IBM

IBM System Storage Tape Encryption Solutions

- Learn about IBM TS1130 Tape Drive and Tivoli Key Lifecycle Manager
- Encrypt data on LTO and TS1100 series cartridges
- Discover usability enhancements

Babette Haeusser
Jonathan Barney
Arthur Colvig

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM System Storage Tape Encryption Solutions

May 2009

SG24-7320-02

Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

Third Edition (May 2009)

This edition applies to the introduction of the IBM TS1130 Tape Drive and Tivoli Lifecycle Manager (TKLM)

Contents

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. 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.

The following paragraph does not apply to the United Kingdom or any other country 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 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.

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.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX 5L™
AIX®
alphaWorks®
AS/400®
CICS®
DB2®
developerWorks®
ESCON®
FICON®
i5/OS®
IBM®
iSeries®
Language Environment®
Netfinity®
OS/400®
POWER®
pSeries®
RACF®
Rational®
Redbooks®
Redbooks (logo) ®
RS/6000®
S/390®
System i5®
System i®
System p®
System Storage™
System x®
System z9®
System z®
Tivoli®
TotalStorage®
VTAM®
WebSphere®
xSeries®
z/OS®
z/VM®
z/VSE™
z9®
zSeries®

The following terms are trademarks of other companies:

AMD, AMD Opteron, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

ACS, Red Hat, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

VMware, the VMware "boxes" logo and design are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

J2SE, Java, JDK, JNI, JRE, JVM, S24, Solaris, StorageTek, Sun, ZFS, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Internet Explorer, Microsoft, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel Itanium, Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

This IBM® Redbooks® publication gives a comprehensive overview of the IBM System Storage™ Tape Encryption solutions that started with the TS1120 Tape Drive in 2006 and have been made available in the TS7700 Virtualization Engine in early 2007. Also in 2007, the IBM Ultrium Linear Tape-Open (LTO) Generation 4 Tape Drive was announced including its support for tape data encryption. In 2008, additional enhancements to the tape drives that support encryption and to key management have been made. This edition of the book has been updated with information about the TS1130 Tape Drive and the IBM Tivoli® Key Lifecycle Manager (TKLM).

This publication is intended for System Programmers, Storage Administrators, Hardware and Software Planners, and other IT personnel involved in planning, implementing, and operating IBM tape data encryption solutions, and anyone seeking details about tape encryption.

This book also provides practical guidance for how to implement an enterprise-wide encryption solution. We describe the general concepts of encryption and the implementation options that are available when using IBM Tape to encrypt tape data. We explain the key management options, including the Encryption Key Manager, which is a Java™ application that allows for enterprise-wide keystores and key management across a wide variety of platforms. We also provide detailed information for planning, implementation, and operation of tape data encryption for IBM z/OS® and Open Systems hosts.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

Babette Haeusser is an IBM Certified IT Specialist and Tape Server Support Specialist at IBM Deutschland GmbH, Germany. She writes extensively and teaches IBM classes worldwide on all areas of Enterprise and Open Systems Tape and Tape Virtualization. Babette joined IBM in 1973 as an application programmer. In 1987, she became an MVS Systems Engineer and specialized in IBM Storage hardware and software, which she has supported in various job roles since then. Before joining the ITSO in early 2005, Babette worked in the Advanced Technical Sales Support team Europe. She led a team of specialists for Enterprise Storage, while she focused on Enterprise Tape, including tape libraries and Virtual Tape Servers.

Jonathan Barney is an Enterprise Security Architect with STG Lab Services, and CISSP. He works extensively on tape encryption customer implementations, and z/OS security solutions including PKI, TKE, and hardware cryptography. Jonathan joined IBM in August 2000 in the IBM System z® System Test Group. He has held various System z development and testing roles until moving to Lab Services.

Arthur Colvig is an Advisory Software Engineer with IBM Tivoli in Beaverton, Oregon. He has 7 years of experience as a Firmware Engineer on the IBM System Storage TS3500 Tape Library (TS3500 tape library). He currently works on the IBM SAN Volume Controller SMI-S API. Art joined IBM in 2001 as a Firmware Engineer on the TS3500. His areas of expertise include storage management, tape libraries and drives, network protocols, and robotics. He has patents in the areas of tape library virtualization, management security and distributed system firmware management.

The team: Babette, Arthur, and Jonathan

Thanks to the following people for their contributions to this project:

Alex R. Osuna,
IBM International Technical Support Organization, Tucson Arizona

Emma Jacobs, Ann Lund, Sangam Racherla,
IBM International Technical Support Organization

Arthur Bariska, Erika Dawson, Dan Estelle, MaYan Wilt,
IBM Tucson

Timothy Hahn,
IBM Software Architect IBM Rational® Enterprise Tools

Brian Goodman, Steven Hart, Sara Hughes, Khan V. Ngo,
IBM Systems &Technology Group, Systems Software Development

John Peck
IBM Software Group, Tivoli

Jeff Ziehm
IBM LTO/3592, Tape Performance - ATS Americas

Thanks to the authors of the previous editions of this book:

- Authors of the first edition, *IBM System Storage TS1120 Tape Encryption: Planning, Implementation, and Usage Guide*, published in December 2006:

  Anthony Abete, Arthur Bariska, Jonathan M. Barney, Babette Haeusser, Ulrich Nowotny

- Authors of the second edition, *IBM System Storage Tape Encryption Solutions*, published in March 2008:

  Anthony Abete, Babette Haeusser, Burt Loper, Axel Melber

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition also includes minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-7320-02
for *IBM System Storage Tape Encryption Solutions*
as created or updated on February 15, 2011.

May 2009, Third Edition

This revision reflects the addition, deletion, or modification of new and changed information.

New information

New information includes:

- IBM TS1130 Tape Drive
- Tivoli Key Lifecycle Manager

Changed information

Changed information includes:

- Updates for TS7700 Encryption
- Updates for the Open Systems Tape Library
- Updates for System z Software support

Part 1

Introducing IBM tape encryption solutions

In this part, we introduce you to the IBM data encryption solutions and describe the underlying concepts and the components that are required for tape encryption. We also provide you with general planning information.

1

Introduction to tape encryption

IBM has three tape drives that are capable of encrypting the data that is written on the tape cartridge:

- IBM System Storage TS1130 Model E06 and Model EU6 Tape Drive
- IBM System Storage TS1120 Model E05 Tape Drive
- IBM System Storage Linear Tape-Open (LTO) Ultrium Generation 4 Tape Drive

In this document, we use abbreviations and refer to the drives as the TS1130 Tape Drive, TS1120 Tape Drive, and the LTO4 Tape Drive.

The *IBM System Storage TS1130 Tape Drive Model E06 and Model EU6* have the capability to encrypt data on tape cartridges. This encryption capability is standard on all TS1130 Tape Drives and is available as a chargeable upgrade feature for existing installed TS1120 Model E05 Tape Drives shipped prior to 8 September 2006. The encryption capability includes drive hardware, and microcode additions and changes.

The *IBM System Storage TS1120 Tape Drive Model E05* shown in Figure 1-1 has the capability to encrypt data on tape cartridges. The TS1120 Tape Drive Model E05 has been enhanced to support data encryption. Starting with shipments beginning 8 September 2006, this encryption capability is standard on all TS1120 Model E05 Tape Drives and is available as a chargeable upgrade feature for existing installed TS1120 Model E05 Tape Drives shipped prior to 8 September 2006. The encryption capability includes drive hardware, and licensed internal code additions and changes.

Figure 1-1 IBM System Storage TS1120 Model E05 Tape Drive

The *IBM System Storage LTO Ultrium Generation 4 Tape Drive* shown in Figure 1-2 has the capability to encrypt data on tape cartridges. The LTO4 Tape Drive was originally designed to support data encryption, and therefore, all LTO4 Tape Drives come standard with the data encryption capability. The encryption capability includes drive hardware, and licensed internal code additions and changes.

Although the LTO4 drive can read LTO2 and LTO3 cartridges and can write to LTO3 cartridges, LTO Ultrium Generation 4 cartridges are required to support encryption.

Figure 1-2 IBM System Storage LTO Ultrium Generation 4 Tape Drive

Although other encryption solutions require hardware resources or processor power when using software encryption, tape data encryption is done with little or no impact on the performance of the TS1130 Tape Drive, TS1120 Tape Drive or the LTO4 Tape Drive. You can easily exchange encrypted tapes with your business partners or data centers that have the necessary key information to decrypt the data.

With the original encryption announcement for the TS1120 Tape Drive, IBM also introduced an IBM Encryption Key Manager (EKM) component for the Java platform feature and is designed to generate and communicate encryption keys for tape drives across the enterprise. The feature uses standard key repositories on supported platforms. The IBM tape data encryption solution provides an enterprise key management solution with common software for Open Systems and mainframe environments that allows sharing of a common keystore across platforms. Integration with z/OS policy, key management, and security capabilities provides a proven, highly secure infrastructure for encryption key management.

With the introduction of the Tivoli Key Lifecycle Manager (TKLM), IBM has made available the next generation of Key Manager software to enable serving keys to the encrypting tape drives. TKLM is intended to give a consistent look and feel for Key Management tasks across the brand, while simplifying those same key management tasks,

The *Encryption Facility for z/OS software* provides a complementary encryption method for sharing tape data with business partners who utilize different tape formats (other than TS1130, TS1120 or LTO4). Encryption Facility for z/OS uses the same secure infrastructure as used with the TS1130, TS1120 or LTO4, thus, providing a comprehensive solution for z/OS clients, encrypting data for internal use, and even encrypting data to share with business partners. Encryption Facility supports decryption of Encryption Facility encrypted data on all platforms across the brand.

IBM tape data encryption provides high performance data encryption. Encryption is performed at the tape drive hardware at the native speeds of the drive. It also supports encryption of large amounts of tape data for backup and archive purposes.

The IBM tape data encryption solution, utilizing theTS1130 Tape Drive, TS1120 Tape Drive or the LTO4 Tape Drive, offers a cost-effective solution for tape data encryption by offloading encryption tasks from the servers, leveraging existing tape infrastructure incorporated in standard IBM Tape Libraries, and eliminating the need for unique appliance hardware.

You can greatly simplify your tape data encryption management, because the solution provides functions that are transparent to your applications when encryption is managed by the operating system (system-managed encryption) or when encryption is managed by the tape library (library-managed encryption). For TS1130 and TS1120 encryption, the cartridge data key is stored in an encrypted form on the tape cartridge. For LTO4 encryption, a pointer to the data key that is used to encrypt the tape is stored on the tape cartridge. Support of a single key management approach can help reduce audit and compliance costs.

When taking a closer look at encryption, several of the most important questions that you want answered are:

- How does tape data encryption work?
- What should we encrypt, and what should we not encrypt?
- Why use tape data encryption? What are the benefits for my organization?

1.1 How tape data encryption works

Encryption, implemented in the tape drive, encrypts the data before it is written to the cartridge. When tape compression is enabled, the tape drive first compresses the data to be written and then encrypts it. This method means no loss of capacity with IBM tape data encryption. If the encryption solution encrypts the data first and then tries to compress the data, the encrypted data usually compresses very little, if at all.

To encrypt the data, the tape drive requires a key to use. This key is provided by the Encryption Key Manager in an encrypted form to make the tape data encryption solution secure.

Figure 1-3 summarizes the process for tape data encryption using TS1130 and TS1120.

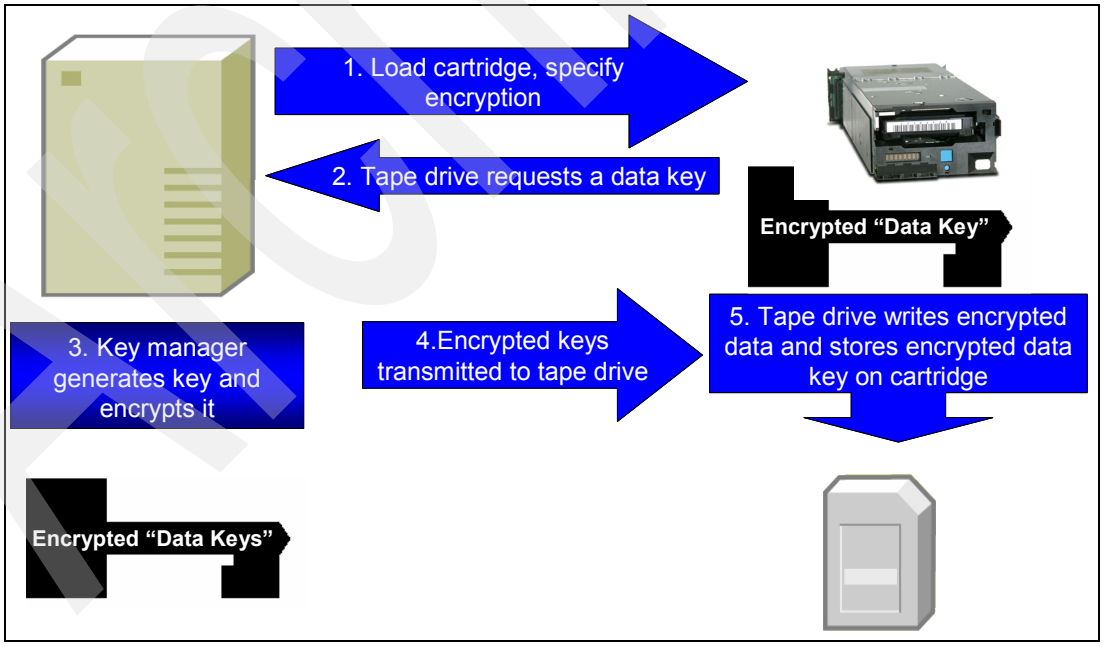


Figure 1-3 TS1120 and TS1130 tape data encryption process flow

Figure 1-4 on page 6 summarizes the LTO4 tape data encryption process flow.

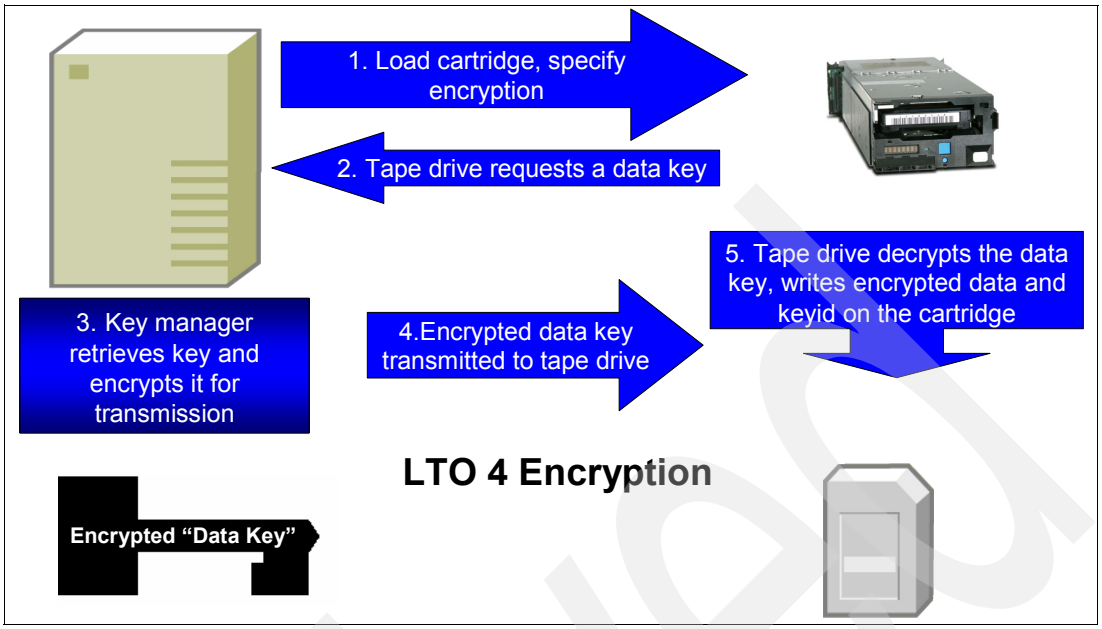


Figure 1-4 LTO4 tape data encryption process

For a detailed description of these processes, refer to Chapter 2, "IBM tape encryption methods" on page 23.

