Remote z/OS Systems ” on page 60).

Error 2

```
Runtime event:[
  timestamp=Wed Sep 06 13:30:54 EDT 2006
  event source=com.ibm.keymanager.g.fb
  outcome=[result=unsuccessful]
  event type=SECURITY_RUNTIME
  message= ***Error: Information not available for protected private keys..
     ErrorCode=0xEE0F
  resource=[name= Drive Serial Number: 000001350808 WWN: 500507630F04BC1B
     Key Alias/Label[0]: Tape_Sol_Tst_Shr_Pvt_1024_Lbl_02;type=file]
  action=stop
```

Possible cause: This error could occur if unrestricted policy files were not installed. Refer to "Copy the Unrestricted Policy Files" on page 46. This error usually appears in the Encryption Key Manager audit log.

| **Note:** This error may also occur with an EE31 error code and the same text. It can
| be resolved by installing the unrestricted policy files.

Error 3

```
# java.lang.NoClassDefFoundError: javax/crypto/b
        at javax.crypto.Cipher.a(Unknown Source)
        at javax.crypto.Cipher.getInstance(Unknown Source)
        at com.ibm.keymanager.g.b.a(b.java:189)
        at com.ibm.keymanager.g.fb.a(fb.java:937)
        at com.ibm.keymanager.g.fb.run(fb.java:1277)
```

Possible cause: The wrong version, or a corrupt copy, of unrestricted policy files may be installed. Note that this error is sent to STDERR (your job execution log) and not the Encryption Key Manager audit log.

Error 4

```
***Error: no such provider: IBMJCE4758. ErrorCode=0xEE0F
Runtime event:[
  timestamp=Mon Sep 18 22:43:26 EDT 2006
  event source=com.ibm.keymanager.logic.MessageProcessor
  outcome=[result=unsuccessful]
  event type=SECURITY_RUNTIME
  message= ***Error: no such provider: IBMJCE4758. ErrorCode=0xEE0F
  resource=[name= Drive Serial Number: 000001350699 WWN: 500507630F0C851C;type=file]
  action=stop
  ]
```

Possible cause: The Java hardware provider has not been added to the java.security provider list. This must be done each time there is a new Java installation/upgrade if you are planning to use ICSF hardware keys. Refer to "Add the Java Hardware Provider (Only if Using ICSF)" on page 47.

Error 5

```
java.security.PrivilegedActionException: java.io.IOException: R_datalib
(IRRSDL00) error: error while getting certificate or trust info (8, 8, 80)
```

Possible cause: Quotation marks surround the keyring name specified in the KeyManagerConfig.properties file (for example, `config.keystore.file = safkeyring:"//EKMSERV/EKMRing"`). Remove the quotation marks.

Error 6

```
java.security.PrivilegedActionException: java.io.IOException: Failed validating certificate paths
        at java.security.AccessController.doPrivileged1(Native Method)
        at java.security.AccessController.doPrivileged(AccessController.java:351)
        at com.ibm.keymanager.b.a.a(a.java:23)
        at com.ibm.keymanager.b.a.a(a.java:148)
        at com.ibm.keymanager.b.a.b(a.java:138)
        at com.ibm.keymanager.i.a.a.h(a.java:711)
        at com.ibm.keymanager.i.a.a.c(a.java:595)
        at com.ibm.keymanager.KMSAdminCmd.main(KMSAdminCmd.java:2)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:85)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:58)
        at
```

Possible cause: A CA Certificate is not connected to the KeyRing (note: At least one CERTAUTH cert is required, even if all certificates are self-signed). This message may only appear in the debug log and occur when attempting to start the Encryption Key Manager Server.

Error 7

java.lang.NoClassDefFoundError

```
java.lang.NoClassDefFoundError: com/ibm/keymanager/logic/EncryptionCBDQuery
:at com.ibm.keymanager.logic.RequestEEDKs.createMsg(RequestEEDKs.java:48)
:at com.ibm.keymanager.logic.RequestEEDKs.<init>(RequestEEDKs.java:39)
:at com.ibm.keymanager.logic.MessageProcessor.ProcessMessage(MessageProcessor.java:351)
:at com.ibm.keymanager.logic.MessageProcessor.run(MessageProcessor.java(Compiled Code))
```

Possible cause: Java is not available. Possibly the file system where java is installed has been dismounted. If using in-band key management, you may also see this IOS error:

```
IOS628E ENCRYPTION ON DEVICE E0A4 HAS FAILED DUE TO COMMUNICATION TIME OUT
IOS000I E0A4,D6,IOE,01,0E00,,**,A0209A,ITSXZ071 948
804C08C022402751 0301FF0000000000 0000000000000092 2004E82062612111
ENCRYPTION FAILURE
CU = 03 DRIVE = 000000 EKM = 000000
```

Error 8

```
JVMJZBL2999T JvmExitHook entered with exitCode=-3,
javaMainReturnedOrThrewExcep
JVMJZBL1043N The Java virtual machine completed with
System.exit(-3)
```

or

```
EKM server is now terminating abnormally with a return code of 4093.
```

Possible cause: The Java version (or just the Encryption Key Manager JAR version) was replaced such that the Encryption Key Manager was upgraded from a build earlier than 20070503 to a build equal to or later than 20070503 ("Determining Which Version of Encryption Key Manager is Installed" on page 131). If that is the case, you must define the Audit.metadata.file.name property in the KeyManagerConfig.properties file. This is the name of the XML file where metadata is saved. This property is required to start versions of Encryption Key Manager with build date 20070503 (when metadata support was added) and later. See Chapter 9, "Using Metadata," on page 199. Check your current Encryption Key Manager version Before you decide to upgrade to the latest Encryption Key Manager.

Check These Important Files for Encryption Key Manager Server Problems

When the Encryption Key Manager fails to start there are three files to check to determine the cause of the problem.

- **native_stdout.log** and **native_stderr.log**
  - Since the Encryption Key Manager Server runs in a background process, it has no console to display its normal informational and error messages. Those messages are logged to these two files.
  - If the Encryption Key Manager Server properties file contains the property **debug.output.file**, then these two files are created in the same directory as the debug log.
  - If the Encryption Key Manager Server properties file does not contain the property **debug.output.file**, then these two files are created in the working directory.
  - These two files are deleted and recreated on every start of the Encryption Key Manager Server.
- **Audit log**
  - Audit log contains records that were logged as the Encryption Key Manager is processing.
  - The location of this file is specified by two properties in **KeyManagerConfig.properties**, the Encryption Key Manager Server configuration properties file:
    - Audit.handler.file.directory – specifies which directory the audit log should be located
    - Audit.handler.file.name – specifies the filename of the audit log.
  - For more information on Audit, see Chapter 8, “Audit Records,” on page 191.

Log Entries for Keystore Passwords Greater than 127 Characters

When the Encryption Key Manager is installed as a Windows Service and the keystore passwords in the KeyManagerConfig.properties file are 128 characters in length or greater, the Encryption Key Manager will fail to start because it has no way to prompt for a password of acceptable length. The native Encryption Key Manager logs will contain entries similar to the following:

native_stdout.log

```
Server initialized
Default keystore failed to load
```

native_stderr.log

```
at com.ibm.keymanager.KeyManagerException: Default keystore failed to load
at com.ibm.keymanager.keygroups.KeyGroupManager.loadDefaultKeyStore(KeyGroupManager.java:145)
at com.ibm.keymanager.keygroups.KeyGroupManager.init(KeyGroupManager.java:605)
at com.ibm.keymanager.EKMServer.c(EKMServer.java:243)
at com.ibm.keymanager.EKMServer.<init>(EKMServer.java:753)
at com.ibm.keymanager.EKMServer.a(EKMServer.java:716)
at com.ibm.keymanager.EKMServer.main(EKMServer.java:129)
```

Debugging Communication Problems Between the CLI Client and the Encryption Key Manager Server

Communication between the Encryption Key Manager CLI client and the Encryption Key Manager Server is done over the ports specified in the TransportListener.ssl.port property in both the server and client configuration properties files and is protected by SSL.

The following is a list of possible reasons why the client may not connect to the Encryption Key Manager Server. It includes steps showing how to determine the problem and correct it.

- The Encryption Key Manager Server is not running, therefore the client has nothing to communicate with.
  1. Issue **netstat –an** from a command window and confirm that the ports specified by the TransportListener.ssl.port and TransportListener.tcp.port properties in the Encryption Key Manager Server properties file are displayed. If the ports are not displayed, then the server is not running
- The TransportListener.ssl.host property in the Encryption Key Manager CLI client properties file does not point to the correct host where the Encryption Key Manager Server is running.
  1. The value of the TransportListener.ssl.host property in the Encryption Key Manager CLI client properties file defaults to `localhost`. Modify the value of this property to point to the correct host.
- The Encryption Key Manager Server and the Encryption Key Manager CLI client are not talking on the same port.
  1. Check the TransportListener.ssl.port properties in both the Encryption Key Manager Server and the CLI client properties files to confirm they are set to the same value.
- The Encryption Key Manager Server and the CLI client cannot find a common certificate to use to secure communications.
  1. Ensure the keystores specified in the TransportListener.ssl.keystore and TransportListener.ssl.truststore CLI client properties contain the same certificates as the Admin.ssl.keystore and Admin.ssl.truststore keystores in the server properties.
  2. Ensure the TransportListener.ssl.keystore.password in the client properties has the correct password.
  3. Ensure none of the certificates in these keystores have expired. JSSE will not use expired certificates to secure communications.
- The Encryption Key Manager CLI client properties file is read-only.
  1. Check the attributes or the permissions on the file to ensure the user running the Encryption Key Manager CLI client has permission to access and modify the file.
- The Encryption Key Manager Server properties file has Server.authMechanism = LocalOS but the required file from the EKMServicesAndSamples package has not been installed or was installed in the wrong location.
  1. See the readme included with the EKMServiceAndSamples package for more information about authentication.

Debugging Key Manager Server Problems

Most problems concerning the key manager involve configuration or starting the key manager server. Refer to Appendix B, Default Configuration File, for information on specifying the debug property.

If the Encryption Key Manager fails to start, check for a firewall.

Either a software firewall or a hardware firewall may be blocking the Encryption Key Manager from accessing the port.