1.2 What to encrypt

Since 2005, over 250 million consumers have been notified of potential security breaches regarding personal information. The source for this information is:

http://www.Privacyrights.org

The loss of computer backup tapes is one type of event that triggers consumer notification. This has led to increasing data protection requirements as indicated in Figure 1-5.

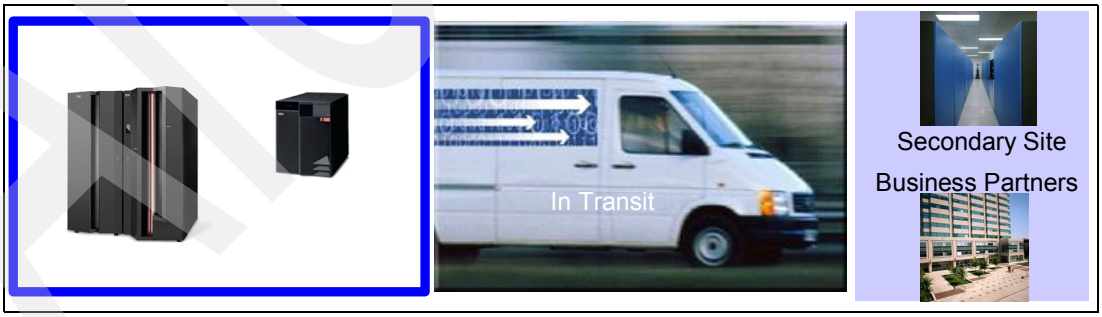


Figure 1-5 Client data protection requirements

What should you encrypt, and just as important, what should you not encrypt? The focus on data security is an ever-increasing:

- California is generally considered the first state to implement a law requiring disclosure of security breaches (July, 2003).
- Legislation has been enacted by 38 states that requires notification in cases of security breaches. The source for this is:

  http://www.Privacyrights.org

- Similar federal legislation has been proposed. The source for this is:

  http://www.epic.org/privacy/bill_track.html

Data protection requirements are driven by a variety of reasons. In addition to regulatory requirements that are driving the need for greater data security, integrity, retention, auditability, and privacy, the reasons for increasing data protection are:

- Severe business impacts that might be caused by loss or theft of data, including financial liability, reputation damage, legal risk, and compliance risk
- Requirement to share data securely with IBM Business Partners and maintain backups at remote locations is increasing
- Requirement to reduce complexity and improve processes around enterprise encryption management is increasing
- Requirement to cost-effectively encrypt large quantities of tape data

The additional drivers for encryption in financial services are:

- A riskier environment

  Internet Banking, for example, relies on open networks with multiple access points to conduct business in real time to drive down costs and improve response times to revenue generating opportunities.

- Growing regulatory burden, such as:
  - Gramm-Leach-Bliley Act (GLBA) of 1999
  - California Law No. SB 1386
  - FCRA/FACTA amendments
  - Basel II

  Not all of the regulations specifically require the use of stored data encryption. However, many organizations are implementing encryption for their protected information in conjunction with other security layers to protect Personally-Identifiable Information.

- Maturing industry standards, such as the Payment Card Industry (PCI) Data Security Standard (DSS)

In summary, simply encrypt everything that you can encrypt and still be able to recover data in event of a disaster. As long as system data can be separated from application data, encrypting everything with no performance impact is easier than choosing which data falls into which legislation for encryption, and trying to keep current on the dynamic privacy rights rules and regulations.

1.3 Why use tape data encryption

Tape data encryption is used to hide and protect sensitive data. If tape data on cartridges leaves data centers, the data is no longer protected through Resource Access Control Facility (RACF®) or similar access protection mechanisms. Tape data encryption can help fulfill

security regulations. Many governmental agencies require disclosure of security breaches. Industry organizations are also increasing their scrutiny of security procedures. Tape data encryption uses an easy and economical way to protect data from unauthorized view.

Important and sensitive data can be protected in many ways. Data can be encrypted by means of special software programs, hardware adapters, facilities, or outside of the device where the data is stored. Encrypting data with software programs takes away processor power, and encrypting data with hardware requires additional investment in hardware for the computers.

The advantage of IBM tape data encryption is that the data is encrypted after compression, and there are no additional software program costs. IBM tape data encryption saves space on tape cartridges and saves additional hardware investments. In addition, outboard encryption in the tape drives might help you protect large volumes of tape data in a cost-effective way. Data on cartridges does not have to be *degaussed* or overwritten with patterns of x'FF' at the end of life of the cartridge. This approach is valid for Write Once Read Many (WORM) cartridges and for normal cartridges.

The encryption key management capability is designed to manage keys across mainframes and Open Systems environments. There is only one component to be managed across multiple platforms.

Tape data encryption can be managed by the applications, or can be system-managed or library-managed. The following chapters discuss the prerequisites necessary to prepare the hardware levels and required software levels. After you complete the preparation steps, you can start the encryption of tape data very quickly.

Additionally, a clever use of encryption is for data shredding. In effect, if you delete an encryption key, all of the data that the encryption key protected, becomes garbage. This use of cryptography requires extreme care to know exactly what data belongs to what key.

1.3.1 Why encrypt data in the drive

The IBM tape-drive encryption solution encrypts the data within the drive using the 256-bit Advanced Encryption Standard (AES) algorithm, rather than receiving previously encrypted data. This system offers several advantages. By encrypting data in the drive, the drive can offer the most efficient data compression, because the drive first compresses the data, then encrypts it, providing more efficient data storage and media usage.

Encrypting in the drive also eliminates having to use additional machines or appliances in the environment by offloading the encryption processing overhead onto the drive. Because the drive can also process unencrypted workloads, the IT environment is further simplified by eliminating the need for separate drives to process data that does not have to be encrypted.

1.3.2 Fundamental to encryption: Policy and key management

Tape drive-based encryption using keys is only part of the solution. A complete solution must also address encryption policy and key management**.** IBM recognizes that policy and key management can vary depending on the environment, and as a result, IBM has developed a flexible solution that allows you to tailor the implementation to your unique environment.

The IBM solution provides policy options at three levels:

- Application layer
- System layer
- Library layer

IBM supports two methods for managing the encryption keys: through the application (in Open Systems) or through a new key manager program called the Tivoli Lifecycle Key Manager. Additionally the previous generation key manager called the Encryption Key Manager (EKM) is still available. The policy implementation also depends on the environment. For example, in a z/OS environment, the encryption policies can be managed by Data Facility Storage Management Subsystem (DFSMS) structures; however, in the Open Systems environments, the policy granularity is based on other methods, such as by drive or by a range of volume serial numbers on cartridges in a library.

1.3.3 Summary

Encryption capability that is provided as a standard feature in the IBM TS1130, IBM TS1120 Tape Drive or the IBM LTO4 Tape Drive makes encrypting data stored on tape cartridges much easier. This capability is increasingly important as legislation continues to grow, requiring the notification of individuals when their personal information has potentially been compromised. The tape drive-based encryption solutions developed by IBM and described here, coupled with the new Encryption Key Manager component, enable key management and encryption in a wide variety of environments.

IBM provides tape drive encryption support in a range of operating systems environments:

- z/OS
- z/VM®
- z/VSE™
- z/TPF
- i5/OS®
- AIX®
- Linux® on System z
- Linux on other platforms
- HP-UX
- Sun™ Solaris™
- Windows® Server 2000, Windows 2003, or Windows 2008 Server

Support is described in detail in the following chapters.

All statements regarding IBM plans, directions, and intent are subject to change or withdrawal without notice. Any reliance on these statements of general direction is at the relying party's sole risk and will not create liability or obligation for IBM.

1.4 Concepts of tape data encryption

In this section, we discuss basic encryption, cryptographic terms, and ideas. Encryption has been used to exchange information in a secure and confidential way for many centuries. Encryption transforms data that is unprotected, or *plain text*, into encrypted data, or *ciphertext,* by using a *key*. It is very difficult to "break" ciphertext in order to change it back to the clear text without the associated encryption key.

Computer technology has enabled increasingly sophisticated encryption algorithms. Working with the U.S. Government National Institute of Standards and Technology (NIST), IBM invented one of the first computer-based algorithms, Data Encryption Standard (DES), in 1974. With the advances in computer technology, DES is now considered obsolete. Today, there are several widely used encryption algorithms, including Triple DES (TDES) and Advanced Encryption Standard (AES).

Early encryption methods used the same key to encrypt clear text to generate cipher text and to decrypt the cipher text to regenerate the clear text. Because the same key is used for both encryption and decryption, this method is called *symmetric encryption*. All of the encryption algorithms previously mentioned use symmetric encryption.

It was only in the 1970s that cryptographers invented asymmetric key algorithms for encryption and decryption. These algorithms use different keys for encryption and decryption. The keys are mathematically related, but deriving one key from the other key is practically impossible. Encryption methods using different keys for encryption and decryption are called *asymmetric encryption.*

Asymmetric encryption addresses certain drawbacks of symmetric encryption, which became more important with computer-based cryptography. We discuss this in detail in the following two sections about symmetric and asymmetric key encryption.

The IBM tape data encryption solution uses a combination of symmetric and asymmetric encryption methods.This combination of symmetric and asymmetric encryption algorithms is prevalent in many security solutions, including TLS, IPSEC, Kerberos.

1.4.1 Symmetric key encryption

Symmetric key encryption uses identical keys - or keys that can be related through a simple transformation - for encryption and decryption. Everyone who gets knowledge of the key can transform the ciphertext back to plain text. If you want to preserve confidentiality, you must protect your key and keep it a secret. Therefore, symmetric encryption is also called *private* or *secret key encryption*, which is not to be confused with the private key in an asymmetric key system.

In Figure 1-6 on page 11, we show a sample encryption and decryption data flow path. Here, we use the symmetric key AES_256_ITSO to encrypt plain text using the AES encryption algorithm, which yields encrypted data. The decryption of the enciphered text uses the same AES_256_ITSO symmetric key and the AES algorithm to decrypt the data back to its plain text format.

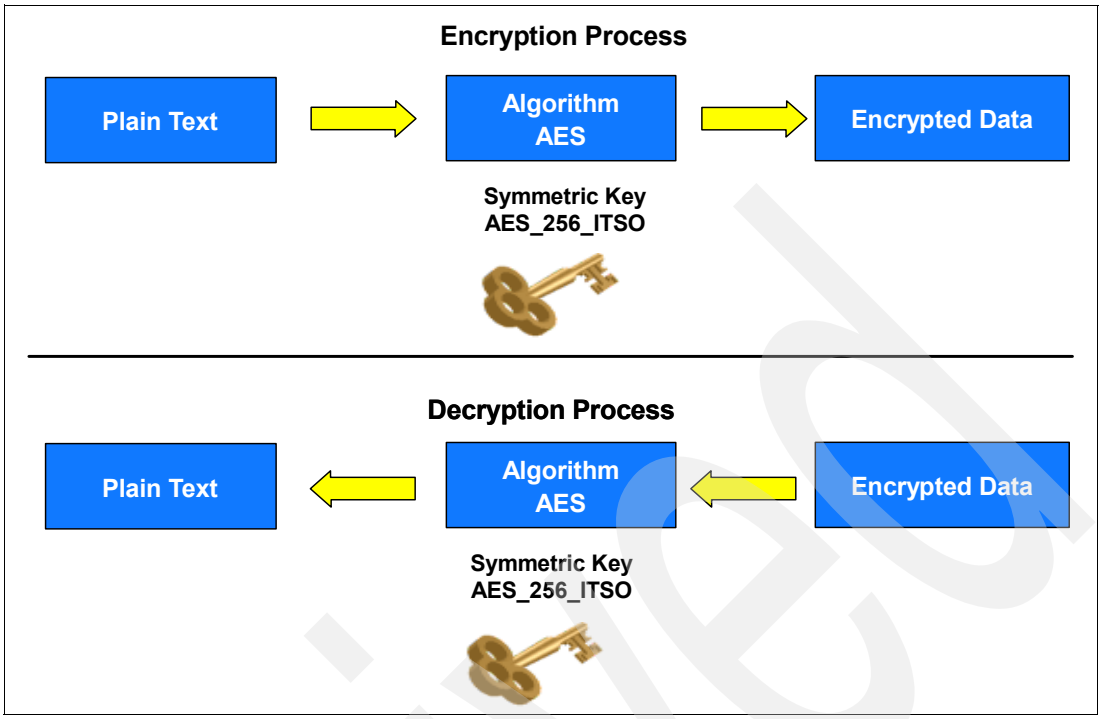


Figure 1-6 Symmetric key encryption

Symmetric key encryption algorithms are significantly faster than asymmetric encryption algorithms, which makes symmetric encryption an ideal candidate for encrypting large amounts of data.

In addition, the comparable key sizes for symmetric encryption as opposed to asymmetric encryption are significantly different. At the time of writing this book, a 128-bit secret key is used for symmetric AES encryption, and the Rivest-Shamir-Adleman (RSA) encryption algorithm suggests a 1024-bit key length.

Secret key algorithms can be architected to support encryption one bit at a time or by specified blocks of bits. The AES standard supports 128-bit block sizes and key sizes of 128, 192, and 256 bits. The IBM tape data encryption solution uses an AES-256 bit key.

Other well-known symmetric key examples include Twofish, Blowfish, Serpent, Cast5, DES, TDES, and IDEA.

Speed and short key-length are advantages of symmetric encryption, but there are two drawbacks, which are the way that keys are exchanged and the number of keys required.

Secure exchange of keys has always been a problem with symmetric encryption. The sender and the recipient have to share a common, secret key. The sender of a confidential message must make sure that no one other than the intended recipient gets knowledge of the key. So, the sender has to transfer the key to the recipient in a secure way, for example, in a face-to-face meeting, through a trusted courier, or a secure electronic channel. This method of transferring keys might work as long as only few people are involved in the exchange of confidential information. When a larger number of people have to exchange keys, the distribution of secret keys becomes difficult and inefficient with this method.