EKM server not started. EKM.properties config could not be loaded or found.

1. This error occurs when starting the KMSAdminCmd or EKMLaunch without specifying the complete path of **KeyManagerConfig.properties** when the properties file is not located in the default path.

   Default path on i5/OS is **/QIBM/ProdData/OS400/Java400/ext/**

   Default path on Windows is **C:/Program Files/IBM/KeyManagerServer/**

   Default path on Unix® type platforms is **/opt/ibm/KeyManagerServer/**
2. Re-enter the command to start the KMSAdminCmd and include the complete path of the **KeyManagerConfig.properties** file. See Appendix B, "Encryption Key Manager Configuration Properties Files" for more information.

EKM server is not started. File name for XML metadata file needs to be specified in the configuration file.

The Audit.metadata.file.name entry is missing from the configuration file.

To correct this problem, add the Audit.metadata.file.name property to the **KeyManagerConfig.properties** configuration file.

Failed to start EKM.Mykeys. The system cannot find the specified file.

1. This error message occurs when the keystore entries in **KeyManagerConfig.properties** do not point to an existing file.
2. To correct this problem, ensure the following entries in the **KeyManagerConfig.properties** file point to existing, valid keystore files:

   Admin.ssl.keystore.name

   TransportListener.ssl.truststore.name

   TransportListener.ssl.keystore.name

   Admin.ssl.truststore.name

   See Appendix B, "Encryption Key Manager Configuration Properties Files" for more information.

Failed to start EKM. File does not exist = safkeyring://xxx/yyy

The error can be caused by specifying the wrong provider in the IJO variable in the Encryption Key Manager environment shell script.

For JCECCARACFKS keystores use:

```
-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider
```

and for JCERACFKS keystores use:

```
-Djava.protocol.handler.pkgs=com.ibm.crypto.provider
```

Failed to start EKM. keystore was tampered with, or password was incorrect.

1. This error occurs if one or more of these entries in the properties file (see Appendix B, "Encryption Key Manager Configuration Properties Files") has the wrong value:

   config.keystore.password (corresponds to config.keystore.file)

   admin.keystore.password (corresponds to admin.keystore.name)

   transportListener.keystore.password (corresponds to transportListener.keystore.name)
2. This error could also occur if the wrong password is entered at the password prompt on start up of the server.
3. If none of the passwords are in the configuration, you are prompted up to three times if all 3 keystores entries in the properties file are unique. If all of the entries in the properties are the same, then you are prompted once.

Failed to start EKM. PIN is incorrect.

This error only occurs when a hardware keystore is used. Possible causes are:
1. config.keystore.password is not correct for accessing the hardware keystore.
2. The slot number to access the hardware keystore is incorrect.

   The **config.keystore.file** should be the pkcs11 configuration file for your hardware device, for example, **/home/user/eracom.cfg**.

Failed to start EKM. Invalid keystore format.

1. This error may occur when the wrong keystore type is specified for one of the keystore entries in the properties file.
2. If all of the keystore entries in the properties file point to the same file, the Encryption Key Manager will use the config.keystore.type value as the keystore type for all keystores.
3. When there is no type entry in the properties file for a particular keystore, the Encryption Key Manager assumes the type is `jceks`.

Failed to start the server. Listener thread is not up and running.

This error may occur for a number of reasons:
1. The following two entries in the **KeyManagerConfig.properties** file point to the same port:

   TransportListener.ssl.port

   TransportListener.tcp.port

   Each of the transport listeners must be configured to listen on its own port.
2. One or both of those entries is configured to a port that is already in use by another service running on the same machine as the Key Manager server. Find ports that are not in use by another service and use those to configure the Key Manager server.
3. On systems running Unix type operating systems, this error may occur if one or both of the ports are lower than 1024 and the user starting the Key Manager server is not root. Modify the transport listener entries in the **KeyManagerConfig.properties** to use ports above 1024.

“[Fatal Error] :-1:-1: Premature end of file.” message in native_stderr.log.

This message occurs when the Encryption Key Manager loads an empty keygroups file. This message is from the XML parser and does not keep the Encryption Key Manager from starting unless it is configured to use keygroups and the file specified by the config.keygroup.xml.file property in **KeyManagerConfig.properties**, the Encryption Key Manager Server properties file, is corrupted.

Error: Unable to find Secretkey in the config keystore with alias:MyKey.

The symmetricKeySet entry in properties file is contains a key alias that does not exist in the config.keystore.file

To correct this problem, modify the symmetricKeySet entry in the configuration file to only contain aliases that exist in the keystore file designated by the config.keystore.file entry in **KeyManagerConfig.properties** OR add the missing symmetric key to the keystore. See Appendix B, “Encryption Key Manager Configuration Properties Files” for more information.

The symmetric key algorithm must be DESede if the zOSCompatibility flag in the configuration file is set to true.

The zOSCompatibility setting in the **KeyManagerConfig.properties** file is set to true with symmetricKeySet pointing to an alias of a key that is AES

To correct this problem, set the zOSCompatibility entry in the configuration file to `false` OR set the value of symmetricKeySet to specify DESede keys.

The symmetric key algorithm must be AES-256 if the zOSCompatibility flag in the configuration file is set to false.

The zOSCompatibility setting in the **KeyManagerConfig.properties** file is set to false with symmetricKeySet pointing to an alias of a key that is DESede

To correct this problem, set the zOSCompatibility entry in the configuration file to `true` OR set the value of symmetricKeySet to specify AES keys.

No symmetric keys in symmetricKeySet, LTO drives cannot be supported.

This is an information message. The Encryption Key Manager server will still start, but LTO drives cannot be supported on this instance of Encryption Key Manager. This is not a problem if there are no LTO drives configured to communicate with this Encryption Key Manager.

Encryption Key Manager-Reported Errors

This section defines error messages that are reported by the Encryption Key Manager and returned in the drive sense data. They are typically called fault symptom codes or FSCs. The table includes the error number, a short description of the failure, and corrective actions. Refer to Appendix B, Default Configuration File, for information on specifying the debug property.

Table 26. Errors that are reported by the encryption key manager

Error Number	Description	Action
EE02	Encryption Read Message Failure: DriverErrorNotifyParameterError: ″Bad ASC & ASCQ received. ASC & ASCQ does not match with either of Key Creation/Key Translation/Key Aquisition operation.″	The tape drive asked for an unsupported action. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug tracing on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EE0F	Encryption logic error: Internal error: ″Unexpected error. Internal programming error in EKM.″	Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug tracing on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
	Error: Hardware error from call CSNDDSV returnCode 12 reasonCode 0.	If using hardware cryptography, ensure that ICSF is started.
EE23	Encryption Read Message Failure: Internal error: ″Unexpected error........″	The message received from the drive or proxy server could not be parsed because of general error. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Enable debug on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.

Table 26. Errors that are reported by the encryption key manager (continued)

Error Number	Description	Action
EE25	Encryption Configuration Problem: Errors that are related to the drive table occurred.	Ensure that the config.drivetable.file.url is correct in the KeyManagerConfig.properties file, if that parameter is supplied. Run the `listdrives -drivename <drivename>` command on the Encryption Key Manager server to verify whether the drive is correctly configured (for example, the drive serial number, alias, and certificates are correct). Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, "Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK," on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug tracing and retry the operation. If the problem persists, contact IBM for support. See "Whom Do I Contact for IBM Support?" on page 181.
EE29	Encryption Read Message Failure: Invalid signature	The message received from the drive or proxy server does not match the signature on it. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, "Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK," on page 131 to determine the latest version). Enable debug on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See "Whom Do I Contact for IBM Support?" on page 181.
EE2B	Encryption Read Message Failure: Internal error: "Either no signature in DSK or signature in DSK can not be verified."	Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, "Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK," on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug tracing on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See "Whom Do I Contact for IBM Support?" on page 181.

Table 26. Errors that are reported by the encryption key manager (continued)

Error Number	Description	Action
EE2C	Encryption Read Message Failure: QueryDSKParameterError: ″Error parsing a QueryDSKMessage from a device. Unexpected dsk count or unexpected payload.″	The tape drive asked the Encryption Key Manager to do an unsupported function. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug tracing on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EE2D	Encryption Read Message Failure: Invalid Message Type	The Encryption Key Manager received a message out of sequence or received a message that it does not know how to handle. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Enable debug on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EE2E	Encryption Read Message Failure: Internal error: Invalid signature type	The message received from the drive or proxy server does not have a valid signature type. Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Enable debug on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EE30	Prohibited request.	An unsupported operation has been requested for a tape drive. Enter the correct, supported command for the target tape drive.

Table 26. Errors that are reported by the encryption key manager (continued)

Error Number	Description	Action
EE31	Encryption Configuration Problem: Errors that are related to the keystore occurred.	Check the key labels that you are trying to use or configured for the defaults. You can list the certificates that are available to the Encryption Key Manager by using the listcerts command. If you know that you are trying to use the defaults, then run the listdrives -drivename *drivename* command on the Encryption Key Manager server to verify whether the drive is correctly configured (for example, the drive serial number, and associated aliases/key labels are correct). If the drive in question has no aliases/key labels associated with it, then check the values of default.drive.alias1 and default.drive.alias2. If this does not help or the alias/key label exists, then collect debug logs and contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EE32	Keystore-related problem.	Most likely cause is either that tape was encrypted using a different Encryption Key Manager with different keys or the key that was used to encrypt this tape has been renamed or deleted from the keystore. Issue `list -keysym` and ensure the request alias is in the keystore.
EEE1	Encryption logic error: Internal error: ″Unexpected error: EK/EEDK flags conflict with subpage.″	Ensure that you are running the latest version of the Encryption Key Manager (refer to Chapter 4, “Upgrading to the Latest Version of Encryption Key Manager or IBM Java SDK,” on page 131 to determine the latest version). Check the versions of drive or proxy server firmware and update them to the latest release, if needed. Enable debug on the key manager server. Try to recreate the problem and gather debug logs. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.
EF01	Encryption Configuration Problem: ″Drive not configured.″	The drive that is trying to communicate with the Encryption Key Manager is not present in the drive table. Ensure that the config.drivetable.file.url is correct in the KeyManagerConfig.properties file, if that parameter is supplied. Run the `listdrives` command to check whether the drive is in the list. If not, configure the drive manually by using the `adddrive` command with the correct drive information or set the ″drive.acceptUnknownDrives″ property to true using the `modconfig` command. Enable debug tracing and retry the operation. If the problem persists, contact IBM for support. See “Whom Do I Contact for IBM Support?” on page 181.

Whom Do I Contact for IBM Support?

The entitlement for software support varies depending on the operating system on which Encryption Key Manager is running, and depending on whether the support requirement is defect-related or implementation-related.

Table 27. IBM Support Contacts

Type of Support	IBM Operating Systems:zOS, AIX, i5OS Linux		Non-IBM Operating Systems: Windows, Solaris, HP/UX
Defect Support	Contact IBM Service with IBM operating system's name or identifier and customer number.	Contact IBM Service with IBM Tape Library's machine type/model and serial number.	Contact IBM Service with TPC/BE name or identifier and customer number.
Implementation Support [1]	Contact SupportLine IBM Service.	Contact SupportLine IBM Service.	Contact SupportLine IBM Service.
[1] An IBM Supportline contract offers the best Encryption Key Manager implementation assistance. Some basic implementation assistance can be obtained by contacting IBM Service. Use the same machine type-model that would be used to report a defect. Should your customer require more extensive implementation assistance, billable onsite services are available from IGS and Lab Services. Contact IGS Inside Sales (888-426-4343 option 3) to obtain a Statement of Work (SOW).			

If there is a defect, IBM Service is always the first point of contact. The method to engage IBM Software Service varies depending on the operating system on which Encryption Key Manager is being run.

For the following IBM operating systems: z/OS, AIX, and i5/OS, contact IBM Service (For US Customers call 800-IBM-SERV). Select the software option, then identify the operating system and the same customer number that was used to order the operating system.

For Linux, select the hardware option, and use the Machtype-Model of the tape library to report the defect.

For the non-IBM operating systems; Windows, Solaris, and HP/UX, select the software option, then identify the software as TPC/BE and supply the same customer number that was used to order TPC/BE.

Note: The relevant operating system here is the operating system on which Encryption Key Manager is running, not the operating system that is generating the encrypted IOs.

Messages

The following messages can be generated by the Encryption Key Manager and displayed on the admin console.

Config File not Specified

Text

```
Configuration file not specified:  KeyManager Configuration file not
specified when starting EKM.
```

Explanation

The KMSAdmin command requires that the configuration file be passed in as a command-line parameter.

System Response

The program stops.

Operator Response

Supply the configuration file and retry the command.

Failed to Add Drive

Text

```
Failed to add drive. Drive already exists.
```

Explanation

The **adddrive** command failed because the drive is already configured with the Encryption Key Manager and exists in the drive table.

Operator Response

Run the **listdrives** command to see if the drive is already configured with Encryption Key Manager. If the drive already exists, the drive configuration can be changed using **moddrive** command. Run **help** for more information.

Failed to Archive the Log File

Text

```
Failed to archive the log file.
```

Explanation

The log file cannot be renamed.

Operator Response

Check file permissions and space on that drive.

Failed to Delete the Configuration

Text

```
"modconfig" command failed.
```

Explanation

Failed to delete the Encryption Key Manager configuration through **modconfig** command.

Operator Response

Check the command syntax using **help** make sure parameters supplied are correct. Please check the audit logs for more information.

Failed to Delete the Drive Entry

Text

```
"deldrive" command failed.
```

Explanation

deldrive command failed to delete the drive entry from the drive table.

Operator Response

Check the command syntax using **help** and make sure parameters supplied are correct. Make sure the drive is configured with the Encryption Key Manager using **listdrives** command. Please check the audit logs for more information.

Failed to Import

Text

```
"import" command failed.
```

Explanation

Drive table or configuration files cannot be imported.

System Response

The Encryption Key Manager server does not start.

Operator Response

Make sure the specified URL exists and has read permissions. Check the command syntax using **help**. Make sure the parameters are correct and retry.

Failed to Modify the Configuration

Text

```
"modconfig" command failed.
```

Explanation

Failed to modify the Encryption Key Manager configuration through **modconfig** command.

Operator Response

Check the command syntax using **help** make sure parameters supplied are correct. Please check the audit logs for more information.

File Name Cannot be Null

Text

```
File name was not supplied for audit log file.
```

Explanation

Audit file name is not supplied through configuration properties for the Encryption Key Manager. This parameter is a required configuration parameter.

System Response

The program stops.

Operator Response

Check that the property `Audit.handler.file.name` is defined in the configuration properties file supplied to Encryption Key Manager and try restarting it.

File size Limit Cannot be a Negative Number

Text

```
Maximum file size for audit log can not be a negative number.
```

Explanation

`Audit.handler.file.size` property value in the Encryption Key Manager configuration file must be a positive number.

System Response

The Encryption Key Manager does not start.

Operator Response

Please specify a valid number for `Audit.handler.file.size` and try restarting the Encryption Key Manager.

No Data to be Synchronized

Text

```
No data can be found to be synchronized with "sync".
```

Explanation

The sync command cannot identify any data to be synchronized.

Operator Response

Check the configuration file supplied exists and if drive table is correctly configured in the configuration file using *config.drivetable.file.url*. Check the syntax using **help** and retry the **sync** command.

Invalid Input

Text

```
Invalid input parameters for the CLI.
```

Explanation

The particular command syntax may not be correct.

Operator Response

Make sure the command entered is correct. Check the command syntax using **help**. Make sure parameters supplied are correct and retry.

Invalid SSL Port Number in Configuration File

Text

```
Invalid SSL port number specified in the EKM configuration file.
```

Explanation

SSL port number supplied in the configuration file is not a valid number.

System Response

The Encryption Key Manager does not start.

Operator Response

Specify valid port number for the `TransportListener.ssl.port` property in the configuration file when starting the Encryption Key Manager and try to restart.

Invalid TCP Port Number in Configuration File

Text

```
Invalid TCP port number specified in the EKM configuration file.
```

Explanation

TCP port number supplied in the configuration file is not a valid number.

System Response

The Encryption Key Manager does not start.

Operator Response

Specify valid port number for the `TransportListener.tcp.port` property in the configuration file when starting the Encryption Key Manager and try to restart. The default TCP port number is 3801.

Must Specify SSL Port Number in Configuration File

Text

```
SSL port number is not configured in the properties file.
```

Explanation

SSL port number is a required property to be configured in configuration properties file. It is used for communication between Encryption Key Manager servers in a multi-server environment.

System Response

The Encryption Key Manager does not start.

Operator Response

Specify valid port number for the `TransportListener.ssl.port` property and try to restart the Encryption Key Manager.

Must Specify TCP Port Number in Configuration File

Text

```
TCP port number is not configured in the properties file.
```

Explanation

TCP port number is a required property to be configured in configuration properties file. It is used for communication between the drive and the Encryption Key Manager.

System Response

The Encryption Key Manager does not start.

Operator Response

Specify valid port number for the `TransportListener.tcp.port` property and try to restart the Encryption Key Manager. The default TCP port number is 3801.

Server Failed to Start

Text

```
EKM server failed to start.
```

Explanation

The Encryption Key Manager server cannot start because of configuration problems.

Operator Response

Check the parameters in the configuration file supplied. Please check the logs for more information.

Sync Failed

Text

```
"sync" command failed.
```

Explanation

Sync operation to synchronize the data between two Encryption Key Manager servers failed.

Operator Response

Make sure IP address specified for remote Encryption Key Manager server is correct and that computer is accessible. Make sure configuration file exists and contains correct drive table information. Check the **sync** command syntax using **help**. Check the logs for more information.

The Specified Audit Log File is Read Only

Text

```
The audit log file can not be opened for writing.
```

Explanation

Audit log file in the Encryption Key Manager configuration specified by the property `Audit.handler.file.name` cannot be opened for writing.

System Response

The Encryption Key Manager does not start.

Operator Response

Please check the permissions on the given audit file and directory and try restarting the Encryption Key Manager.

Unable to Load the Admin Keystore

Text

```
Keystore for Admin cannot be loaded.
```

Explanation

Admin keystore supplied to the Encryption Key Manager cannot be loaded. Admin keystore is used between Encryption Key Manager servers for server side communication in multi-server environment.

System Response

The Encryption Key Manager does not start.

Operator Response

Check the configuration file setup. Make sure the properties `admin.keystore.file`, `admin.keystore.provider` and `admin.keystore.type` in the Encryption Key Manager configuration file are correct (refer to Appendix B) and the keystore file exists and has read permission. Make sure the password supplied for admin keystore either through `admin.keystore.password` property or entered on the command line is correct. Try restarting the Encryption Key Manager.

Unable to load the keystore

Text

```
Keystore for EKM can not be loaded.
```

Explanation

Keystore specified to the Encryption Key Manager cannot be loaded.

System Response

The Encryption Key Manager does not start.

Operator Response

Check the configuration file setup. Make sure the properties `config.keystore.file`, `config.keystore.provider` and `config.keystore.type` in the Encryption Key Manager configuration file are correct and the keystore file exists and has read permission. Make sure the password supplied for the Encryption Key Manager keystore either through `config.keystore.password` property or entered on the command line is correct. Try restarting.

Unable to Load the Transport Keystore

Text

```
Transport keystore cannot be loaded.
```

Explanation

Transport keystore supplied to the Encryption Key Manager cannot be loaded. Transport keystore is used between Encryption Key Manager servers for client side communication in multi-server environment.

System Response

The Encryption Key Manager does not start.

Operator Response

Check the configuration file setup. Make sure the properties `transport.keystore.file`, `transport.keystore.provider` and `transport.keystore.type` in the Encryption Key Manager configuration file are correct and the keystore file exists and has read permission. Make sure the password supplied for admin keystore either through `transport.keystore.password` property or entered on the command line is correct. Try restarting Encryption Key Manager.

Unsupported Action

Text

```
User entered action for the CLI which is not supported for EKM.
```

Explanation

Action supplied for **sync** command is not supported or understood by the Encryption Key Manager. The valid actions are merge or rewrite.

Operator Response

Check the command syntax using **help** and try again.