The second drawback of symmetric encryption is the large number of required keys. When a group of people are to exchange symmetrically encrypted information, each possible pair of two people in this group has to share a secret key. The number of required keys grows very

fast with the number of people in the group. The number of keys required in relation to the number of people can be calculated with the following formula, where *k* is the number of keys, and *n* is the number of people:

$k_n = n(n-1)/2$

As you can see in Figure 1-7, the number of required keys grows very fast. For a group of 100 people, 4,950 different keys are required. A group of 1,000 people requires 499,500 keys. Key distribution and key management are challenges.

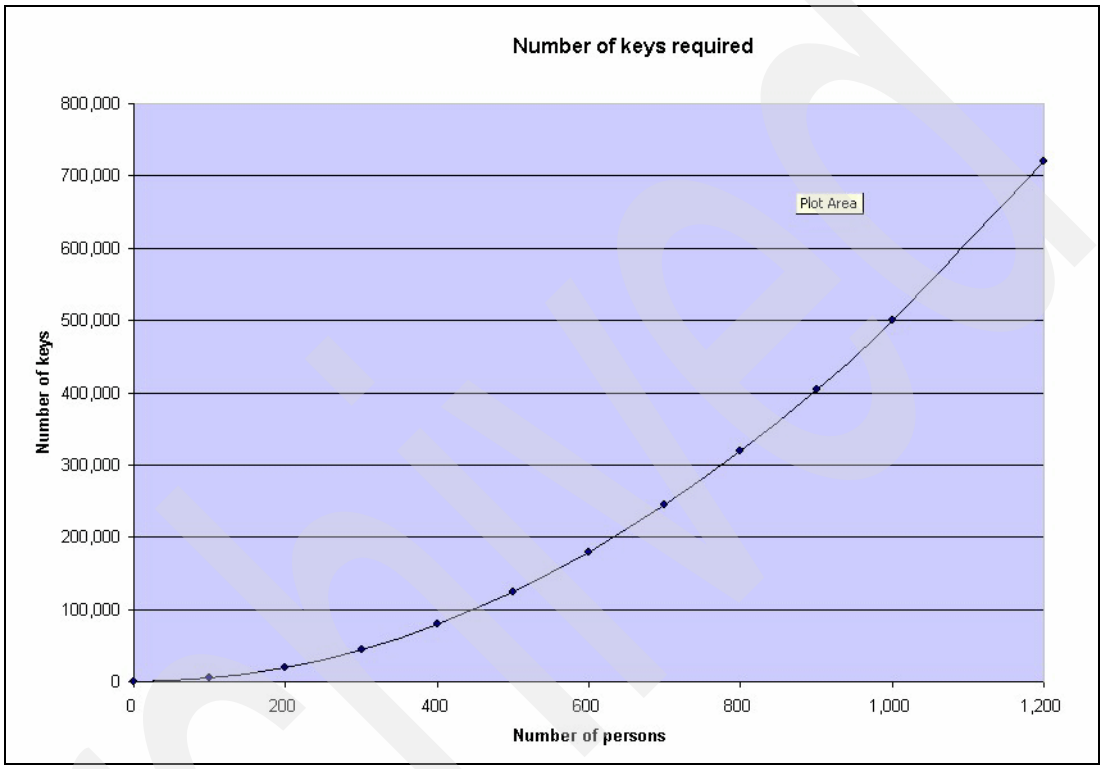


Figure 1-7 Number of keys required for symmetric encryption

The IBM tape data encryption solution utilizes an AES algorithm with a key length of 256 bits for the encryption on the tape drive. The AES algorithm is based on the Rijndael algorithm. AES is an accepted standard that supports a subset of the key sizes and block sizes that the Rijndael algorithms supports.

Note: The Rijdindael algorithm supports block sizes of 128, 160, 192, 224, and 256 bits. It supports key sizes of 128, 160, 192, 224, and 256 bits.

The shortcomings of symmetric encryption in terms of key distribution and key management are addressed by asymmetric key encryption, which we describe in the next section.

1.4.2 Asymmetric key encryption

Asymmetric key encryption method uses key pairs for encrypting and decrypting data. One key is used to encrypt the data, and the other key is used to decrypt the data. Because the key used for encrypting a message cannot be used for decrypting it, this key does not have to be kept a secret. It can be widely shared and is therefore called a *public key.* Anyone who

wants to send secure data to an organization can use its public key. The receiving organization then uses its *private key* to decrypt the data. The private key is the corresponding half of the public-private key pair and must always be kept a secret. Because asymmetric encryption uses public-private key pairs, it is also called *public-private key encryption* or *public key encryption.*

Public-private key encryption is useful for sharing information between organizations and is widely used on the Internet today to secure transactions, including Secure Sockets Layer (SSL).

The concept of asymmetric encryption is relatively new. For centuries, cryptographers believed that the sender and the recipient had to share the same secret key. In the early 1970s, British cryptographers Ellis, Cocks, and Williamson discovered a way to use different keys for encrypting and decrypting data. Because they were working for GCHQ, a British intelligence agency, their findings were kept secret until 1997. In 1976, Whitfield Diffie and Martin Hellman invented a solution to the problem, which has since become known as Diffie-Hellman key exchange. In 1977 Ron Rivest, Adi Shamir, and Leonard Adleman published an algorithm for public-key encryption.

Well-known examples of asymmetric key algorithms are:

- RSA
- Diffie-Hellman
- Elliptic curve cryptography (ECC)
- ElGamal

Today, the Rivest-Shamir-Adleman (RSA) algorithm is the most widely used public key technique.

Note: RSA uses *trapdoor functions*. Trapdoor functions are mathematical functions that are easy to compute in one direction, but are difficult to compute in the reverse direction without additional information. This additional information is called the trapdoor. In the case of RSA, the private key is the trapdoor.

The advantage of asymmetric key encryption is the ability to share secret data without sharing the same encryption key. But there are disadvantages, too. Asymmetric key encryption is computationally more intensive and therefore significantly slower than symmetric key encryption. In practice, you will often use a combination of symmetric and asymmetric encryption. We describe this method in 1.4.3, “Hybrid encryption” on page 15.

The IBM tape data encryption solution utilizes the asymmetric RSA algorithm to encrypt symmetric AES keys.

Digital Signature

You can use public-private key pairs to protect the content of a message, and also to digitally sign a message. For example, if Tony wants to send JoHann a digitally signed message, Tony will not use JoHann’s public key to encrypt the message, but Tony’s own private key. The content of the encrypted message is not protected, because anyone can decrypt the message by using Tony’s public key. But, if JoHann is able to decrypt Tony’s message with Tony’s public key, JoHann can be sure that Tony sent the message. JoHann has proof that the message was encrypted with Tony’s private key and JoHann knows that only Tony has access to this key.

In practice, predominantly for efficiency reasons, a hash value of the message is signed rather than the whole message, but the overall procedure is the same.

In the previous example, JoHann has to make sure that Tony's public key really belongs to Tony, and not to someone pretending to be Tony. If JoHann cannot confirm this himself, JoHann will need a trusted third party to verify Tony's identity. A *certificate* issued and signed by a *Certification Authority* (CA) can confirm that the public key belongs to Tony. A certificate binds together the identity of a person or organization and its public key. If JoHann trusts the CA, JoHann can be sure that it really was Tony who sent the message.

We discuss certificates in detail in 1.4.4, "Digital certificates" on page 16.

Of course, you can combine public key encryption and digital signature to produce a message that is both encryption-protected and digitally signed.

Example of public-private key encryption

Figure 1-8 shows an encryption and decryption data path when using public key encryption algorithms. In the diagram, the plain text is enciphered using the public key and an RSA encryption algorithm, which yields the encrypted data.

Starting with the enciphered text, a private key is used, with the RSA algorithm to decrypt the data back to plain text.

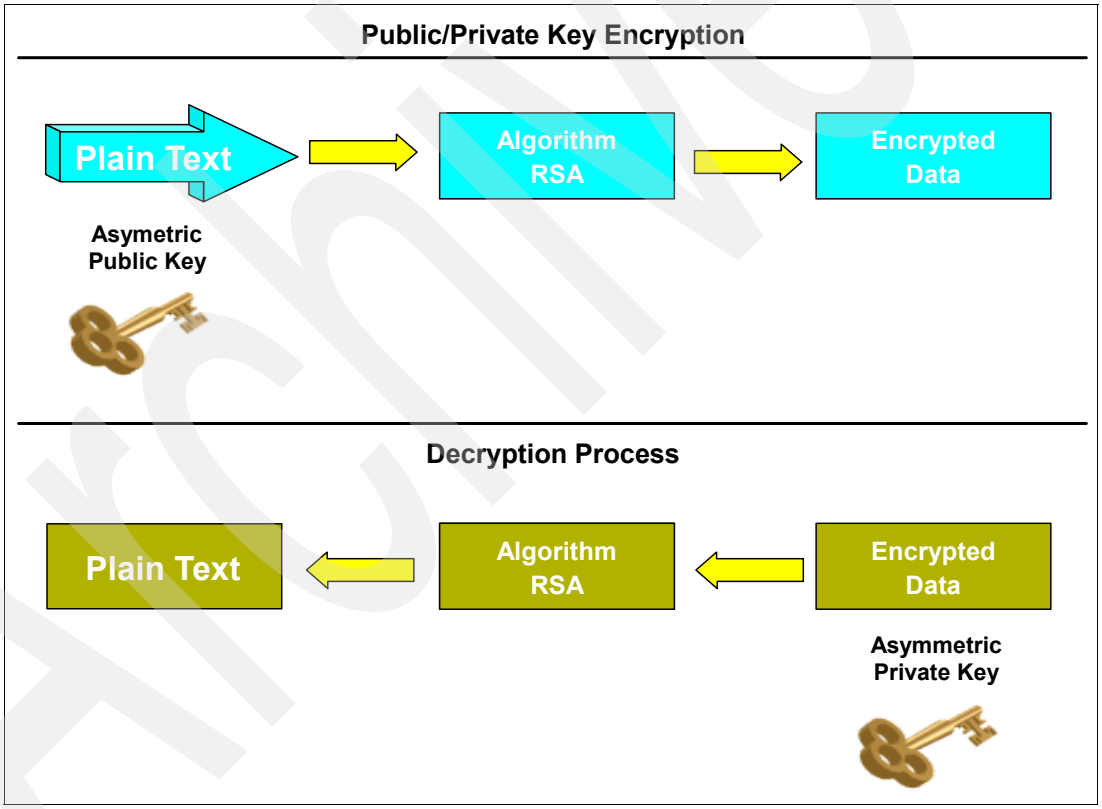


Figure 1-8 Public-private key encryption

In Figure 1-9 on page 15, we show a more complicated example of data protection and sharing using an asymmetric key pair. In this example, Tony has a private key, and JoHann has a copy of Tony's public key. Tony sends JoHann a message that is encrypted with Tony's private key. JoHann then uses the public key to decrypt the message. When the message is decrypted to clear text, this proves to JoHann that he is in fact communicating with Tony, because only Tony has a copy of the private key. JoHann then public-key encrypts the data that he wants to protect and sends it to Tony. Tony can use his private key to decrypt the data.

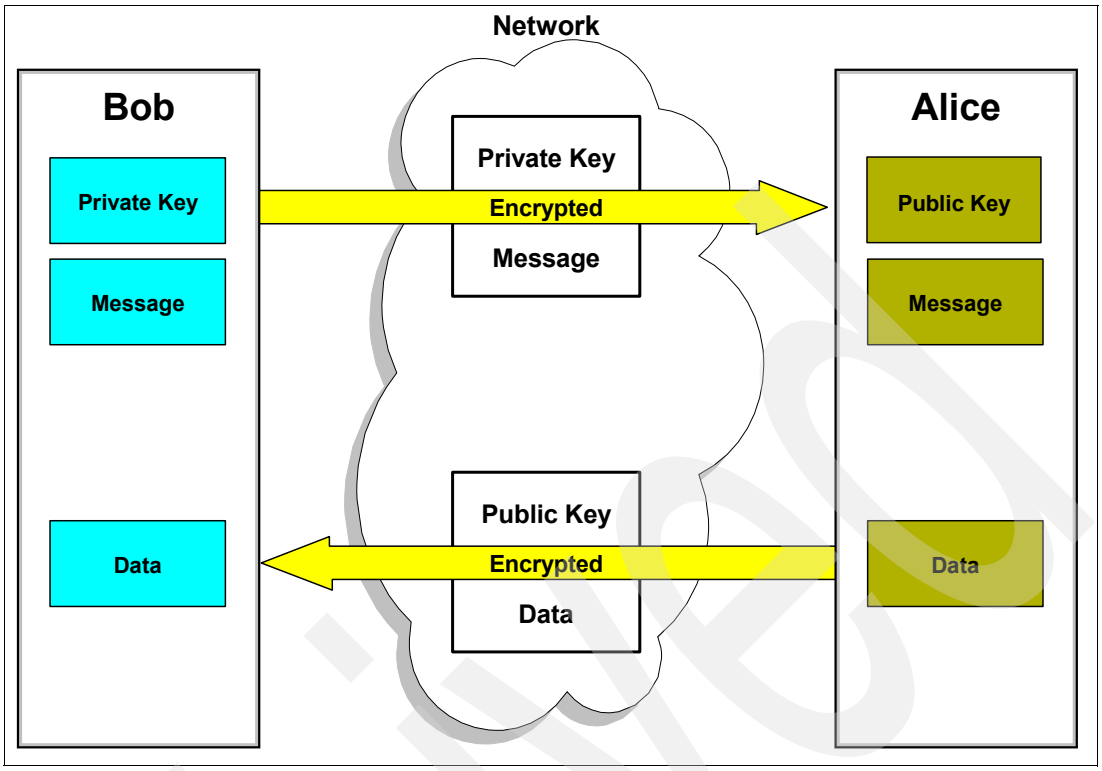


Figure 1-9 Identity verification using public-private key encryption

Both asymmetric and symmetric key encryption schemes are powerful ways to protect and secure data. In 2.3, “Methods of managing IBM tape encryption” on page 32, we discuss how to use these schemes in conjunction for the IBM tape data encryption solution to give us an extremely secure way of protecting data.

1.4.3 Hybrid encryption

In practice, encryption methods often combine symmetric and asymmetric encryption. Thus, they can take advantage of fast encryption with symmetric encryption and still securely exchange keys using asymmetric encryption.

Hybrid methods use a symmetric data key to actually encrypt and decrypt data. They do not transfer this symmetric data key in the clear, but use public-private key encryption to encrypt the data key. The recipient is able to decrypt the encrypted data key and use the data key to encrypt or decrypt a message.

Hybrid encryption methods allow you to combine secure and convenient key exchange with fast and efficient encryption of large amounts of data.

The IBM tape data encryption solution uses a symmetric AES data key to encrypt and decrypt data. This data key is protected by the asymmetric RSA algorithm and is not available in the clear when tape drives and the Enterprise Key Manager (EKM) or Tivoli Key Lifecycle Manager (TKLM) communicate. For details about EKM, refer to 2.1, “IBM Encryption Key Manager” on page 24. For details about TKLM, refer to 2.2, “Tivoli Key Lifecycle Manager” on page 29.

Application-Managed Encryption does not use asymmetric encryption. Refer to 2.3.4, “Application-Managed Encryption” on page 40 for more information.