You can use Hardware Crypto Keystore for EKM or Remote Admin Connection, but not Both

Text

```
You can use hardware crypto keystore for EKM or remote Admin connection,
but not for both.
```

Explanation

Hardware crypto can be used either for keystore for the Encryption Key Manager or remote Admin connection, but not for both.

System Response

The Encryption Key Manager server does not start.

Operator Response

Check the keystores used for the Encryption Key Manager and Admin SSL connection. If the keystores used are the same, that is, both the properties `Admin.ssl.keystore.name` and `config.keystore.file` are the same, then `Admin.ssl.keystore.type` cannot be equal to **PKCS11IMPLKS**. Refer to Chapter 3, "Installing the Encryption Key Manager and Keystores," on page 45 for details on configuring the Encryption Key Manager.

You can use Hardware Crypto Keystore for Client Side SSL Connection or Server Side SSL Connection but not Both

Text

```
You can use hardware crypto keystore for client side SSL connection or
server side SSL connection but not both.
```

Explanation

Hardware crypto can be used either for keystore for client side SSL connection or for server side SSL connection but not for both connections.

System Response

The Encryption Key Manager server does not start.

Operator Response

Check the properties `Admin.ssl.keystore.type` and `TransportListener.ssl.keystore.type` in the configuration file provided for the Encryption Key Manager. Both of the values cannot be equal to **PKCS11IMPLKS**, that is, both of them cannot use hardware crypto.. Refer to Chapter 3, “Installing the Encryption Key Manager and Keystores,” on page 45 for details on configuring the Encryption Key Manager.

Chapter 8. Audit Records

Note: The audit record formats described in this chapter are not considered to be programming interfaces. The format of these records may change from release to release. The format is documented in this chapter in case some parsing of the audit records is desired.

Audit Overview

The audit subsystem writes textual audit records to a set of sequential files as various auditable events occur during the Encryption Key Manager's processing of requests. The audit subsystem writes to a file (directory and file name are configurable). The file size of these files is also configurable. As records are written to the file, and the size of the file reaches the configurable size, then the file is closed, renamed based on the current timestamp, and another file is opened and records are written to the newly created file. The overall log of audit records is thus separated into configurable sized files, their names sequenced by the timestamp of when the size of the file exceeds the configurable size.

To keep the amount of information in the overall audit log (spanning all of the sequential files created) from growing too large and exceeding the space available in the filesystem, you might consider creating a script or program to monitor the set of files in the configured audit directory/folder/container. As files are closed and named based on the timestamp, the file's contents should be copied and appended to the desired long-term, continuous log location and then cleared. Be careful not to remove or alter the file which is having records written to it by the Encryption Key Manager while running (this file does not have a timestamp in the file name).

Audit Configuration Parameters

The following parameters are used in the Encryption Key Manager's configuration file to control which events are logged in the audit log, where the audit log files are written to, and the maximum size of the audit log files.

Audit.event.types

Syntax

Audit.event.types={*type*[;*type*]}

Usage

Used to specify which audit types should be sent to the audit log. Possible values for configuration parameter are:

all	All event types
authentication	Authentication events
data_synchronization	Events that occur during synchronization of information between Encryption Key Manager servers
runtime	Events that occur as a part of processing operations and requests sent to the Encryption Key Manager

configuration_management	Events that occur as configuration changes are made
resource_management	Events that occur as resource (tape drive) settings in the Encryption Key Manager are changed

Examples

An example specification for this configuration value is:

```
Audit.event.types=all
```

Another example is:

```
Audit.event.types=authentication;runtime;resource_management
```

Audit.event.outcome

Syntax

Audit.event.outcome={*outcome*[;*outcome*]}

Usage

Used to indicate whether events occurring as a result of successful operations, unsuccessful operations, or both should be audited. Specify **success** for events to be logged which occur as a result of successful operations. Specify **failure** for events to be logged which occur as a result of unsuccessful operations.

Examples

An example specification for this configuration value is:

```
Audit.event.outcome=failure
```

To enable both successful and unsuccessful cases:

```
Audit.event.outcome=success;failure
```

Audit.eventQueue.max

Syntax

Audit.eventQueue.max=*number_events*

Usage

Used to set the maximum number of event objects to be held in the memory queue. This parameter is optional but recommended. the default is zero.

Example

```
Audit.eventQueue.max=8
```

Audit.handler.file.directory

Syntax

Audit.handler.file.directory=*directoryName*

Usage

This parameter is used to indicate into which directory the audit record files should be written. On z/OS, this must specify a directory in the UNIX System Services file system. Note that if the directory does not exist, the Encryption Key Manager will attempt to create the directory. If not successful, however, the Encryption Key Manager will not start. It is recommended that the directory exist prior to running the Encryption Key Manager. Note also that the User ID under which the Encryption Key Manager runs must have write access to the directory specified.

Examples

To set the directory to **/var/ekm/ekm1/audit**:

```
Audit.handler.file.directory=/var/ekm/ekm1/audit
```

Audit.handler.file.size

Syntax

Audit.handler.file.size=*sizeInKiloBytes*

Usage

This parameter is used to indicate the size limit upon which an audit file is closed and a new audit file is then written to. Note that the actual size of the resulting audit file may exceed this value by several bytes as the file is closed after the size limit has been exceeded.

Examples

To set the maximum file size to roughly 2 megabytes, enter:

```
Audit.handler.file.size=2000
```

Audit.handler.file.name

Syntax

Audit.handler.file.name=*fileName*

Usage

Use this parameter to specify the base file name, within the specified audit directory to use as the base name in creating audit log files. Note that this parameter must contain only the base file name and not the fully qualified path name. The full name of the audit log file will have the value corresponding to the time upon which the file was written appended to this name.

To show this, consider an example where the Audit.handler.file.name value is set to **ekm.log**. The full name of the file(s) will be something like: `ekm.log.2315003554`. The appended string can be used to help determine the order in which the audit log files were created – higher number values indicate newer audit log files.

Examples

An example setting the base name to **ekm.log** is:

```
Audit.handler.file.name=ekm.log
```

Audit.handler.file.multithreads

Syntax

Audit.handler.file.multithreads={**yes** | **true** | **no** | **false**}

Usage

If specified as **true**, then a separate thread is used to write the event data to the audit log, allowing the current thread of execution (operation) to continue without waiting for the write to the audit log to complete. Use of multiple threads is the default behavior.

Examples

An example setting the base name to **true** is:

```
Audit.handler.file.multithreads=true
```

Audit.handler.file.threadlifespan

Syntax

Audit.handler.file.threadlifespan=*timeInSeconds*

Usage

This parameter is used to specify the maximum time a thread should be expected to require in order to write an audit log entry. This value is used during clean up processing to allow threads to complete their work before interrupting them. If a background thread has not completed its work within the time allotted by the threadlifespan parameter, then upon clean up processing, the thread will be interrupted.

Examples

To set the expected time a thread to write to the audit log should require to 10 seconds, specify:

```
Audit.handler.file.threadlifespan=10
```

Audit Record Format

All audit records use a similar output format which is described here. All audit records contain some common information including timestamp and record type, along with information specific to the audit event which occurred. The general format for audit records is shown here:

```
AuditRecordType:[
  timestamp=timestamp
  Attribute Name=Attribute Value
  ...
  ]
```

Each record spans multiple lines in the file, with the first line of the record beginning with the audit record type beginning at the first character on the line, followed by a colon (;) and an opening left bracket ([). Subsequent lines associated with the same audit record are indented two (2) spaces to assist in readability of the log records. The last line for a single audit record contains a closing right bracket (]) indented two (2) spaces. The number of lines for each audit record varies based on the audit record type and the additional attribute information that is provided with the audit record.

The timestamp for the audit records is based on the system clock of the system on which the Encryption Key Manager is running. If these records are to be correlated based on timestamp with events occurring on other systems, some type of time synchronization should be used to ensure that the clocks of the various systems in the environment are synchronized to an acceptable level of accuracy.

Audit Points in the Encryption Key Manager

The Encryption Key Manager can write audit records, based on configuration, for many events that occur during the processing of requests. In this section, the set of events that can be audited is described along with the audit record configuration category, which must be enabled in order for these audit records to be written to the audit files (see Table 28).

Table 28. Audit record types that the Encryption Key Manager writes to audit files

Audit Record Type	Audit Type	Description
Authentication	`authentication`	Used to log authentication events
Data Synchronization	`data_synchronization`	Used to log data synchronization processing
Runtime	`runtime`	Used to log various important processing events which occur within the Encryption Key Manager server while handling requests
Resource Management	`resource_management`	Used to log changes to how resources are configured to the Encryption Key Manager
Configuration Management	`configuration_management`	Used to log changes to the configuration of the Encryption Key Manager server

Audit Record Attributes

The following lists show the attributes available to each of the audit record types.

Authentication event

The format for these records is:

```
Authentication event:[
  timestamp=timestamp
  event source=source
  outcome=outcome
  event type=SECURITY_AUTHN
  message=message
  authentication type=type
  users=users
  ]
```

Note that the message value only appears if information for it is available.

Data Synchronization event

The format for these records is:

```
Data synchronization event:
  timestamp=timestamp
  event source=source
  outcome=outcome
  event type=SECURITY_DATA_SYNC
  message=message
  action=action
  resource=resource
  user=user
  ]
```

Note that the message and user values only appear if information for them is available.

Runtime event

The format for these records is:

```
Runtime event:
  timestamp=timestamp
  event source=source
  outcome=outcome
  event type=SECURITY_RUNTIME
  message=message
  resource=resource
  action=action
  user=user
  ]
```

Note that the message and user values only appear if information for them is available.

Resource Management event

The format for these records is:

```
Resource management event:
  timestamp=timestamp
  event source=source
  outcome=outcome
  event type=SECURITY_MGMT_RESOURCE
  message=message
  action=action
  user=user
  resource=resource
  ]
```

Note that the message value only appears if information for it is available.

Configuration Management event

The format for these records is:

```
Configuration management event:
  timestamp=timestamp
  event source=source
  outcome=outcome
  event type=SECURITY_MGMT_CONFIG
  message=message
```

```
action=action
command type=type
user=user
]
```

Note that the `message` value only appears if information for it is available.

Audited Events

Table 29 describes the events that cause audit records to be created. The table lists the audit record type that is logged when this event occurs.