1.4.4 Digital certificates

Digital certificates are a way to bind public key information with an identity. Part of the information that is stored in a digital certificate is:

- Name of the issuer
- Subject Distinguished Name (DN)
- Public key belonging to the owner
- Validity date for the public key
- Serial number of the digital certificate
- Digital signature of the issuer

In this section, we discuss the X.509 Public Key Infrastructure (PKI), certificate chains, certificate request, and certificate responses. X.509 is a well established and accepted standard for certificate management.

In Figure 1-10 on page 17, we have an abstract simplified version of part of the process of a self-signed certificate. It shows that both the issuer and subject of the certificate are IBM. This certificate has a public key, a private key, and a public key that is *signed* by the private key of this certificate. Data can be encrypted using a public key, which can then be decrypted by a private key. This situation means that only the entity with the private key can decrypt the data and ensures that only the entity for whom the data is intended can decrypt it.

When the private key is used to encrypt data, additional aspects must be considered. In this case, we have a copy of the public key as clear text, and a copy that is encrypted by our private key. This case means that *anyone* with a copy of our freely shared public key can decrypt the data.

This approach means that when we send copies of our public key out in a certificate format, the entity receiving the certificate can verify that the public key they were sent was sent by us, was not intercepted in transit, and was not tampered with.

Because we have the only copy of our private key, we are the only entity that can encrypt a copy of the public key in the certificate. If the entity uses our public key to decrypt the enciphered copy of the public key in the certificate, if the decrypted public key matches the clear public key, and if the owners of the public key trust that only we have our private key, they know that when they use that public key to encrypt data, we are the only entity with the capability to decrypt it. Figure 1-10 on page 17 shows a sample digital certificate.

In general, using a public key to encrypt data secures that data, ensuring confidentiality. When using a private key to encrypt data, the following is true:

- Identity proof
- Message integrity
- Non-repudiation

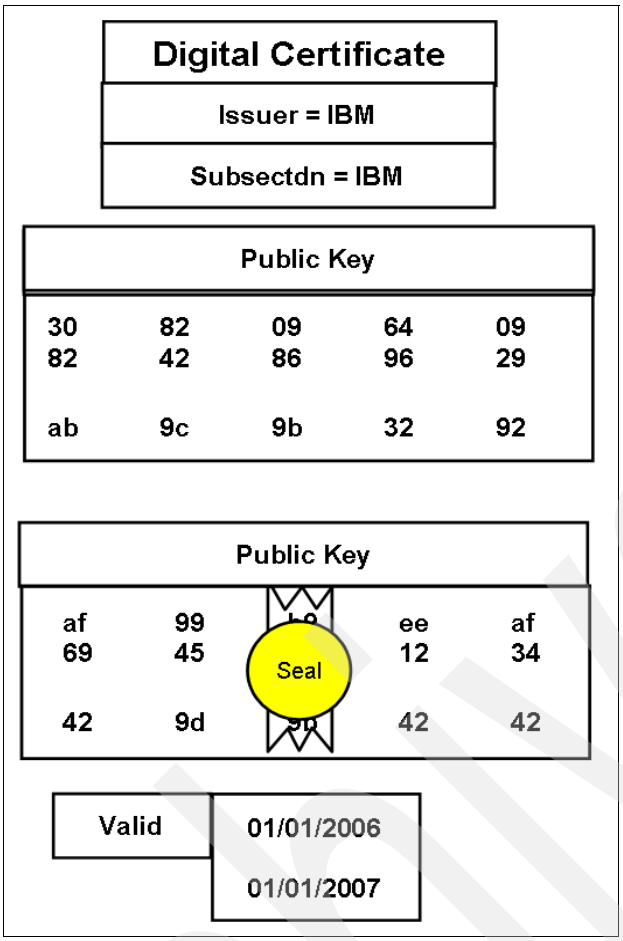


Figure 1-10 Sample digital certificate

When sending information that was private key-encrypted, the receiver of the message knows that the message must have been sent by the entity with the private key; the receiver also can verify that the message was not tampered with. Finally, the entity receiving a message that was private key-encrypted knows that the message that they got cannot be denied by the sender. In other words, because only the sender has the private key, the sender must have sent it.

Certificate authorities

A *certificate authority (CA)* is a company that holds and makes available trusted certificates. Companies can send certificates to a CA to be added to the chain of trust. As long as a company trusts the CA, certificates that are issued by that CA can be trusted.

For example, Figure 1-11 on page 18 describes what company ZABYXC does to generate a certificate request to the JohannTonyArtCA third-party certificate authority (CA) company. In the figure, we see that company ZABYXC already trusts JohannTonyArtCA, because ZABYXC has a copy of the JohannTonyArtRootCA in its certificate repository. This copy of JohannTonyArtRootCA has only the public key and an encrypted copy of the public key, which is encrypted with JohannTonyArtRootCA's private key.

Company ZABYXC also has a self-signed personal certificate with a public and a private key associated with it. Using certificate managing tools, company ZABYXC exports a copy of its self-signed personal certificate that includes only the certificate information, the public key, and the encrypted version of the public key.

This certificate request is sent to JohannTonyArtCA.

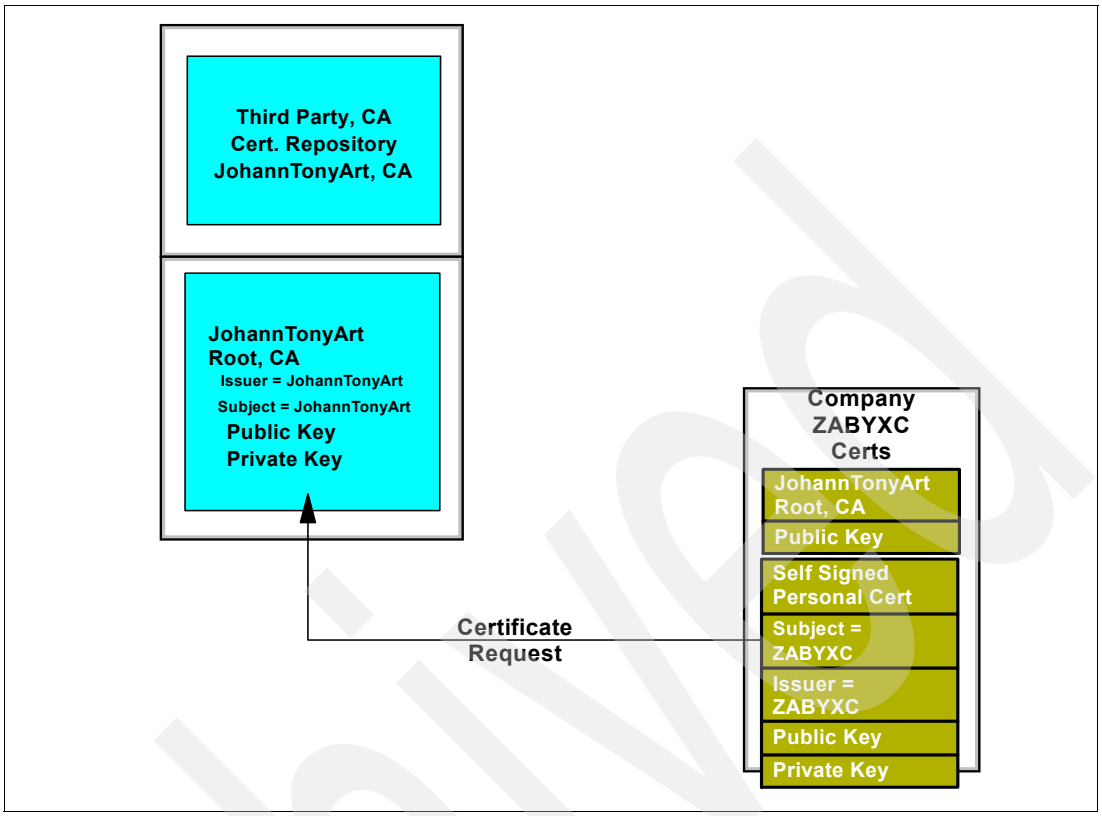


Figure 1-11 Certificate request

In Figure 1-12 on page 19, JohannTonyArtCA receives the certificate response from company ZABYXC. JohannTonyArtCA then uses the private key from JohannTonyArtRootCA to encrypt a copy of the certificate request's public key and attaches both the clear public key and the new encrypted copy of the public key to a certificate response. In addition, the certificate response has the issuer changed to JohannTonyArtCA. This response is sent to company ZABYXC.

When Company ZABYXC receives the certificate response from JohannTonyArtCA, Company ZABYXC imports the certificate into the company's certificate repository. The company replaces the self-signed personal certificate in the repository, and it keeps the private key previously associated with the personal certificate.

Company ZABYXC can verify that the certificate response came from JohannTonyArtCA, because they have a copy of JohannTonyArtRootCA. They can use the public key from JohannTonyArtRootCA to verify that the certificate response came from JohannTonyArtCA.

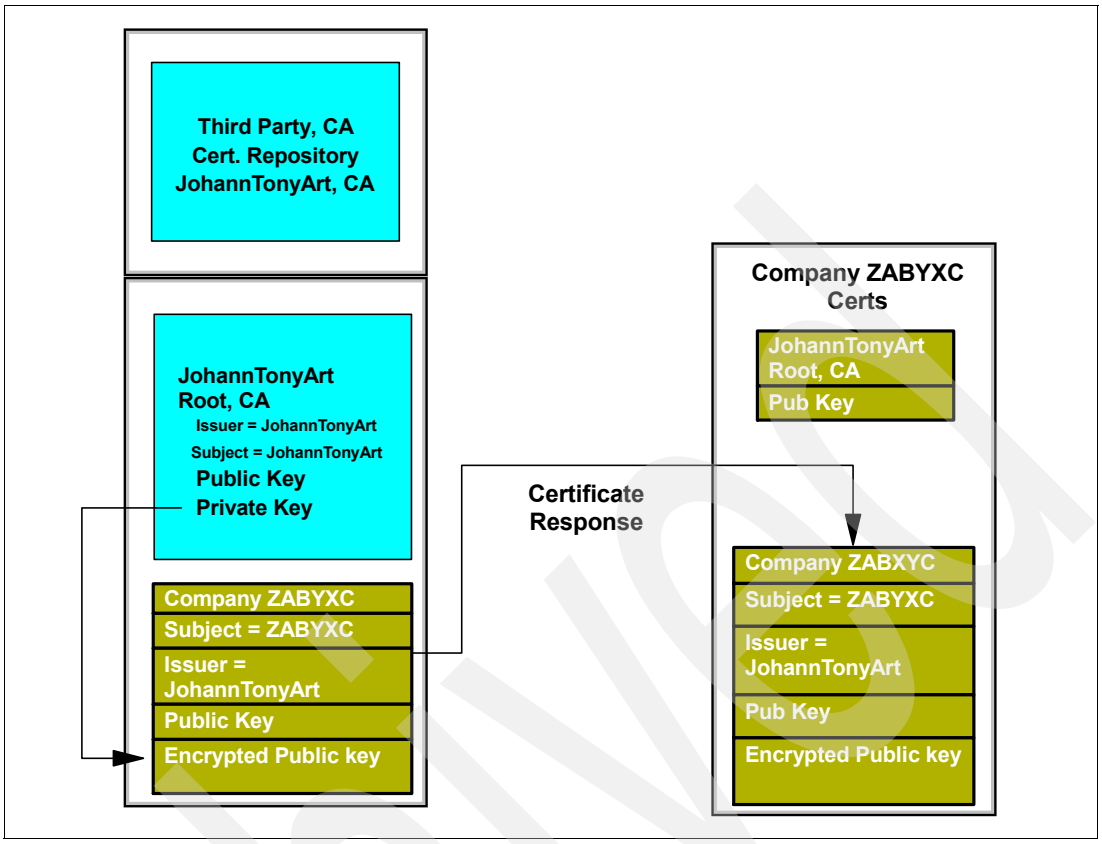


Figure 1-12 Certificate response

If company ZABYXC wants another company to share data with it, the company can now export a copy of its personal certificate, which contains its public key, and the public key signed by JohannTonyArtRootCA. When ZABYXC sends this certificate to a business partner, the business partner can add JohannTonyArtRootCA to its own certificate repository and then use that to verify the personal certificate sent to it by Company ZABYXC.

Having an extended certificate chain when dealing with PKI is possible. In a longer certificate chain, the JohannTonyArtRootCA is the root CA with a validity of several years. Next in the chain is a ZABYXCCA signed by the JohannTonyArtRootCA. This certificate can have a shorter validity period and might have to be re-requested. The third-party CA keeps the private key information for these certificates. When company ZABYXC generates a certificate request in this situation, it receives a certificate response signed by company ZABYXCCA.

Figure 1-13 on page 20 shows an extended certificate chain. To verify certificate validity in this situation, the whole chain has to be in the certificate repository.

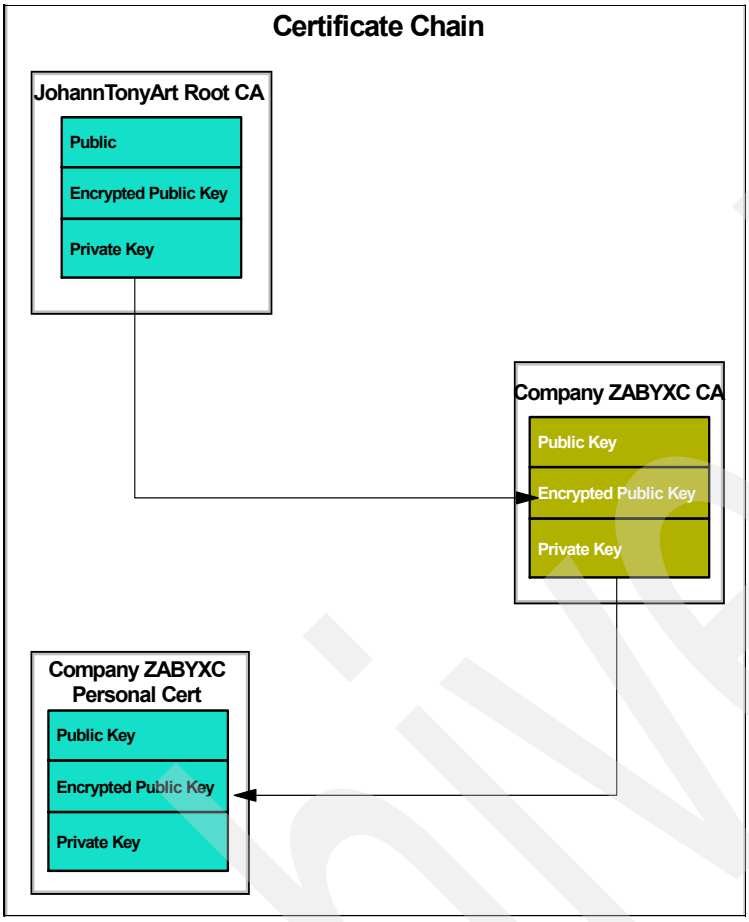


Figure 1-13 Certificate chain

Secure Sockets Layer example

Figure 1-14 on page 21 shows a simple Secure Sockets Layer (SSL) handshake with ClientAuthentication required. When more than one Encryption Key Manager (EKM) is used in the IBM tape data encryption solution, the primary and secondary EKMs can synchronize information using an SSL connection. The default SSL setup for the EKM is to not do ClientAuthentication.