Table 29. Audit record types by audited event

Audited Event	Audit Record Type
User successfully authenticated	authentication
User authentication failed	authentication
Data successfully sent to other EKM	data_synchronization
Error sending data to other EKM	data_synchronization
sync command processed	data_synchronization
Error processing sync command	data_synchronization
Command line processing started	runtime
exit command received	runtime
Unknown command entered	runtime
Message received from drive	runtime
Error processing message from drive	runtime
Error from message received from drive	runtime
Error updating drive table with information received from drive	runtime
Error retrieving information from drive table	runtime
Error retrieving information from keystore	runtime
Error processing certificate from keystore	runtime
Error finding private key from keystore	runtime
Error computing cryptographic values	runtime
Message exchange processed successfully	runtime
Message processing started	runtime
Command line processing started	runtime
Problem found using cryptographic services	runtime
New drive discovered	runtime
Error configuring drive to drive table	runtime
Successfully started processing messages from drive	runtime
Received and processed stopekm command	runtime
Drive removed from drive table	resource_management
Error removing drive from drive table	resource_management
Drive table import successful	resource_management
Error importing drive table	resource_management

Table 29. Audit record types by audited event (continued)

Audited Event	Audit Record Type
Drive table export successful	resource_management
Error exporting drive table	resource_management
listcerts command successful	resource_management
Drive add to drive table successful	resource_management
Error adding drive to drive table	resource_management
listdrives command successful	resource_management
Error processing listdrives command	resource_management
Drive table modify successful	resource_management
Error modifying drive table	resource_management
Successful KeyStore open	resource_management
Error opening KeyStore	resource_management
Configuration property changed	configuration_management
Error changing configuration property	configuration_management
Configuration property deleted	configuration_management
Error deleting configuration property	configuration_management
Configuration import successful	configuration_management
Error importing configuration	configuration_management
Configuration export successful	configuration_management
Error exporting configuration	configuration_management
listconfig command successful	configuration_management

Chapter 9. Using Metadata

The Encryption Key Manager must be configured to create an XML file that captures vital information as data is being encrypted and written to tape. This file can be queried by volume serial number to display the alias or key label that was used on the volume. Conversely, the file can be queried by alias to display all volumes associated with that key label/alias.

Note: If you do not configure a metadata file, the Encryption Key Manager will not start.

As encryption processing is performed, the Encryption Key Manager collects the following data:

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

When the collected data reaches a certain limit, it is written to an XML file. The default limit, which can be set in the Encryption Key Manager properties file (KeyManagerConfig.properties), is 100 records. Once the file is written, it can be queried as long as the Encryption Key Manager is running. To prevent the file from growing too large, it is automatically rolled over to a new file after a maximum file size is reached. The default maximum file size for rollover, which can also be set in the Encryption Key Manager properties file, is 1 MB. Only a current and a previous file version is saved. The values to set in the Encryption Key Manager configuration properties file are:

Audit.metadata.file.name
Name of XML file where metadata is saved. This is required.

Audit.metadata.file.size
The maximum filesize, specified in kilobytes, before rolling the file over from current to previous version. This is optional. The default is 1024 (1MB).

Audit.metadata.file.cachecount
The number of records to be cached before writing the metadata file. This is optional. The default is 100.

XML File Format

The file contains records in the following format.

```
<KeyUsageEvent>
 <DriveSSN>FVTDRIVE0000</driveSSN>          -Drive Serial Number
 <VolSer>TESTER</volSer>                  -Volume Serial
 <DriveWWN>57574E414D453030</driveWWN>       -drive WWN
 <keyAlias2>cert2</keyAlias2>              -Key Alias1
 <keyAlias1>cert1</keyAlias1>             - keyAlias2
 <dateTime>Tue Feb 20 09:18:07 CST 2007</dateTime>    - creation date
</KeyUsageEvent>
```

Note: for LTO 4 drives there will only be <keyAlias1></keyAlias1> record and DKi will be recorded.

Querying the Metadata XML File

Use the EKMDataParser tool to query the metadata file. This tool parses the XML file using Document Object Model (DOM) techniques and cannot be run from the Encryption Key Manager command line interface. It is invoked as follows:

java com.ibm.keymanager.tools.EKMDataParser -filename *full_path_to_metadata_file* {**-volser** *volser* | **-keyalias** *alias*}

metadata_path
: This is the same directory path specified for the metadata file in Audit.metadata.file.name in the **KeyManagerConfig.properties** file.

-filename
: *filename* is required and must be the name of the XML metadata file. This will usually match the name specified in the Audit.metadata.file.name property in the **KeyManagerConfig.properties** file.

-volser
: The volume serial number of the tape cartridge you are searching for in the XML file. Either **-volser** or **-keyalias** must be specified.

-keyalias
: The key label or alias you are searching for in the XML file. Either **-volser** or **-keyalias** must be specified.

Example

Assuming that the metadata filename property (Audit.metadata.file.name) in **KeyManagerConfig.properties** is set to a value of `metadata` and the file is located in your local directory where the Encryption Key Manager runs, the following command would filter (display) only the XML records related to volser 72448:

```
<jvm_path>/bin/java com.ibm.keymanager.tools.EKMDataParser -filename metadata -volser 72448
```

The output would be formatted as follows:

Table 30. Metadata Query Output Format

keyalias1	keyalias2	volSer	dateTime	driveSSN	dki
cert1	cert2	72448	Wed Mar 14 10:31:32 CDT 2007	FVTDRIVE0004	

Recovering from a Corrupted Metadata File

The Encryption Key Manager metadata file can become corrupted if the Encryption Key Manager is improperly shutdown or the system where the Encryption Key Manager is running crashes. Improper editing or modification of the metadata file can also corrupt it. The corruption will go unnoticed until the EKMDataParser parses the metadata file. The EKMDataParser may fail with an error similar to the following:

```
[Fatal Error] EKMData.xml:290:16: The end-tag for element type "KeyUsageEvent" must
 end with a '>' delimiter.
org.xml.sax.SAXParseException: The end-tag for element type "KeyUsageEvent" must
 end with a '>' delimiter.
at org.apache.xerces.parsers.DOMParser.parse(Unknown Source)
at org.apache.xerces.jaxp.DocumentBuilderImpl.parse(Unknown Source)
at javax.xml.parsers.DocumentBuilder.parse(Unknown Source)
at com.ibm.keymanager.tools.EKMDataParser.a(EKMDataParser.java:136)
at com.ibm.keymanager.tools.EKMDataParser.a(EKMDataParser.java:26)
at com.ibm.keymanager.tools.EKMDataParser.main(EKMDataParser.java:93)
```

If this error occurs, it is due to a missing XML ending tag for an element. The Encryption Key Manager metadata file can be recovered to allow the EKMDataParser to parse the file again.

1. Make a backup copy of the Encryption Key Manager metadata file.
2. Edit the Encryption Key Manager metadata file.
3. In XML, there is should be an initial tag and a corresponding ending tag for each piece of data or event.
   - Some examples of an initial tag:
     - <KeyUsageEvent>
     - <driveSSN>
     - <keyAlias1>
   - Some examples of an ending tag:
     - </KeyUsageEvent>
     - </driveSSN>
     - </keyAlias1>
4. Scan the file and look for unmatched tags. The error message from the EKMDataParser lists which tag is missing its ending tag. This should make the search somewhat easier.
5. When an unmatched tag is found, temporarily delete the event or add the necessary tags to complete the event.
   - For example, the following excerpt from a Encryption Key Manager metadata file shows a first KeyUsageEvent that has no ending tag:

     ```
     <KeyUsageEvent>
     <driveSSN>001310000109</driveSSN>
     <volSer>      </volSer>
     <driveWWN>5005076312418B07</driveWWN>
     <keyAlias1>key00000000000000000F</keyAlias1>
     <dki>6B657900000000000000000F</dki>
     <dateTime>Thu Aug 30 09:50:53 MDT 2007</dateTime>
     <KeyUsageEvent>
     <driveSSN>001310000100</driveSSN>
     <volSer>      </volSer>
     <driveWWN>5005076312418ABB</driveWWN>
     <keyAlias1>key000000000000000000</keyAlias1>
     <dki>6B6579000000000000000000</dki>
     <dateTime>Thu Sep 06 16:49:39 MDT 2007</dateTime>
     </KeyUsageEvent>
     ```

     Adding a </KeyUsageEvent> between the lines <dateTime>Thu Aug 30 09:50:53 MDT 2007</dateTime> and <KeyUsageEvent> would complete the first <KeyUsageEvent>.

Repairing the file corruption will allow the EKMDataParser to successfully parse the data.

Appendix A. Sample Files

Sample configuration properties files are available for download from the Encryption Key Manager website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 in the EKMServicesAndSamples file.