In the first portion of the handshake, the client sends a “Hello” message to the server; the server responds by sending its own “Hello” message back and sending its trusted certificate. If the client finds that it does indeed have a trusted certificate entry to verify that the server is in fact the correct server, the handshake continues. In this example, we perform ClientAuthentication, which causes the server to send a certificate request to the client. After this step, the server “Hello” response is completed.

The next portion of the handshake is related to the ClientAuth value set to true. Here, the client sends its certificate to the server, and if the server finds that it has the matching certificate entry in its truststore, the handshake can continue, because the client’s identity is verified.

After the client and the server have verified that they are indeed who they claim to be, they exchange keys and decide with which SSL cipher to communicate. Data can then be encrypted and sent across the network. Not only does SSL allow data communication to be protected between a client and server, it also is used to prove client and server identities.

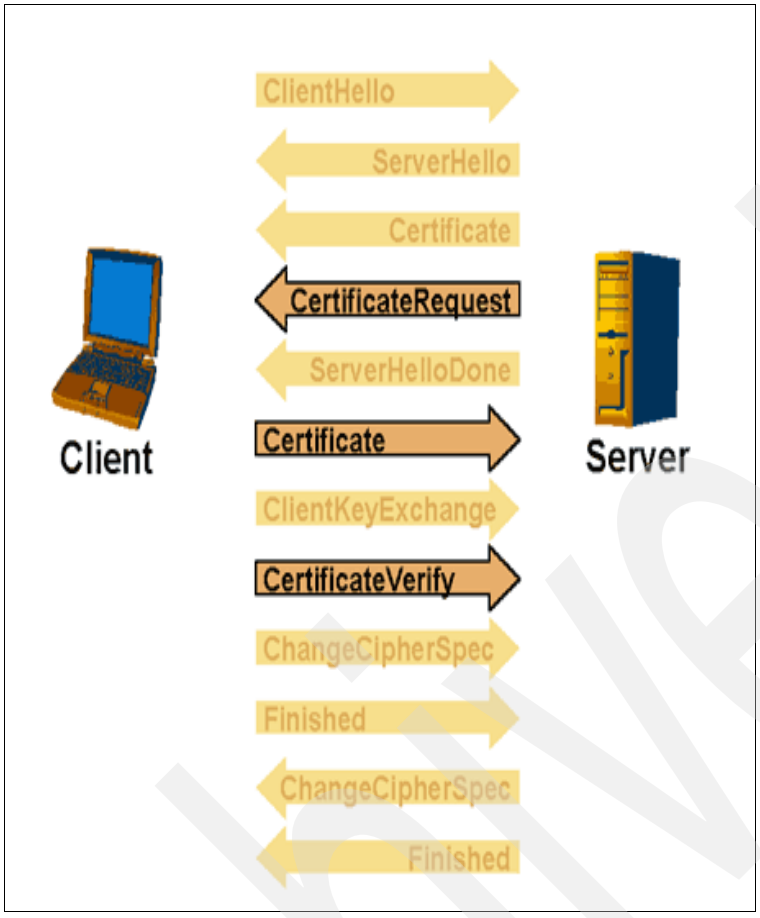


Figure 1-14 SSL Handshake example using ClientAuth

2

IBM tape encryption methods

In this chapter, we discuss the *Tivoli Key Lifecycle Manager* (TKLM) and the *Encryption Key Manager* (EKM), which are both Java software programs that manage keys enterprise-wide and provide encryption-enabled tape drives with keys for encryption and decryption.The TKLM is an IBM product that adds key management functionality over the EKM.

We describe various methods of managing IBM tape encryption. These methods differ in where the encryption policies reside, where key management is performed, whether a key manager is required, and, if a key manager is required, how the tape drives communicate with it.

IBM supports three methods of encrypting data on tape:

- System-Managed Encryption (SME)
- Library-Managed Encryption (LME)
- Application-Managed Encryption (AME)

Only two of these methods, SME and LME, require the implementation of an external component, the TKLM, to provide and manage keys. With AME, key provisioning and key management are handled by the application.

When describing the encryption methods, we trace the flow of data and keys. We explain how the tape drive communicates with the EKM or the application and how symmetric keys and asymmetric keys are transferred to the drive. For AME, we describe how the application communicates with the tape drives.

In each section, we briefly discuss the criteria that can influence your decision for or against a specific encryption method. For more information, refer to Chapter 4, “Planning for software and hardware” on page 91).

2.1 IBM Encryption Key Manager

In your enterprise, a large number of symmetric keys, asymmetric keys, and certificates can exist. All of these keys and certificates need to be managed. Key management can be handled either internally by an application, such as Tivoli Storage Manager, or externally by an Encryption Key Manager (EKM). In this section, we discuss the EKM.

LTO4, like the encryption-capable 3592 drives TS1120 and TS1130, provides three methods of encryption management from which to choose. These methods differ in where you choose to locate your EKM application. Your operating environment determines which is the best method for you, with the result that key management and the encryption policy engine might be located in any one of the following three environmental layers as shown in Figure 2-1.

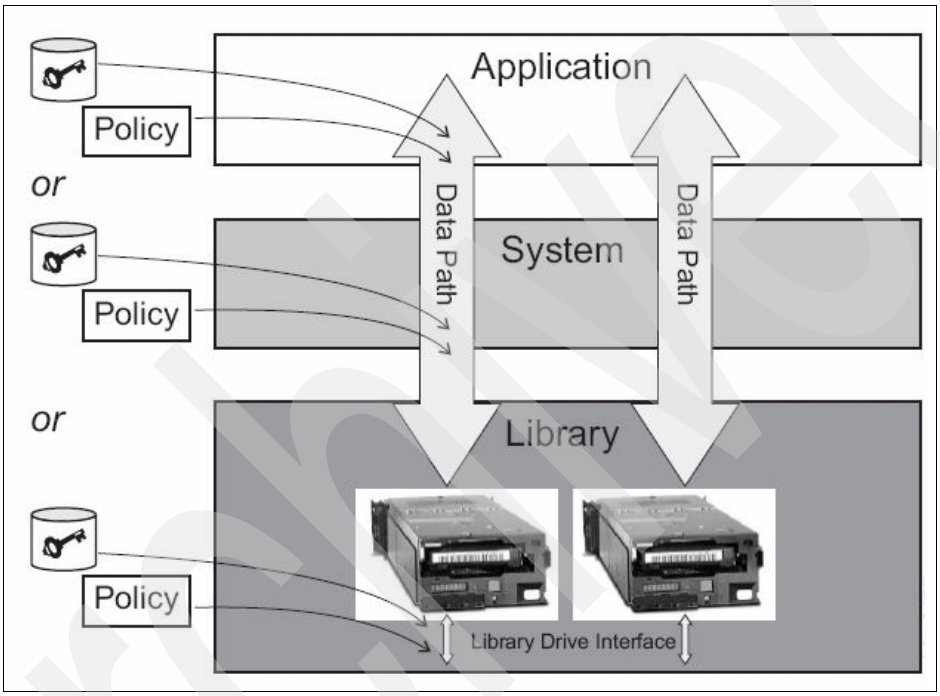


Figure 2-1 EKM architecture

The IBM Encryption Key Manager component for the Java platform is a Java software program that assists IBM encryption-enabled TS1120 Tape Drives and Linear Tape-Open (LTO) Ultrium 4 Tape Drives by providing, protecting, storing, and maintaining encryption keys that are used to encrypt information being written to, and decrypt information being read from, tape media. EKM operates on a variety of operating systems. Currently, the supported operating systems are:

- z/OS
- i5/OS
- AIX
- Linux
- Hewlett-Packard UNIX® (HP-UX)
- Sun Solaris
- Windows

EKM is designed to be a shared resource deployed in several locations within an enterprise. It 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).

IBM supplies the EKM, free of charge on IBM operating systems. On platforms that are not IBM platforms, TPC-BE must be purchased to gain access to the EKM. For IBM operating systems, download the current version of the IBM Encryption Key Manager for the Java platform, the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User' Guide,* GA76-0418, and a sample configuration file for EKM. To download, use either of the following methods:

- Go to:

  `http://www.ibm.com/support/search.wss?rs=1139&tc=STCXRGL&dc=D400&dtm`

- Go to the IBM Web site home page:

  `http://www.ibm.com`

2.1.1 Encryption Key Manager components and resources

The sole task of the Encryption Key Manager is to handle serving keys to the encrypting tape drives. The EKM does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. To perform these tasks, EKM has to rely on external components. In the following sections, we describe the components of EKM and resources that are used by EKM.

Figure 2-2 shows the EKM components and external resources.

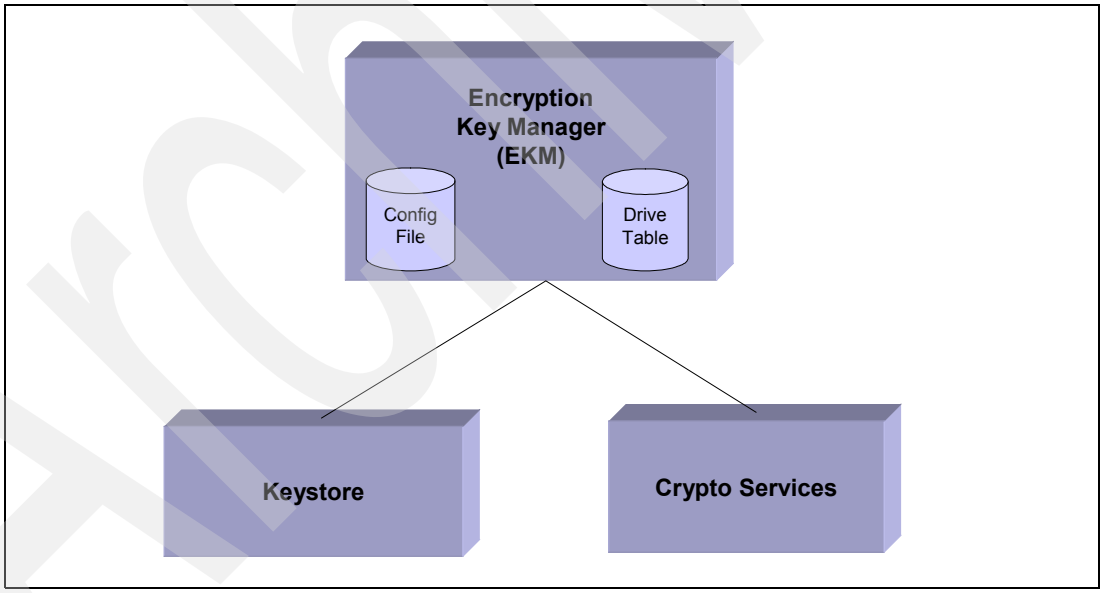


Figure 2-2 EKM components and resources

Note: In Figure 2-2, Crypto Services can refer to either Java security providers (software), or cryptographic hardware installed on the system.

Tape drive table

The tape drive table is used by EKM to track the tape devices that it supports. The tape drive table contains the list of the drives that can communicate with the EKM. You can populate this

list with additional drives by using the EKM *adddrive* command, or you can set a variable in the configuration file so that EKM adds unknown drives to the list automatically.

The tape drive table also stores the default key labels for TS1100 drives and the active key set list for LTO drives.

The tape drive table is a non-editable, binary file whose location is specified in the configuration file. A number of EKM commands are available to add, delete, modify, and view keys and certificates. You may change the location of the tape drive table to meet your requirements.

Tip: The option to automatically accept unknown tape drives can facilitate the task of populating the tape drive table with your drives. For security reasons, you might want to turn off this option as soon as all of your tape drives have been added to the table. In a business and continuity recovery site however like Sunguard or IBM BCRS it is required to accept unknown tape drives.

Configuration file

The configuration file is an editable file which tells your EKM how to operate. Among others, you specify in this file where the keystore and the drive table are located.

We discuss the configuration file extensively in Chapter 5, “Planning for EKM and its keystores” on page 139, and later in Part 2, “Implementing and operating the EKM” on page 137 where we describe the full set of configuration options.

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 EKM to perform cryptographic operations. A keystore can be either hardware-based or software-based.

EKM supports several types of Java keystores, offering a variety of operational characteristics to meet your requirements:

- JCEKS: clear symmetric keys, clear asymmetric keys
- JCE4758KS/JCECCAKS: clear symmetric keys, clear asymmetric keys, secure symmetric keys, secure asymmetric keys
- JCE4785RACFKS/JCECCARACFKS: secure asymmetric keys
- JCERACFKS: clear asymmetric keys
- PKCS11IMPLKS: types of keys supported depends on the PKCS11 implementation
- IBMi5OSKeyStore: clear asymmetric keys

We discuss the characteristics of these keystores in 5.3, “EKM and keystore considerations” on page 146.

Note: Encryption on IBM LTO Ultrium 4 drives requires a keystore that supports symmetric keys.

Cryptographic Services

EKM uses the IBM Java Security components for its cryptographic capabilities. EKM does not provide cryptographic capabilities and therefore does not require, nor is allowed to obtain,

FIPS 140-2 certification. However, EKM takes advantage of the cryptographic capabilities of the IBM Java Virtual Machine 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. In the Configuration Properties file, setting the FIPS configuration parameter to `ON` causes EKM to use the IBMJCEFIPS provider for all cryptographic functions.

Note: Do *not* use hardware-based keystore types when the FIPS parameter is set `ON`.

More information about the IBMJCEFIPS provider, its selection, and its use is at:

`http://www.ibm.com/developerworks/java/jdk/security/50/FIPShowto.html`

2.1.2 Encryption keys and 3592 and LTO4 differences

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 is that was constructed this way, the harder it is to break the encryption code. Both the IBM and T10 methods of encryption use 256-bit Advanced Encryption Standard (AES) algorithm keys to encrypt data. The 256-bit AES is the encryption standard currently recognized and recommended by the U.S. Government, and allows three key lengths.The 256-bit keys are the longest allowed by AES.

The two types of encryption algorithms can be used by EKM are symmetric and asymmetric. 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. The 256-bit AES keys are symmetric keys. TS1120, TS1130 and LTO4 *all* use 256-bit AES symmetric keys to encrypt user data.

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 public-private encryption algorithm is also referred to as the RSA algorithm for public key cryptography, which is named after the inventors, Ron Rivest, Adi Shamir, and Leonard Adleman (Rivest-Shamir-Adleman or RSA algorithm).

EKM uses both symmetric and asymmetric keys. It uses symmetric encryption for high-speed encryption of user or host data, and asymmetric encryption (which is necessarily slower) for protecting the symmetric key.

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. The following applies when implementing encryption using either LME or SME. EKM and all of its supported tape drives (TS1120, TS1130, and LTO4) use symmetric, 256-bit AES keys to encrypt user data. The keys used to encrypt client data are referred to as *data keys (DKs)*. Important differences exist between the way TS1120 and TS1130 Tape Drives and LTO Ultrium 4 Tape Drives handle these DKs.

3592 and LTO4 Tape Drives

IBM encryption capable drive types use 256-bit AES keys to encrypt user data. The TS1120 and TS1130 encryption DKs are randomly generated when needed, used, and then discarded. LTO4 encryption DKs are randomly pregenerated and then kept in the EKM keystore.

The LTO4 Tape Drive also uses 256-bit AES symmetric data keys to encrypt user data when writing to LTO4 cartridges; however, the LTO4 data key is pregenerated and not randomly generated like the 3592 drives. EKM selects a pre-generated data key in a round-robin fashion. The data key is sent to the drive in a secure manner the same as the TS1120 and TS1130. Unlike the 3592 drives, the data key is not stored on the cartridge. On an LTO4 cartridge, a pointer to the encrypting data key (key label or alias) is stored instead. The data key must be accessible in a keystore based on the alias or key label and available to EKM the volume can be read. Table 2-1 reflects key usage by encryption management method.

Tip: As of IBM 31-bit SDK for z/OS, J2TE, V5.0, SDK SR9 level or later, functionality has been added to the hwkeytool (Key and Certificate Management Tool) to generate SECURE symmetric keys. An example of this function in a z/OS EKM using a JCECCAKS is in Appendix F, "TS1100 and LTO4 SECURE key EKM on z/OS" on page 637.

TS1120 and TS1130 Tape Drives

In addition to 256-bit AES symmetric data keys that are *randomly* generated for each volume being encrypted, EKM also uses public-private (*asymmetric*) key cryptography to protect the symmetric data encryption keys that are generated and retrieved as they pass between EKM and 3592 tape drives. Public-private (asymmetric) key cryptography is also used to protect the data key while it is stored on the cartridge. See Table 2-1.

Table 2-1 Key usage by drive type and management method

Encryption management method	Keys used by	
	TS1120 drive or TS1130	LTO4 drive
System-Managed Encryption or Library-Managed Encryption using EKM	One randomly generated unique DK[a] per cartridge	One pregenerated DK per cartridge
Application-Managed Encryption (no EKM)	One randomly generated unique DK per cartridge	One DK per cartridge

a. DK is symmetric AES 256-bit data key

2.1.3 Key exchange

EKM acts as a process awaiting key generation or key retrieval requests sent to it through a TCP/IP communication path between EKM 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 EKM. The tasks that the EKM performs upon receipt of the request are different for the TS1100 series drives.

TS1120 and TS1130 Tape Drives

EKM requests an Advanced Encryption Standard (AES) key from the cryptographic services 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 EKM 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. EKM also allows protected AES keys to be rewrapped, or rekeyed, using different RSA keys from the original keys that were 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 a more detailed description, refer to 2.3.3, "Encrypting and decrypting with SME and LME" on page 38.

LTO Ultrium 4 Tape Drives

The EKM 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 data being written to tape.

When an encrypted tape is read by an LTO Ultrium 4 Tape Drive, the EKM 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.

2.2 Tivoli Key Lifecycle Manager

In your enterprise, a large number of symmetric keys, asymmetric keys, and certificates can exist. All of these keys and certificates have to be managed. Key management can be handled either internally by an application, such as Tivoli Storage Manager, or externally by an Encryption Key Manager (EKM). In this section, we discuss the Tivoli Key Lifecycle Manager (TKLM).

The TKLM product is an application that performs key management tasks for IBM hardware that is encryption-enabled like the IBM encryption-enabled TS1120 and TS1130 Tape Drives and Linear Tape-Open (LTO) Ultrium 4 Tape Drives. The tasks provide, protect, store, and maintain encryption keys that are used to encrypt information being written to, and decrypt information being read from tape media. TKLM operates on a variety of systems. Currently, the supported operating systems are:

- AIX 5.3: 64 bit
- Red Hat® AS 4.0 x86: 32 bit
- SUSE® Linux 9.0 and 10 x86: 32 bit
- Solaris 10 Sparc: 64 bit
- Windows Server® 2003: 32 bit

TKLM is designed to be a shared resource deployed in several locations within an enterprise. It 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).

2.2.1 Tivoli Lifecycle Key Manager components and resources

The sole task of the Tivoli Lifecycle Key Manager is to handle the serving of keys to the encrypting tape drives. The TKLM does not perform any cryptographic operations, such as generating encryption keys, and it does not provide storage for keys and certificates. To perform these tasks, TKLM has to rely on external components. In the following sections, we describe the components of TKLM and resources that are used by TKLM. In addition to the key-serving function the TKLM also brings the following additional functions over the EKM:

- Lifecycle functions
  - Notification of certificate expiration through the Tivoli Integrated Portal (TIP)
  - Automated rotation of certificates
  - Automated rotation of groups of keys
- Usability enhancements of EKM
  - Provides a graphical user interface (GUI)
  - Initial configuration wizards
  - Migration wizards
- Integrated backup and restore of TKLM file
  - One button to create and restore a single backup packaged as a JAR file
- TIP installation manager
  - Simple to use installation for Windows, Linux, AIX, Solaris
  - Can be a silent installation

Figure 2-3 shows the TKLM components and external resources.

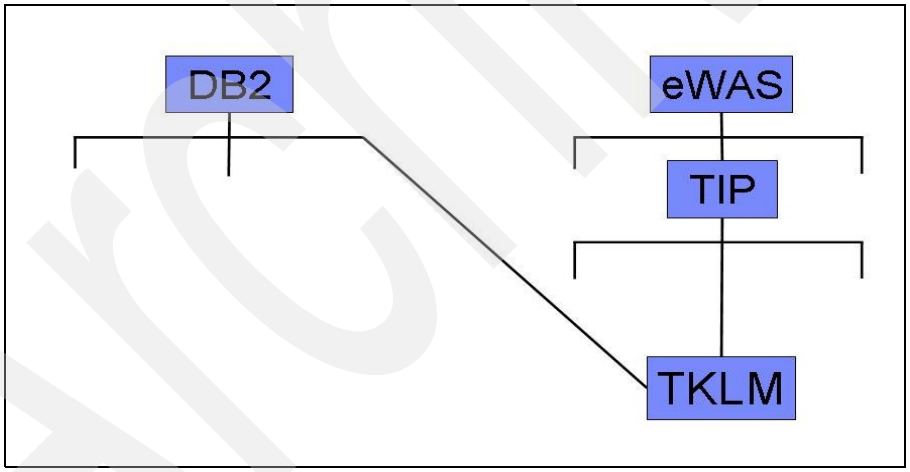


Figure 2-3 TKLM components and resources

DB2

In TKLM the drive table is now stored in DB2®, giving the user a more robust interface to managing drives, and the keys and certificates that are associated with those drives. In the EKM, the only place to find which tapes were encrypted with what keys, and similar audit information was in the EKM audit log and the EKM `metadata.xml` file. In TKLM, this information is stored in the TKLM DB2 tables enabling the user to more easily search and query that information.

Tip: The option to automatically accept unknown tape drives can facilitate the task of populating the tape drive table with your drives. For security reasons, you might want to turn off this option as soon as all of your tape drives have been added to the table. In a business and continuity recovery site however, like Sunguard or IBM BCRS, it is required to accept unknown tape drives.

Configuration file

Similar to the EKM, the TKLM also has an editable configuration file with additional configuration parameters that is not offered in the GUI. The file can be text-edited, however the preferred method is to modify the file through the TKLM command-line interface (CLI).

We discuss the configuration of TKLM extensively in Chapter 10, "Implementing TKLM" on page 325 where we describe the full set of configuration options.

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 (JRE™). A *keystore* holds the certificates and keys (or pointers to the certificates and keys) used by TKLM to perform cryptographic operations. A keystore can be either hardware-based or software-based.

TKLM supports several types of Java keystores, offering a variety of operational characteristics to meet your requirements.

TKLM Open systems

TKLM on open systems supports the JCE keystore (JCEKS). This keystore supports both CLEAR key symmetric keys, and CLEAR key asymmetric keys. Symmetric keys are used for LTO4 encryption drives and asymmetric keys are used for TS1100 Tape Drives.

We discuss the characteristics of these keystores in detail in 5.3, "EKM and keystore considerations" on page 146.

Cryptographic services

TKLM uses the IBM Java Security components for its cryptographic capabilities. TKLM does not provide cryptographic capabilities and therefore does not require, or allowed to obtain, FIPS 140-2 certification. However, TKLM takes advantage of the cryptographic capabilities of the IBM Java Virtual Machine (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, either through text editing or by using the TKLM CLI, you can make TKLM use the IBMJCEFIPS provider for all cryptographic functions.

You can find more information about the IBMJCEFIPS provider, its selection, and its use at:

`http://www.ibm.com/developerworks/java/jdk/security/50/FIPShowto.html`

2.2.2 Key exchange

TKLM acts as a process awaiting key generation or key retrieval requests sent to it through a TCP/IP communication path between TKLM 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 TKLM. The tasks that the TKLM performs upon receipt of the request are different for the TS1100 Tape Drive.

TS1100 Tape Drives

TKLM requests an Advanced Encryption Standard (AES) key from the cryptographic services and serves it to the tape drives in two protected forms:

- Encrypted or wrapped, using Rivest-Shamir-Adleman (RSA) key pairs. TS1100 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.

Additionally the libraries now support SSL encrypted connections between the TKLM and library for key exchanges. However, note that when not using SSL for key exchange the key material will be encrypted in another fashion. The transport of the keys are always secure across the TCP/IP connection.

When an encrypted tape cartridge is read by a TS1100 Tape Drive, the protected AES key on the tape is sent to TKLM 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. TKLM also allows protected AES keys to be rewrapped, or rekeyed, using different RSA keys from the original keys that were 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 having to rewrite the entire tape and enables a tape cartridge's data key to be reencrypted with a business partner's public key. For a more detailed description, refer to 2.3.3, "Encrypting and decrypting with SME and LME" on page 38.

LTO Ultrium 4 Tape Drives

The TKLM 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 TKLM 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.

2.3 Methods of managing IBM tape encryption

There are three methods of tape encryption management supported by the IBM tape data encryption solution. These methods differ in where the encryption policy engine resides, where key management is performed, and how the EKM is connected to the drive. Encryption policies control which volumes need to be encrypted.

Key management and the encryption policies can be located in any one of the following three environmental layers:

- System layer
- Library layer
- Application layer

In accordance with the layers, we call these methods:

- System-Managed Encryption (SME)
- Library-Managed Encryption (LME)
- Application-Managed Encryption (AME)

Not all operating systems, applications, and tape libraries support all of these methods, and where they are supported; not all of the methods are equally suitable. When you plan for tape encryption, select the encryption method depending on your operating environment. In the following sections, we explain the characteristics of AME, SME, and LME.

Note: For the remainder of this chapter, in all text and figures, the process flows are the same for the EKM and TKLM. You can substitute the TKLM for the EKM for these discussions.

2.3.1 System-Managed Encryption

In a System-Managed Encryption (SME) implementation, encryption policies reside within the system layer. This method of tape encryption requires an EKM for key management. SME is fully transparent to the application and library layers. Figure 2-4 on page 34 shows a schematic illustration of SME.

SME is supported on z/OS, z/VM, z/VSE, z/TPF, Linux on zSeries®, and a number of open systems platforms. On z/OS, z/VM, z/VSE, z/TPF, and Linux on zSeries, SME is the only encryption method supported. SME is supported on z/OS using Data Facility Storage Management Subsystem (DFSMS). On open systems platforms, the IBM tape device driver is used for specifying encryption policies on a per-drive basis.

The following open systems operating systems are currently supported:

- AIX
- Windows
- Linux
- Solaris

SME offers you centralized enterprise-class key management, which facilitates tape interchange and migration. Another advantage is its support for stand-alone drives. Drawbacks are its policy granularity on open systems, additional responsibilities for the storage administrator, and the dependency of data access on the availability of the EKM and the key path.

SME shares most of its advantages and disadvantages with LME, but with two major differences. Naturally, LME does not support stand-alone tape drives. However in an open systems environment, LME gives you a better policy granularity than SME, because you can control encryption on a per-volume basis with TS3500 and 3494 tape libraries. On z/OS, you can control encryption on volume level through the use of DSMFS.

In a System z environment that does not support encryption, or in an open systems environment with stand-alone drives and an application that does not support encryption, SME is the only choice. In all other environments, consider LME as an alternative.

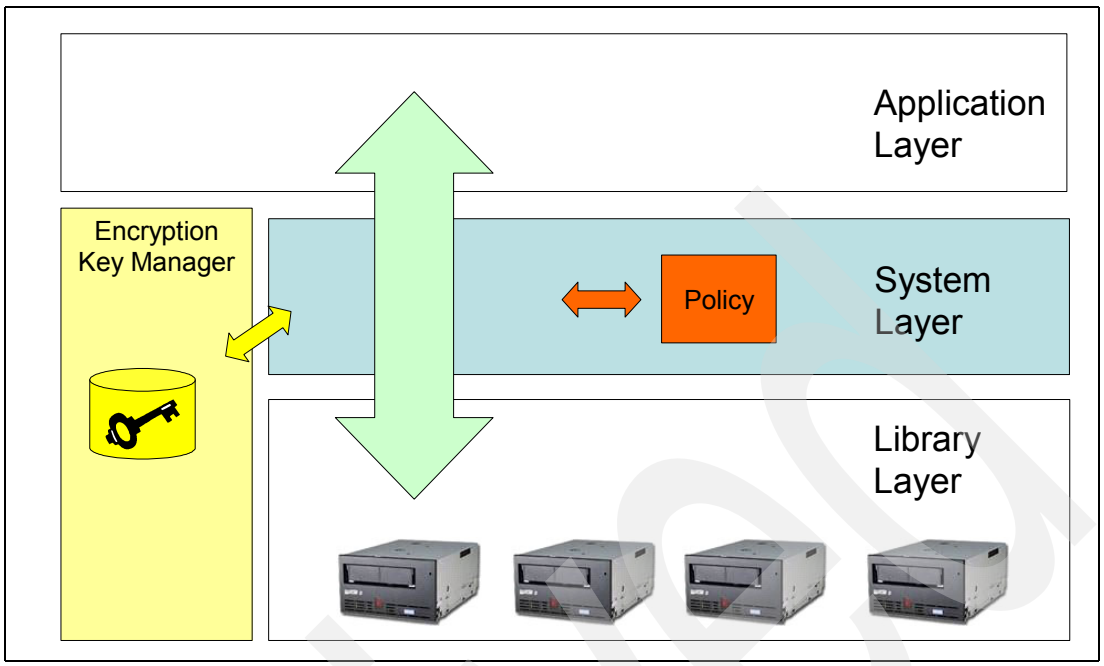


Figure 2-4 System-Managed Encryption (SME)

System-Managed Encryption for open systems

Encryption policies specifying when to use encryption are set up in the IBM tape device driver. For details about setting up System-Managed Encryption on tape drives in an Open Systems environment, refer to the *IBM Tape Device Driver Installation and User's Guide,* GC27-2130*,* and the planning and operator guide for your tape library.

On Open Systems, this support can be described as *in-band* where tape drive requests to the EKM component travel over the Fibre Channels to the server hosting the EKM.