Sample Startup Daemon Script

Attention: It is impossible to overstate the importance of preserving your keystore data. Without access to your keystore you will be unable to decrypt your encrypted tapes. Please be sure to save your keystore and password information.

z/OS Platforms

Create/Edit contents as shown in the following example. The text in italics for comment purposes only and need not be entered. Be sure to review the installation documentation for the z/OS Java product which contains additional guidelines, annotated samples, and step by step installation instructions for the JZOS launcher function.

```
//EKM PROC JAVACLS='com.ibm.jzosekm.EKMConsoleWrapper',
//   ARGS=,                            < Args to Java class
//   LIBRARY='SYS1.SIEALNKE',          < STEPLIB FOR JVMLDM module
//   VERSION='14',                     < 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
//*
//********************************************************************
//EKM  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=USER.PLX4.PROCLIB(EKM2ENV),DISP=SHR
//*
```

Sample Encryption Key Manager Server Configuration Properties Files

The following is a sample Encryption Key Manager properties file with all of the keystore entries pointing to the same software keystore:

```
Admin.ssl.keystore.name = /keymanager/testkeys
Admin.ssl.keystore.type = jceks
Admin.ssl.truststore.name = /keymanager/testkeys
Admin.ssl.truststore.type = jceks
```

```
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /keymanager/audit
Audit.handler.file.name = kms_audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /keymanager/metafile.xml
config.drivetable.file.url = FILE:///keymanager/drivetable
config.keystore.file = /keymanager/testkeys
config.keystore.provider = IBMJCE
config.keystore.type = jceks
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = /keymanager/testkeys
TransportListener.ssl.keystore.type = jceks
TransportListener.ssl.port = 443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = /keymanager/testkeys
TransportListener.ssl.truststore.type = jceks
TransportListener.tcp.port = 3801
```

This is a sample Encryption Key Manager properties file with the certificates/keys used to secure the decrypting key held in a hardware keystore and the rest of the keystores using the same software keystore. The only difference between the properties file above and the one below are the entries in bold.

```
Admin.ssl.keystore.name = /keymanager/testkeys
Admin.ssl.keystore.type = jceks
Admin.ssl.truststore.name = /keymanager/testkeys
Admin.ssl.truststore.type = jceks
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /keymanager/audit
Audit.handler.file.name = kms_audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /keymanager/metafile.xml
config.drivetable.file.url = FILE:///keymanager/drivetable
config.keystore.file = /opt/Eracom/lib/libcryptoki.so
config.keystore.provider = IBMPKCS11|mpl
config.keystore.type = IBMPKCS11|MPLKS
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = /keymanager/testkeys
TransportListener.ssl.keystore.type = jceks
TransportListener.ssl.port = 443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = /keymanager/testkeys
TransportListener.ssl.truststore.type = jceks
TransportListener.tcp.port = 3801
```

This is a sample Encryption Key Manager properties file with all of the keystore entries pointing to a different keystore. Entries in bold differ from the first sample properties file above.

```
Admin.ssl.keystore.name = /keymanager/adminkeys.jceks
Admin.ssl.keystore.type = jceks
Admin.ssl.truststore.name = /keymanager/admintrustkeys
Admin.ssl.truststore.type = jceks
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
Audit.handler.file.directory = /keymanager/audit
Audit.handler.file.name = kms_audit.log
Audit.handler.file.size = 10000
Audit.metadata.file.name = /keymanager/metafile.xml
config.drivetable.file.url = FILE:///keymanager/drivetable
config.keystore.file = /keymanager/drive.keys
config.keystore.provider = IBMJCE
config.keystore.type = jceks
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.keystore.name = /keymanager/sslkeys
TransportListener.ssl.keystore.type = jceks
TransportListener.ssl.port = 443
TransportListener.ssl.protocols = SSL_TLS
TransportListener.ssl.truststore.name = /keymanager/ssltrustkeys
TransportListener.ssl.truststore.type = jceks
TransportListener.tcp.port = 3801
```

Sample Encryption Key Manager Client Configuration Properties File

```
TransportListener.ssl.truststore.name=EKMKeys.jck
debug.output.file=debug
TransportListener.ssl.ciphersuites=JSSE_ALL
TransportListener.ssl.host=localhost
TransportListener.ssl.keystore.type=jceks
TransportListener.ssl.truststore.type=jceks
debug.output=simple_file
TransportListener.ssl.port=443
TransportListener.ssl.keystore.name=EKMKeys.jck
TransportListener.ssl.protocols=SSL_TLS
```

Appendix B. Encryption Key Manager Configuration Properties Files

The Encryption Key Manager requires two configuration property files: one for the Encryption Key Manager server, and one for the CLI client. Each of these files is treated and parsed as a Java.util.Properties load file, which imposes certain restrictions on the format and specification of properties:

- Configuration properties are recorded one-per-line. The value(s) for a given property extend to the end of the line.
- Property values, such as passwords, that contain spaces need not be enclosed in quotation marks.
- Keystore passwords must not be greater than 127 characters in length.
- Accidental whitespace at the end of a line may be interpreted as part of a property value.

Sample configuration properties files are available for download from the Encryption Key Manager website at http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504 in the EKMServicesAndSamples file.

Encryption Key Manager Server Configuration Properties File

The following comprises the complete set of properties in the Encryption Key Manager server configuration file (KeyManagerConfig.properties). The order of property settings in the file does not matter. Comments may appear in the file. To add a comment, use a "#" in the first column of a line. On z/OS, "#" is expected to be in IBM-1047 codepage, i.e. X'7B'.

Note: Changes made to the KeyManagerConfig.properties file may be lost at shutdown. Therefore, be sure the Encryption Key Manager server is not running before editing configuration properties. To stop the Encryption Key Manager server issue the **stopekm** command from the CLI client. Your changes are activated when the Encryption Key Manager server is restarted.

Admin.ssl.ciphersuites = *value*

Specifies the cipher suites to be used for communication between Encryption Key Manager servers. A cipher suite describes the cryptographic algorithms and handshake protocols Transport Layer Security (TLS) and Secure Sockets Layer (SSL) use for data transfer.

Required	Optional.
Values	Possible values are any cipher suites supported by IBMJSSE2.
Default	JSSE_ALL

Admin.ssl.keystore.name = *value*

This is the name of the database of key pairs and certificates used for Secure Socket Layer client operations such as **sync** commands between Encryption Key Manager Servers. In a sync operation, the certificate that the Secure Sockets client presents to the Secure Sockets server comes from this keystore.

Required	Optional. Used only with **sync** command. Defaults to value of **config.keystore.file** property.

Admin.ssl.keystore.password = password
Password to access Admin.ssl.keystore.name

Required	Optional. If not supplied, may be prompted for on start of Encryption Key Manager. When specified, the value for this property is obfuscated for additional security and the stanza name itself in the properties file will be replaced with a new stanza that is named 'Admin.ssl.keystore.password.obfuscated.'

Admin.ssl.keystore.type = *value*
Type of keystore used.

Required	Optional.
Default	jceks

Admin.ssl.protocols = *value*
Security protocols.

Required	Optional.
Values	SSL_TLS | SSL | TLS
Default	SSL_TLS

Admin.ssl.timeout = *value*
Specifies how long a socket waits for a read() before throwing a SocketTimeoutException.

Required	Optional.
Values	Specified in minutes. 0 means no timeout
Default	1

Admin.ssl.truststore.name = *value*
This is the name of the database file that is used to check the trust of the Secure Sockets Server certificate that the server presents to the Secure Sockets client.

Required	Optional. Used only with **sync** command. Defaults to value of **config.keystore.file** property.

Admin.ssl.truststore.type = *value*
Type of keystore used.

Required	Optional.
Default	jceks

Audit.event.outcome = *value*
Only audit events that resulted in the specified outcome are recorded

Required	Yes.
Values	success | failure. Both can be specified separated by comma or semicolon.
Default	success

Audit.event.Queue.max = 0
The maximum number of event objects in the audit memory queue before they will be flushed to file.

Required	Optional. Recommended.
Values	0 - ? (0 means flush immediately.)
Default	0

Audit.event.types = *value*
Only audit events that resulted in the specified outcome are recorded

Required	Yes.
Values	all | authentication | authorization | data synchronization | runtime | audit management | authorization terminate | configuration management | resource management | none. Multiple values can be specified separated by a comma or semicolon.
Default	all

Audit.handler.file.directory = ../audit
Directory where Audit.handler.file.name will be located

Required	Optional. Recommended.

Audit.handler.file.multithreads = *value*
Specifies if the audit handler should dispatch separate threads to process audit records.

Required	Optional.
Values	true | false
Default	true

Audit.handler.file.name = kms_audit.log
File name where audit entries will be logged.

Required	Yes.

Audit.handler.file.size = 100
Size to which Audit.Handler.file.name will grow before it begins to overwrite

Required	Optional. Recommended.
Values	0 - ? (specified in kilobytes.)
Default	100

Audit.handler.file.threadlifespan = *value*
Limits the lifetime of an audit record processing thread. Only useful if audit.handler.file.multithreads= true..

Required	Optional.
Values	Specified in milliseconds.
Default	10000

Audit.metadata.file.cachecount = 100
Specifies the number of records to store in memory before writing the metadata file.

Required	No
Default	100

Audit.metadata.file.name = *value*
Specifies the name of the XML file where metadata records are to be saved.

Required	Yes.

Audit.metadata.file.size = 1024
Specifies the maximum file size, specified in KB, the XML metadata file may achieve before the file is closed and a new file is started. Only a current and previous version of the file is saved.

Required	No
Default	1024

cert.valiDATE = *value*
Specifies whether to perform certificate date validation. A value of **true** specifies that the certificate's notBefore and notAfter dates are used to validate the certificate. Any certificate with a date range falling outside these Java DATE specifications cannot be used for encryption (but can still be used to decrypt, or read, encrypted tapes). If this property is set to a value of **false** (or any other value, or left unspecified), no other certificate date validation is performed. This property only applies to TS1120 or TS1130 Tape Drives. It is not used for LTO.

Required	Optional.
Values	true | false
Default	false

config.drivetable.file.url = FILE:../filedrive.table
File containing information concerning the tape drive such as serial number, certificates, etc.

Required	Yes.

config.keygroup.xml.file = *value*
Specifies the name of the XML file where individual aliases are stored by key groups. (Not used for TS1120 or TS1130.)

Required	Optional.

config.keystore.file = *value*
Specifies the keystore to be used.

Required	Yes.

config.keystore.password = password
Password to access config.keystore.file. When specified, the value for this property is obfuscated for additional security and the stanza name itself in the properties file will be replaced with a new stanza that is named 'config.keystore.password.obfuscated.'

Required	Optional. If not supplied, may be prompted for on start of the Encryption Key Manager.

config.keystore.provider = IBMJCE

Required	Optional.

config.keystore.type = jceks

Required	Optional. Recommended.
Default	jceks

debug = *value*
Enables debug for the specified Encryption Key Manager component.

Required	Optional.

Values	all | audit | server | drivetable | config | admin | transport | logic | keystore | console | <u>none</u>. Can take multiple values separated by commas.
Default	none

debug.output = *value*
Routes debug output to specified location.

Required	Optional.
Values	simple_file | console (not recommended).

debug.output.file = debug
Path and filename where debug output is to be written.