System-Managed Encryption for System z

On z/OS encryption, policies specifying when to use encryption are set up in DFSMS (Data Facility Storage Management Subsystem). You can also use additional software products, such as IBM Integrated Cryptographic Service Facility (ICSF) and IBM Resource Access Control Facility (RACF). Key generation and management is performed by the Encryption Key Manager (EKM), 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 Tape Drives that are connected to an IBM Virtualization Engine TS7700, encryption key labels are assigned using the Maintenance Interface on a per-storage-pool basis. 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, refer to Chapter 13, "Implementing TS7700 Tape Encryption" on page 421 or refer to the white paper, *IBM Virtualization Engine TS7700 Series Encryption Overview*, which is available at:

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

For information about encryption, refer to *DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592)*, SC26-7514.

Encryption key paths

System-Managed Encryption on z/OS use either of the following key flows:

- In-band encryption key flow: Tape drive requests to the Encryption Key Manager component travel over the ESCON/FICON® channels to the server proxy that is TCP/IP-connected to the Encryption Key Manager
- Out-of-band encryption key flow: The tape controller establishes the communication to the EKM server over a TCP/IP connection. Out-of-band support requires a router.

Out-of-band support also runs on VM, VSE, TPF, and Linux on zSeries and is your only option on those operating system platforms. The TS7700 Virtualization Engine also uses out-of-band support.

In-band key flow

In-band key flow, shown in Figure 2-5, occurs between EKM and the tape drive through a FICON proxy on the FICON/ESCON® interface. The FICON proxy supports failover to the secondary key path on failure of first-specified EKM 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 “encryption failure unit checks” to the host.
- Must be reconfigured when new encryption drives are introduced for attachment or when an encryption-capable drive is enabled for encryption.

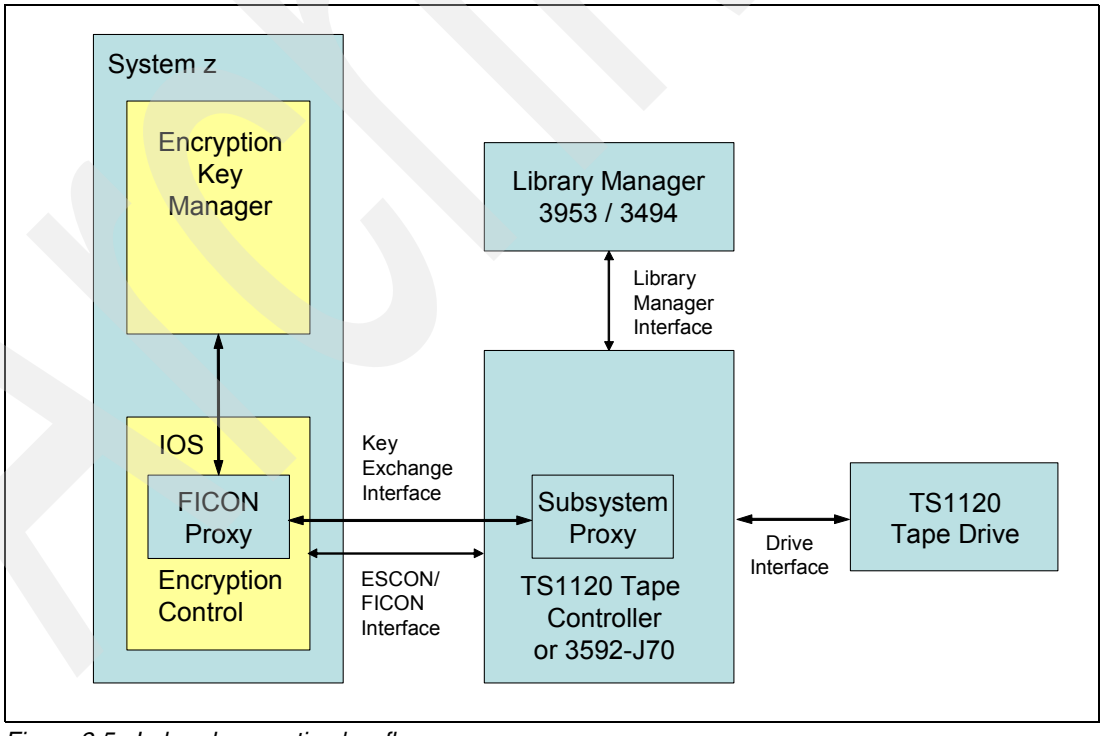


Figure 2-5 In-band encryption key flow

Out-of-band key flow

Out-of-band key flow, which is shown in Figure 2-6, occurs between the EKM and the tape drive through a subsystem proxy, which is located in the 3592 controller or TS7700 Virtualization Engine on the EKM interface. Impact on service requirements can be greater than for in-band key flow because of the introduction of two routers on the EKM interface, to and from the controller.

The controller and the TS7700:

- Support failover to the secondary key path on failure of the first-specified EKM path addresses
- Report drive status in SMIT displays
- Pass encryption-related errors from the drive to the host
- Report “encryption failure unit checks” to the host
- Must be reconfigured when new encryption drives are introduced for attachment or when an encryption-capable drive is enabled for encryption

You may enter up to two EKM IP/domain addresses (and up to two ports) for each controller, and two domain name server IP addresses.

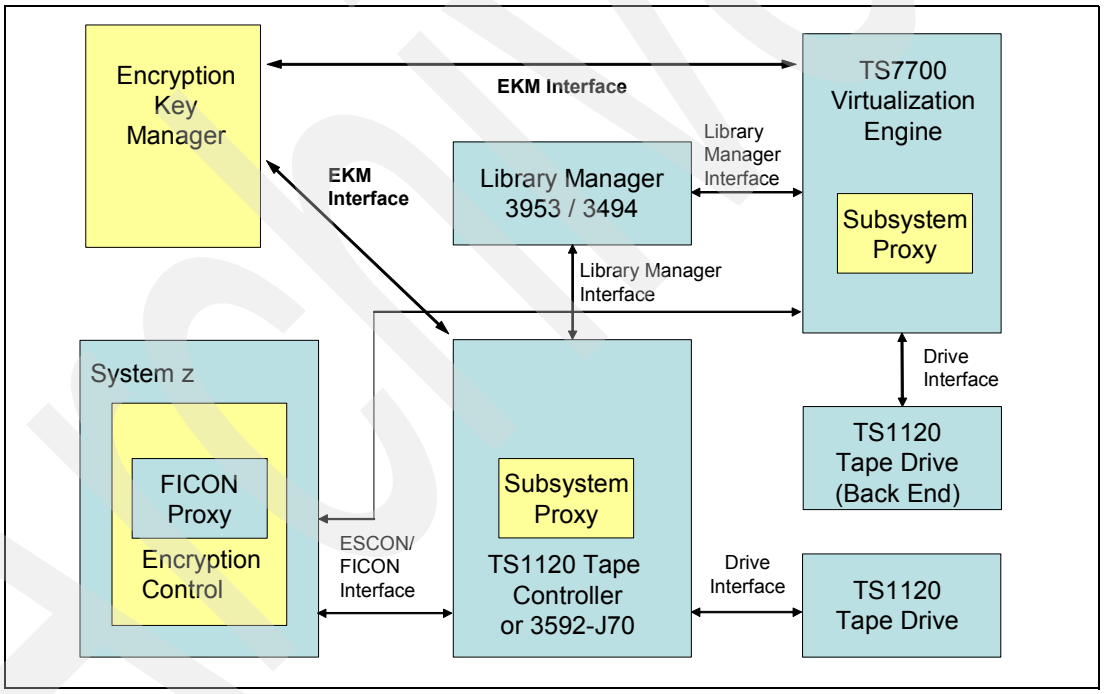


Figure 2-6 Out-of-band encryption key flow

2.3.2 Library-Managed Encryption

In a Library-Managed Encryption (LME) implementation, encryption policies reside within the tape library. This method of tape encryption requires an EKM for key management. LME is fully transparent to the application and system layers. Figure 2-7 on page 37 shows an illustration of Library-Managed Encryption.

LME offers you the broadest range of application and operating system support. Centralized enterprise-class key management facilitates tape interchange and migration. If you

implement LME on a TS3500 or 3494 tape library, you get policy granularity on a per-volume basis. LME comes with additional responsibilities for the storage administrator as compared to AME. Data access depends on the availability of the EKM and the key path.

In most Open Systems environments, LME is the preferred method for tape encryption.

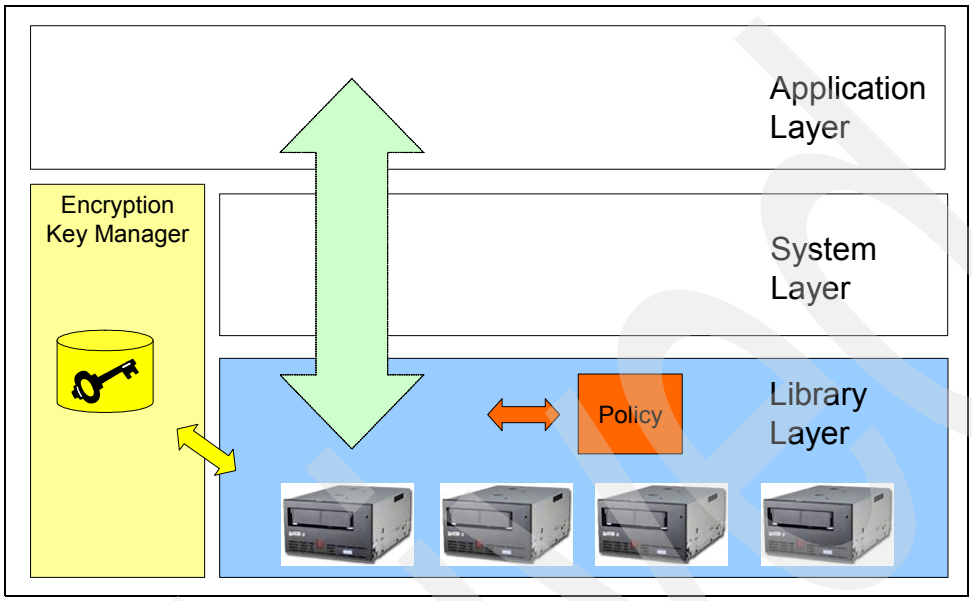


Figure 2-7 Library-Managed Encryption (LME)

LME can be implemented on:

- Open systems-attached TS3500 tape library with TS1120 and LTO Ultrium 4 Tape Drives
- Open systems-attached 3494 or TS3400 tape library with TS1120 Tape Drives
- TS3310, TS3200, or TS3100 tape library with LTO Ultrium 4 Tape Drives

Key generation and management is handled by EKM, running on a host with a TCP/IP connection to the library. Policy control and keys pass through the library-to-drive interface; therefore, encryption is transparent to the applications.

For TS3500 and IBM 3494 tape libraries, you can use barcode encryption policies (BEPs) to specify when to use encryption. On an IBM TS3500 Tape Library, you set these policies through the IBM System Storage Tape Library Specialist Web interface. On a 3494 tape library, you can use the Enterprise Automated Tape Library Specialist Web interface or the Library Manager Console. With BEPs, policies are based on cartridge volume serial numbers. LME also allows for encryption of all volumes in a library, independent of barcodes.

For certain applications, such as Symantec NetBackup, LME includes support for Internal Label Encryption Policy (ILEP). When ILEP is configured, the TS1120 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.

The following IBM tape libraries support Library-Managed Encryption:

- IBM System Storage TS3500 Tape Library
- IBM TotalStorage® 3494 Tape Library
- IBM System Storage TS3310 Tape Library
- IBM System Storage TS3200 Tape Library
- IBM System Storage TS3100 Tape Library

Note: System-Managed Encryption and Library-Managed Encryption interoperate with one another. A tape that is encrypted using SME can be decrypted using LME, and the other way around, provided that they both have access to the same keys and certificates.

2.3.3 Encrypting and decrypting with SME and LME

Encryption and decryption with System-Managed Encryption and with Library-Managed Encryption are identical as far as their process flow.

SME and LME encryption processes

Figure 2-8 on page 39 describes the flow of encrypted data to tape, and how keys are communicated to the tape drive and then stored on the tape media. In this particular example, we assume an EKM is running on an abstract server, and that the tape library and, consequently, the tape drives are connected to another abstract server. These can be the same server or different servers, because whether the server is the same or not does not affect the outcome.

We assume that a certificate from a business partner had been imported into this keystore. It only has a public key associated with it; the business partner has the corresponding private key.

Now, our server sends a write request to the drive. Our drive is encryption-capable, and the host has requested encryption. As part of this initial write, the drive obtains two Key Encrypting Key (KEK) labels from the host or a proxy, which are aliases for two Rivest-Shamir-Adleman (RSA) algorithm KEKs. The drive requests that the EKM send it a data key (DK) and to encrypt the DK using the public KEKs aliased by the two KEK labels.

The EKM validates that the drive is in its list of valid drives. After validation, the EKM obtains a random DK from cryptographic services. The EKM then retrieves the public halves of the KEKs aliased by the two KEK labels. The EKM then requests that cryptographic services create two encrypted instances of the DK using the public halves of the KEKs, therefore, creating two Externally Encrypted Data Keys (EEDKs).

The EKM sends both EEDKs to the tape drive. The drive stores the EEDKs in the cartridge memory (CM) and three locations on the tape. The EKM also sends the DK to the drive in a secure manner. The drive uses the separately secured DK to encrypt the data.

The two modes for creating the EEDK are:

- CLEAR or LABEL: In this mode, the KEK label is stored in the EEDK.
- Hash: In this mode, a Hash of the public half of the KEK is stored in the EEDK.

When sharing business partner KEKs, we recommend using the Hash mode. The Hash mode lets each party use any KEK label when importing a certificate into their keystore. The alternative is to use the CLEAR or LABEL mode and then have each party agree on a KEK label.