Required	Optional. Required when debug.output = simple_file. Path to file must exist.

drive.acceptUnknownDrives = *value*
Automatically adds new drive contacting the Encryption Key Manager to drive table

Required	Yes.
Values	true | <u>false</u>
Default	false

Security note - This setting in combination with a valid drive.default.alias1 setting allows tapedrives that connect to the Encryption Key Manager to be added and operational without a administrator validating that addition. See "Automatically update tape drive table" in Chapter 3 for more information.

drive.default.alias1 = *value*
| Specifies a TS1120 or TS1130 drive default alias for use if one is not specified in drive table. This value can be different from the value specified for drive.default.alias2 or the same. (Not used for LTO.)

Required	Optional.

drive.default.alias2 = *value*
| Specifies a TS1120 or TS1130 drive default alias for use if one is not specified in drive table. This value can be different from the value specified for drive.default.alias1 or the same. (Not used for LTO.)

Required	Optional.

fips = *value*
Federal Information Processing Standard. See "Federal Information Processing Standard 140-2 Considerations" in Chapter 2 for more information.

Required	Optional.
Values	on | <u>off</u>
Default	off

Note: You should not use hardware-based keystore types when the fips parameter is set on.

maximum.threads = 200
Maximum number of threads the Encryption Key Manager can create.

Required	Optional.

requireHardwareProtectionForSymmetricKeys = *value*

Specifies whether symmetric keys need to be protected on z/OS if hardware crypto is used.

Required	Optional. Applies to z/OS only.
Values	true | false
Default	false

Server.authMechanism = *value*

Specifies the authentication mechanism to be used with local/remote clients. When the value is set to `EKM`, the CLI client user must login to the server using usr/passwd as EKMAdmin/changeME. (This password can be changed with chgpasswd command.) When the value is specified as `LocalOS`, client authentication is done against the local operating system registry. The CLI client user must login to the server with OS usr/passwd. For local OS-based authentication on Linux and Unix-based platforms, additional steps are required:

1. Download EKMServiceAndSamples.jar on the server machine from the Encryption Key Manager website (**http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504**) .
2. Extract the contents of into a temporary directory
3. Copy the libjaasauth.so file from the LocalOS-setup appropriate to your platform to <Java-Home>/jre/bin for AIX and to <Java-Home>/jre/lib/$architecture for Linux, Solaris, and HP-UX.

For z/OS, i5/OS, and Windows platforms this file is not necessary.

A readme file on the Encryption Key Manager website provides more installation details.

Required	Optional.
Values	EKM | LocalOS
Default	EKM

Server.password = *value*

Internal property. Do not edit.

symmetricKeySet = {GroupID | keyAliasList [, keyAliasList,]}

| Specifies the symmetric key aliases and key groups to be used for LTO 4 tape drives. (Not used for TS1120 or TS1130.)

Required Optional. Applies to LTO 4 tape cartridges only.

Values

Specify one value for *GroupID* or one or more values for *keyAliasList*.

GroupID specifies a key group name to prime the list of symmetric keys and serve as default when no alias is specified for the tape drive. The *GroupID* must match an existing key group ID in the KeyGroups.xml file. If not, a KeyManageException is returned. If more than one *GroupID* is specified, a KeyManagerException is returned. When you specify a valid *GroupID*, the last key used in the Key Groups XML is tracked and a random selection for the next key is used each time getKey is called from the

KeyGroups.xml for the list of symmetric keys.Each specification of *keyAliasList* contains either a value for *keyAlias* or *keyAliasRange*.

keyAlias specifies the Backus-Naur Form (BNF) for a name or alias of a symmetric key in the keystore up to 12 characters long, or a sequentialKeyID exactly 21 characters long.

keyAliasRange specifies a sequentialKeyID and hexadecimal digits up to 18 characters, separated by a hyphen (-). If 18 characters are specified the first two characters must be 00. Must be specified on one line and contain no cr-lf.

GroupID specifies the name of a group of aliases.

Example `symmetricKeySet = KMA0238ab34,KMB0000034acd2345678a,THZ001-FF` This instructs the Encryption Key Manager to use aliases KMA0238ab34, KMB0000034acd2345678a and the range of aliases THZ000000000000000001 through THZ0000000000000000FF when serving keys to LTO 4 tape drives. These keys must exist in the keystore specified by **config.keystore.file** in the properties file.

sync.action = *value*

Specifies what should be done with the data during an auto synchronize.

Required	Optional.
Values	rewrite | merge
Default	merge

Note: merging configuration information is the same as rewriting it.

sync.ipaddress = *ip_addr:ssl*

Specifies the IP address and port of the remote Encryption Key Manager to auto synchronize.

Required	Optional. If this property is unspecified or specified incorrectly, the sync function is disabled.
Values	IP address of remote server:SSL port number

sync.timeinhours = *value*

Specifies how many hours to wait before doing an auto synchronize with a remote Encryption Key Manager.

Required	Optional.
Values	Specified in hours.
Default	24

sync.type = *value*

Specifies what data to auto synchronize.

Required	Optional.
Values	config | drivetab | all
Default	drivetab

TransportListener.ssl.ciphersuites = JSSE_ALL

Cipher suites to be used for communication between Encryption Key

Manager servers. A cipher suite describes the cryptographic algorithms and handshake protocols Transport Layer Security (TLS) and Secure Sockets Layer (SSL) use for data transfer.

Required	Optional.
Values	Values – any cipher suites supported by IBMJSSE2.

TransportListener.ssl.clientauthentication = 0
SSL authentication needed for communication between Encryption Key Manager servers.

Required	Optional.
Values	0 - no client authentication (default) 1 - server wants to do client authentication with the client 2 - the server must do client authentication with the client

TransportListener.ssl.keystore.name = *value*
The name of the database used by the Encryption Key Manager Server to hold the certificate and private keys for the Secure Socket Server. This certificate is given to the Secure Socket client for authentication and trust checking. This keystore is also used by the Encryption Key Manager Client to talk to the Encryption Key Manager Server and act as a Secure Sockets client.

Required	Yes.

TransportListener.ssl.keystore.password = password
Password to access TransportListener.ssl.keystore.name. When specified, the value for this property is obfuscated for additional security and the stanza name itself in the properties file will be replaced with a new stanza that is named 'TransportListener.ssl.keystore.password.obfuscated.'

Required	Optional.

TransportListener.ssl.keystore.type = jceks

Required	Optional. Recommended.
Values	<u>JCEKS</u> | PKCS11ImplKS | IBMi5OSKeystore | JCE4758KS | JCECCAKS | JCERACFKS | JCE4758RACFKS | JCECCARACFKS

TransportListener.ssl.port = *value*
Port the Encryption Key Manager server will listen on for requests from other Encryption Key Manager Servers or from the Encryption Key Manager CLI client.

Required	Yes.
Values	Port number, 443 for example. This must match the TransportListener.ssl.port property in the CLI client configuration properties file.

TransportListener.ssl.protocols = SSL_TLS
Security protocols

Required	Optional.
Values	SSL_TLS (default) | SSL | TLS

TransportListener.ssl.timeout = 10
Specifies how long socket waits on a read() before throwing a SocketTimeoutException.

Required	Optional.
Value	Specified in minutes.
Default	1

TransportListener.ssl.truststore.name = *value*
The name of the database of public keys and signed certificates used to verify the identities of other clients and servers. If the TransportListener.ssl.clientauthentication property is **not** set to the default value of 0 (no client authentication), then the Encryption Key Manager Server, acting as the Secure Socket Server, must authenticate the client by using this file. This truststore is also used by the Encryption Key Manager Client to talk to the Encryption Key Manager Server and act as a Secure Sockets client.

Required	Yes.

TransportListener.ssl.truststore.type = jceks

Required	Optional. Recommended.
Values	JCEKS | PKCS11ImplKS | IBMi5OSKeystore | JCE4758KS | JCECCAKS | JCERACFKS | JCE4758RACFKS | JCECCARACFKS

TransportListener.tcp.port = *value*
Port the Encryption Key Manager server will listen on for requests from tape drives. The default TCP port number is 3801.

Required	Yes.
Values	Port number, 10 for example.

TransportListener.tcp.timeout = *value*
Specifies how long a socket waits on a read() before throwing a SocketTimeoutException.

Required	Optional.
Values	Specified in minutes. 0 means no timeout.
Default	10

useSKIDefaultLabels = *value*
Specifies whether the Encryption Key Manager creates Externally Encrypted Data Keys (EEDKs) for encrypted TS1120 or TS1130 cartridges using the X509 Subject Key Identifier (SKI) hash instead of the label when a default alias, either from the drive table or configuation file, is used. When a value of **true** is specified, the EEDKs are created using the SKI hash on the certificate. If this property is set to a value of **false** (or any other value, or left unspecified), the certificate's default label is used.

Note: This property has no function for LTO tapes.

Required	Optional.
Values	true | false
Default	false

zOSCompatibility = *value*
Use this option if plan to exchange tapes between z/OS and non-z/OS systems.

This option effects all keystores that may be configured to the Encryption Key Manager on all platforms that are supported by the Encryption Key Manager.

For z/OS platforms only: If Encryption Key Manager configuration file parameter `requireHardwareProtectionForSymmetricKeys` is set to true, this value must also be set to true to allow the key protecting the data key to be encrypted in host storage.

Required Optional.

Values

true	The encrypted tape can be read by an instance of Encryption Key Manager running on the z/OS Platform.
false	Default. The encrypted tape can only be read by an instance of Encryption Key Manager running on the z/OS platform if the Encryption Key Manager is running with Java 5.0 (it cannot be decrypted by an Encryption Key Manager running with Java 1.4.2). In addition, instances of Encryption Key Manager running on z/OS platforms cannot use the z/OS encrypted key support as provided by ICSF and zSeries hardware assisted cryptography.

CLI Client Configuration Properties File

This file, ClientKeyManagerConfig.properties, contains a subset of the properties contained in the KeyManagerConfig.properties file. This subset includes the following properties.

TransportListener.ssl.ciphersuites = JSSE_ALL

Cipher suites to be used for communication between Encryption Key Manager servers and CLI clients. A cipher suite describes the cryptographic algorithms and handshake protocols Transport Layer Security (TLS) and Secure Sockets Layer (SSL) use for data transfer.

Required Optional.

Values This value must match the value specified for TransportListener.ssl.ciphersuites in the Encryption Key Manager Server properties file, KeyManagerConfig.properties.

TransportListener.ssl.host = *value*

Identifies the Encryption Key Manager Server to the Encryption Key Manager CLI Client.

Required Optional.

Values IP address or hostname

Default localhost

Examples

```
TransportListener.ssl.host = 9.24.136.444
TransportListener.ssl.host = ekmsvr02
```

Note: Not used in KeyManagerConfig.properties file.

TransportListener.ssl.keystore.name = *value*

This keystore is also used by the Encryption Key Manager Client to talk to the Encryption Key Manager Server and act as a Secure Sockets client.

Required	Yes.

TransportListener.ssl.keystore.type = jceks

Type of keystore.

Required	Optional. Recommended.
Default	jceks

TransportListener.ssl.port = *value*

This is the port the CLI client will use to communicate with Encryption Key Manager servers.

Required	Yes.
Values	This value must match the value specified for TransportListener.ssl.port in the Encryption Key Manager Server properties file, KeyManagerConfig.properties.

TransportListener.ssl.protocols = SSL_TLS

Security protocols

Required	Optional.
Values	This value must match the value specified for TransportListener.ssl.protocols in the Encryption Key Manager Server properties file, KeyManagerConfig.properties.

TransportListener.ssl.truststore.name = *value*

The name of the database of public keys and signed certificates used to verify the identities of other clients and servers.

Required	Yes.

TransportListener.ssl.truststore.type = jceks

Type of truststore.

Required	Optional. Recommended.
Default	jceks

Sample configuration properties files are available for download in the EKMServicesAndSamples file from the Encryption Key Manager website (**http://www.ibm.com/support/docview.wss?&uid=ssg1S4000504**) .

Appendix C. Frequently Asked Questions

What is the change to the delivery mechanism for IBM Java on non-IBM operating systems in 2008?

The Encryption Key Manager is written using IBM Java, and therefore requires IBM Java to run. The current delivery vehicle for IBM Java when the Encryption Key Manager is running on a non-IBM operating system is the TotalStorage Productivity Center/Limited Edition (TPC/LE) CD. The customer does not need to install TPC/LE. This is just the delivery vehicle for IBM Java on these platforms.

IBM Java can be obtained in this manner until February 8, 2008. After that date IBM will no longer offer TPC/LE. After February 8, 2008, customers who want to run the Encryption Key Manager on non-IBM operating systems must obtain IBM Java by ordering TPC/Basic Edition (TPC/BE). TPC/BE is a chargeable program product whereas TPC-LE is a no-charge offering. TPC/BE includes a one year warranty. Several maintenance options are also available for ordering with TPC/BE

Can some combination of application-based key management and system- or library-managed encryption be used?

No. Currently, Tivoli Storage Manager is the only application that provides application-managed tape encryption for TS1120 and TS1130 tape drives. When application-managed encryption is used, the encryption is transparent at the system and library layers. Likewise, when system- or library-managed encryption is used, the process is transparent at the other layers. Each method of encryption management is exclusive of the others. For system-managed encryption and library-managed encryption, the applications need not be changed in any way.

Must the Encryption Key Manager be installed and running on every system that might generate a request to encrypt or decrypt a tape?

With either library- or system-managed encryption, the system from which the tape drive write request originates need NOT be the system on which the Encryption Key Manager is running. Furthermore, an instance of Encryption Key Manager need NOT be running on every system from which an encrypting tape drive is accessed.

Since RACF keyrings don’t have a password, what should the following statements have coded for a password value?

Either omit these three statements from the configuration properties file or code them as follows:

```
config.keystore.password = password
TransportListener.ssl.keystore.password = password
TransportListener.ssl.truststore.password = password
```

Where password is, literally, password.

How do I know what keystore to choose?

The keystore choice is dependent upon the system on which the Encryption Key Manager is running. Different platforms have different, platform-specific features (often tied to the hardware crypto support on those platforms). These differences manifest themselves in the keystore choice. Once the platform on which the Encryption Key Manager will run is chosen (and with multiple Encryption Key Managers running, there may be multiple platforms used), then the keystore type must be chosen.

If I include the ″drive.acceptUnknownDrives = True″ parameter, should I still include the ″config.drivetable.file.url = FILE:/filename″ parameter in the configuration file?

config.drivetable.file.url must always be specified. It is where the drive information will be. If you set `drive.acceptUnknownDrives = True` you also should specify the `drive.default.alias1` and `drive.default.alias2` variables to the correct certificate alias/key label.

Is FILE:/filename the correct syntax for the `config.drivetable.file.url` property? FILE:///filename appears in the sample file, and FILE:../ in the description.

The examples are correct. This is a URL specification and is not what people normally expect for a directory structure specification

Must I use forward or backward slashes when specifying fully-qualified paths in the KeyManagerConfig.properties file for an instance of Encryption Key Manager running on Windows?

Because KeyManagerConfig.properties is a Java properties file, only forward slashes are recognized in pathnames, even in Windows. If you use back slashes in the KeyManagerConfig.properties file, errors will occur.

Why doesn't the Encryption Key Manager list command show the keys I've added (or deleted) to my keystore when using an nCipher Hardware Crypto card?

By default, the nCipher card caches the card contents in memory and returns the contents of this cache on a list command, even after issuing the refreshks Encryption Key Manager admin command. To resolve this issue, set the nCipher environment variable `CKNFAST_ASSUME_SINGLE_PROCESS=0`.

Does the Encryption Key Manager perform any Certificate Revocation List (CRL) checking?

No, the Encryption Key Manager does not perform any CRL checking

What happens when the certificate being used to encrypt the tapes expires? Will the Encryption Key Manager read previously encrypted tapes?

It does not matter to Encryption Key Manager if the certificate has expired. It will continue to honor these certificates and read previously encrypted tapes. However the expired certificate must remain in the keystore in order for previously encrypted tapes to be read or appended.

Will the Encryption Key Manager require that a certificate be renamed on renewal?

The Encryption Key Manager is configured by default to honor new key requests with expired certificates. When the Encryption Key Manager is configured this way certificate renewal is not required. If this function is disabled and this private key/certificate pair must still be used for new key requests, then the user must renew the certificate. The certificate alone (validity dates) would be renewed but not the associated keys.

Will later versions of Encryption Key Manager still read the encrypted tapes created with earlier versions of the software?

Yes. The Encryption Key Manager will honor certificates regardless of release.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries (or regions). Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to: *IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.*

For license inquiries regarding double-byte (DBCS0 information, contact the IBM Intellectual Property Department in your country (or region) or send inquiries, in writing, to: *IBM World Trade Asia Corporation 2-31 Roppongi 3-chome, Minato-ku, Tokyo 106, Japan*

The following paragraph does not apply to the United Kingdom or any other country (or region) where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states (or regions) do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, or other countries (or regions), or both:

Table 31. Trademarks

AIX	eServer	Netfinity®	System Storage
AS/400®	FICON	NUMA-Q®	System/390®
DFSMS/MVS®	IBM	OS/390®	Tivoli®
DFSMS/VM®	IBMLink™	OS/400®	TotalStorage
DFSMSdfp™	i5/OS	pSeries	VM/ESA®
DFSMShsm™	iSeries™	Redbooks	VSE/ESA™
DFSORT™	Magstar®	RS/6000®	xSeries
ES/3090™	MVS™	S/390®	z/OS
ES/9000®	MVS/ESA™	SD/2®	z/VM®
ESCON®	MVS/XA™	SP™	zSeries®

Intel is a registered trademark of Intel Corporation in the United States, or other countries (or regions), or both.

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

Microsoft, Windows, and Windows NT® are registered trademarks of Microsoft Corporation in the United States, or other countries (or regions), or both.

UNIX® is a registered trademark of The Open Group in the United States and other countries (or regions).

Glossary

This glossary defines the special terms, abbreviations, and acronyms used in this publication and other related publications.

AES. Advanced Encryption Standard. A block cipher adopted as an encryption standard by the US government.

alias. See key label.

certificate. A digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated.

certificate label. See key label.

certificate store. See keystore.

DK. Data Key. An alphanumeric string used to encrypt data.

EEDK. Externally Encrypted Data Key. A Data Key that has been encrypted (wrapped) by a Key Encryption Key prior to being stored in the data cartridge. See KEK.

EEFMT2. Enterprise Encryption Format. AES 256-bit encrypted data written recorded at the performance and capacity format used by the native 3592 Model E05.

encryption. The conversion of data into a cipher. A key is required to encrypt and decrypt the data. Encryption provides protection from persons or software that attempt to access the data without the key.

KEK. Key Encrypting Key. An alphanumeric, asymmetric key used to encrypt the Data Key. See EEDK.

key label. A unique identifier used to match the EEDK with the private key (KEK) required to unwrap the protected symmetric data key. Also called alias or certificate label depending on which keystore is used.

key ring. See keystore.

keystore. A database of private keys and their associated X.509 digital certificate chains used to authenticate the corresponding public keys. Also called certificate store or key ring in some environments.

PKDS. Public Key Data Set. Also PKA cryptographic Key Data Set.

private key. One key in an asymmetric key pair, typically used for decryption. The Encryption Key Manager uses private keys to unwrap protected AES data keys prior to decryption.

public key. One key in an asymmetric key pair, typically used for encryption. The Encryption Key Manager uses public keys to wrap (protect) AES data keys prior to storing them on the tape cartridge.

rekey. The process of changing the asymmetric Key Encrypting Key (KEK) that protects the Data Key (DK) stored on an already encrypted tape, thereby allowing different entities access to the data.

RSA. Rivest-Shamir-Adleman algorithm. A system for asymmetric, public-key cryptography used for encryption and authentication. It was invented in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The security of the system depends on the difficulty of factoring the product of two large prime numbers.

Index

Readers' Comments — We'd Like to Hear from You

IBM Encryption Key Manager component for the Java platform
Introduction, Planning, and User's Guide

Publication No. GA76-0418-08

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Send your comments to the address on the reverse side of this form.

If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
GA76-0418-08

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department GZW
9000 South Rita Road
Tucson, Arizona U.S.A. 85775-4401

Fold and Tape **Please do not staple** Fold and Tape

GA76-0418-08

Cut or Fold
Along Line

Printed in USA

GA76-0418-08

Spine information:

IBM Encryption Key Manager
component for the Java platform

IBM Encryption Key Manager Introduction, Planning,
and User's Guide