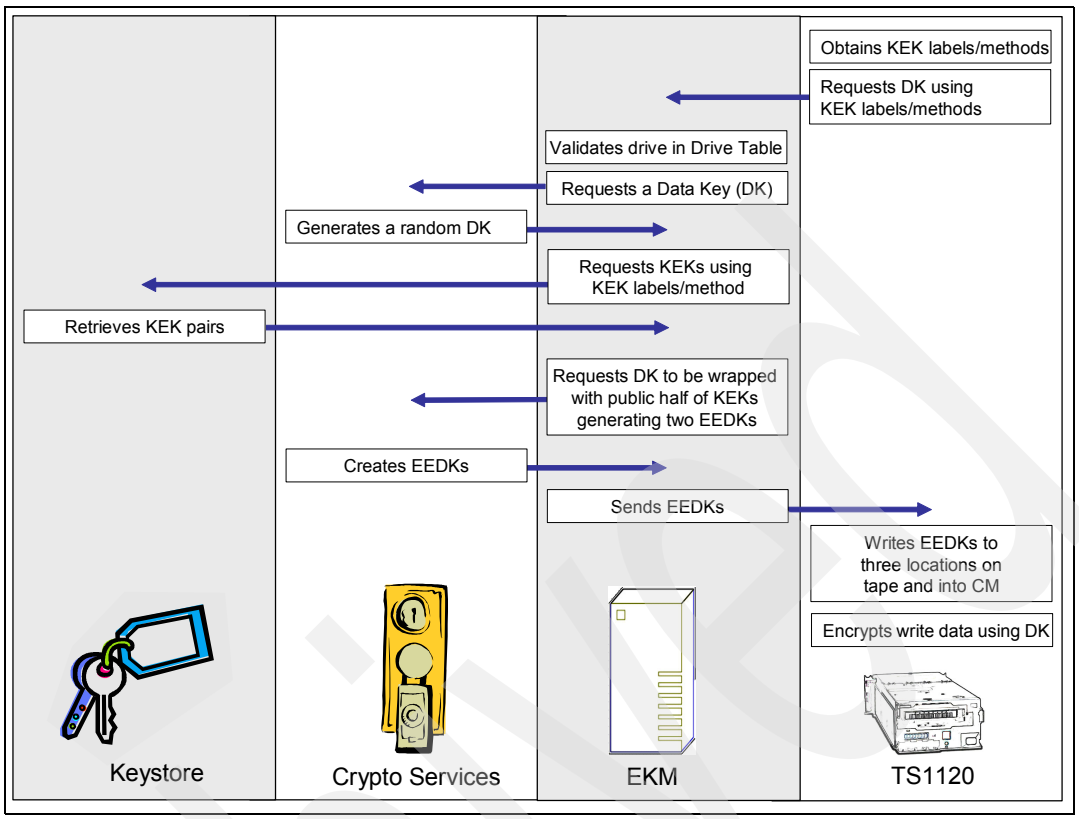


Figure 2-8 Key and data flow for encryption using SME or LME

SME and LME decryption processes for TS1120

Figure 2-9 on page 40 shows the key and data flow for decryption. In this example, we assume that the data was encrypted at another site. For the decryption process, the tape has two EEDKs stored in its cartridge memory. We call these EEDK1 and EEDK2. EEDK1 was stored with the CLEAR (or LABEL) mode selected, and EEDK2 was stored with the Hash mode selected.

An encrypted tape is mounted for a read or a write append. The two EEDKs are read from the tape. The drive asks the EKM to decrypt the DK from the EEDKs. The EKM validates that the drive is in its list of valid drives. After validation, the EKM requests the keystore to provide the private halves of each KEK used to create the EEDKs. The KEK label associated with EEDK1 cannot be found in the keystore, but the Hash of the public key for EEDK2 is found in the keystore.

The EKM asks cryptographic services to decrypt the DK from EEDK2 using the private half of the KEK associated with EEDK2. The EKM then sends the DK to the drive in a secure manner. The drive then decrypts the data on the tape. In our example, we described reading from an encrypted tape. Exactly the same communication between tape drive and the EKM takes place for a write-append.

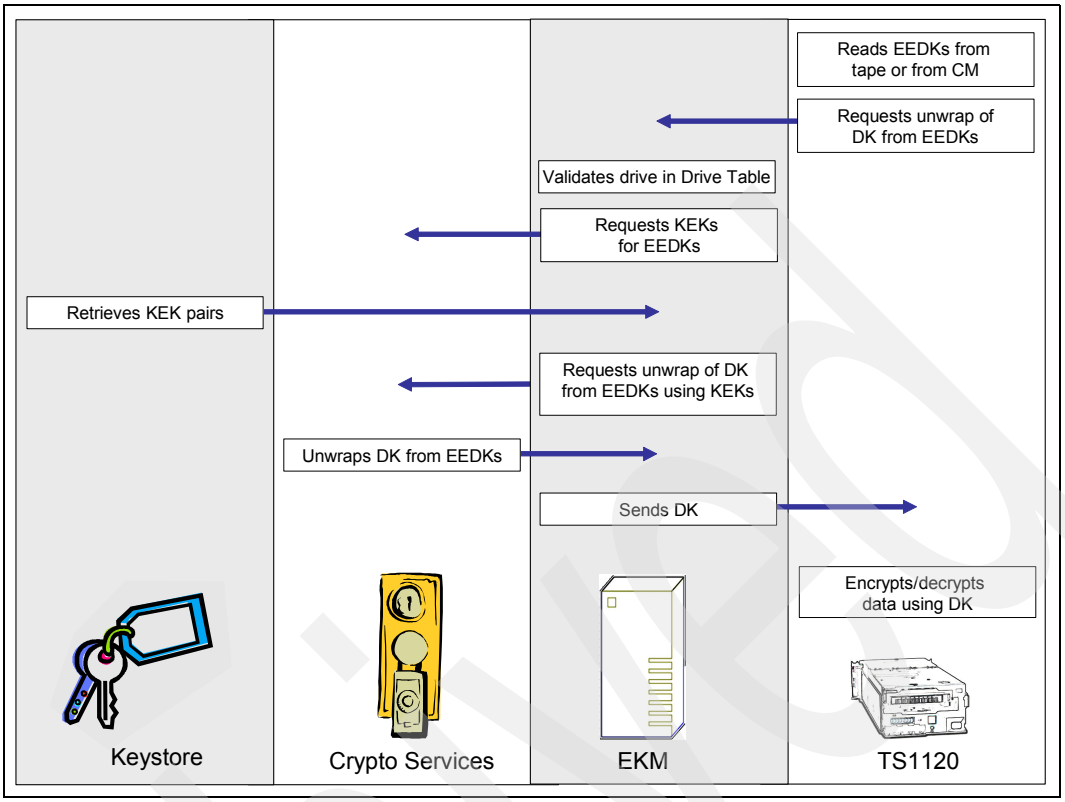


Figure 2-9 Key and data flow for decryption using SME or LME

2.3.4 Application-Managed Encryption

For Application-Managed Encryption (AME), the application has to be capable of generating and managing encryption keys and of managing encryption policies. Tivoli Storage Manager contains these capabilities. 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. Refer to Figure 2-10 on page 41.

AME is the easiest encryption method to implement and adds the fewest responsibilities for the storage administrator. Because the data path and the key path are the same, there is no additional risk to data and drive availability. Policy granularity depends on the application. With Tivoli Storage Manager, you control encryption on a storage pool basis. There is no centralized key management with AME, because the application generates, stores, and manages the encryption keys. The lack of centralized key management makes tape interchange and migration more difficult.

AME can be the most convenient solution when Tivoli Storage Manager is the only application that utilizes tape encryption.

Tivoli Storage Manager does not restrict you to using AME. You can also choose SME or LME to encrypt Tivoli Storage Manager data.

Note: Tape volumes written and encrypted using the AME method can only be decrypted with an AME solution. In addition, because the data keys reside only in the Tivoli Storage Manager database, the same database must be used.

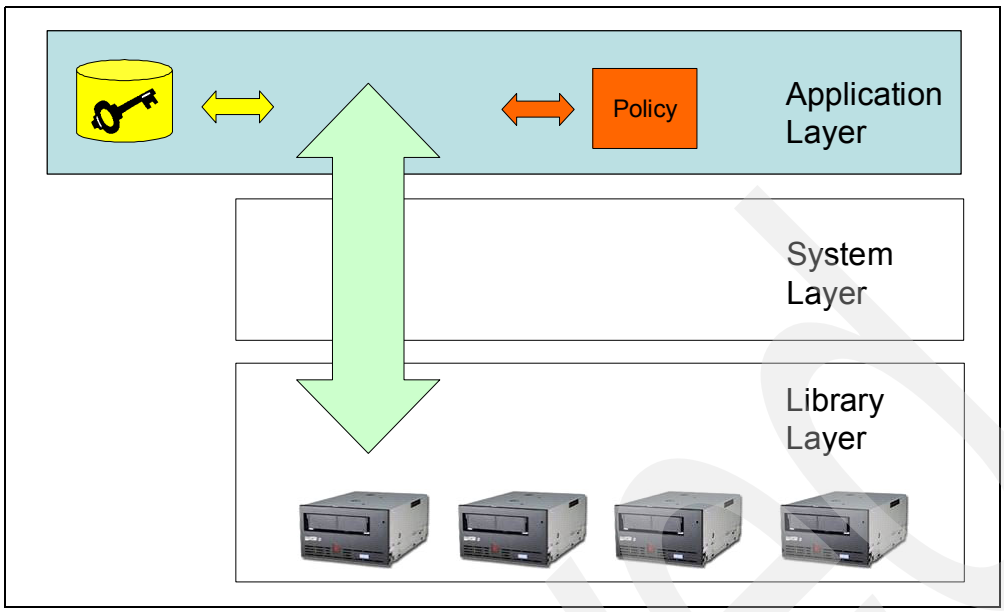


Figure 2-10 Application-Managed Encryption (AME)

AME on IBM TS1120 and LTO Ultrium 4 Tape Drives can use either of two encryption command sets, the IBM encryption command set developed for EKM or the T10 command set defined by the International Committee for Information Technology Standards (INCITS).

AME using the TS1120 and TS1130 Tape Drives 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 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 AME, refer to your Tivoli Storage Manager documentation or the following Web site:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp`

Note: EKM is not required by, or used by, AME.

Example for Application-Managed Encryption

In this section, we describe how data is encrypted to tape using Tivoli Storage Manager as the key manager. Figure 2-11 on page 42 shows a high-level diagram depicting how data and keys travel to and from the tape when writing from the beginning of the tape (BOT).

In the figure, a tape drive mounts a tape for encryption. The tape drive then sends its tape ID or VOLSER to Tivoli Storage Manager. Tivoli Storage Manager then generates a 256-bit AES data key, encrypts the data key, and stores the encrypted data key and the tape identifier in the Tivoli Storage Manager database. Tivoli Storage Manager then sends the data key to the

tape drive. Using the AES algorithms and the data key that was sent to it, the tape drive encrypts data to the tape.

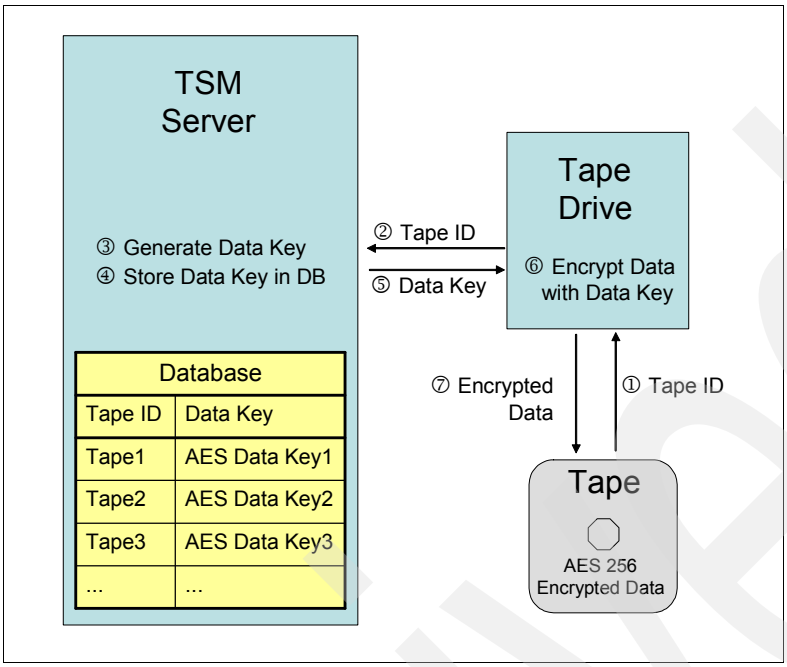


Figure 2-11 Application-Managed Encryption data flow for encryption

Figure 2-12 on page 43 depicts the flow of data and keys when using AME to read or write append an encrypted cartridge. In this diagram, Tivoli Storage Manager is the application that controls encryption.

In our example, an encrypted tape is mounted for decryption. The tape drive reads the tape ID or VOLSER and sends that information to Tivoli Storage Manager. Tivoli Storage Manager looks up that tape ID in its internal database and finds the entry associated with it. Tivoli Storage Manager then decrypts the data key from the entry. Tivoli Storage Manager then sends the data key to the tape drive.

Now, the tape drive can read data from the tape, decrypting the data as it reads using the 256-bit AES data key and the AES algorithms to yield clear data.

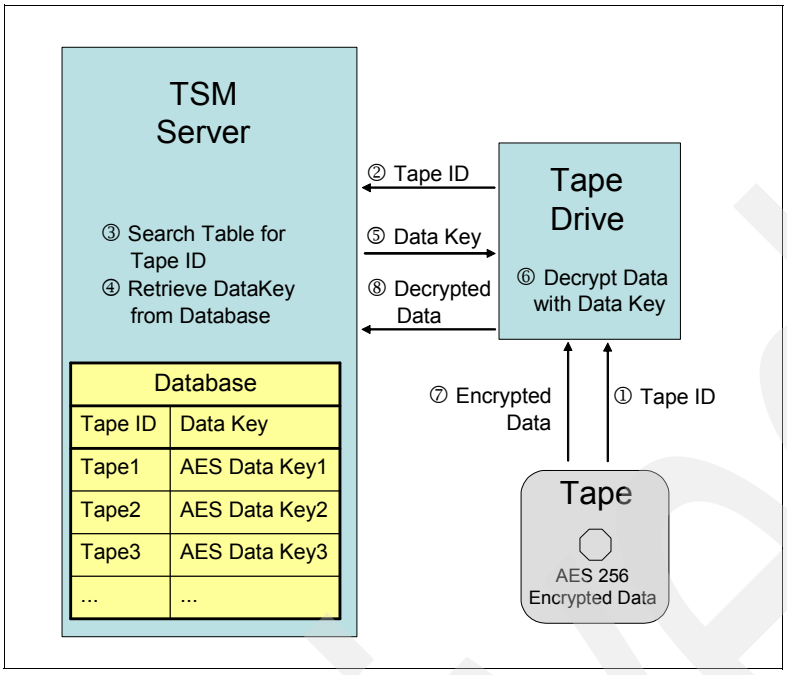


Figure 2-12 Application-Managed Encryption data flow for decryption

2.3.5 Mixed mode example

In the previous sections, we described how encryption works with different tape encryption methods. This section describes a mixed environment, using different types of tape encryption methods. In reality, an enterprise is likely to have several types of systems. The tape solutions can commingle and allow all of those disparate setups to take advantage of tape data encryption.

In Figure 2-13 on page 44, we introduce a picture of an enterprise taking advantage of both in-band and out-of-band encryption. In addition, this picture shows both a System-Managed Encryption solution and a Library-Managed Encryption solution implemented concurrently in the same enterprise.

In our example, an EKM is running on a z/OS system, taking advantage of hardware cryptography for its keystore. There is also a miscellaneous IBM server system with an EKM running and an open systems server attached to a TS3500 or 3494 tape library. The z/OS system and the open systems server are attached to two separate libraries. The IBM server, which is running a backup EKM, is not attached to any libraries, but it is attached to the shared LAN/WAN.

We can see that the open systems server is running LME on its library. All data is sent across Fibre Channel to the library to be encrypted to tape. The keys that this library uses for encryption, however, come across the network from either the z/OS EKM or the IBM server EKM.

The library that is attached to the z/OS system shows both in-band and out-of-band encryption. The z/OS system attached to the library is using in-band encryption. EKM communication to the library is across the Fibre Channel, and data is also sent across the Fibre Channel. In addition, this library is pointing to the backup EKM on the IBM server system. The keys that are served to the library from the IBM server flow across the network which requires the library's control unit to have network access.

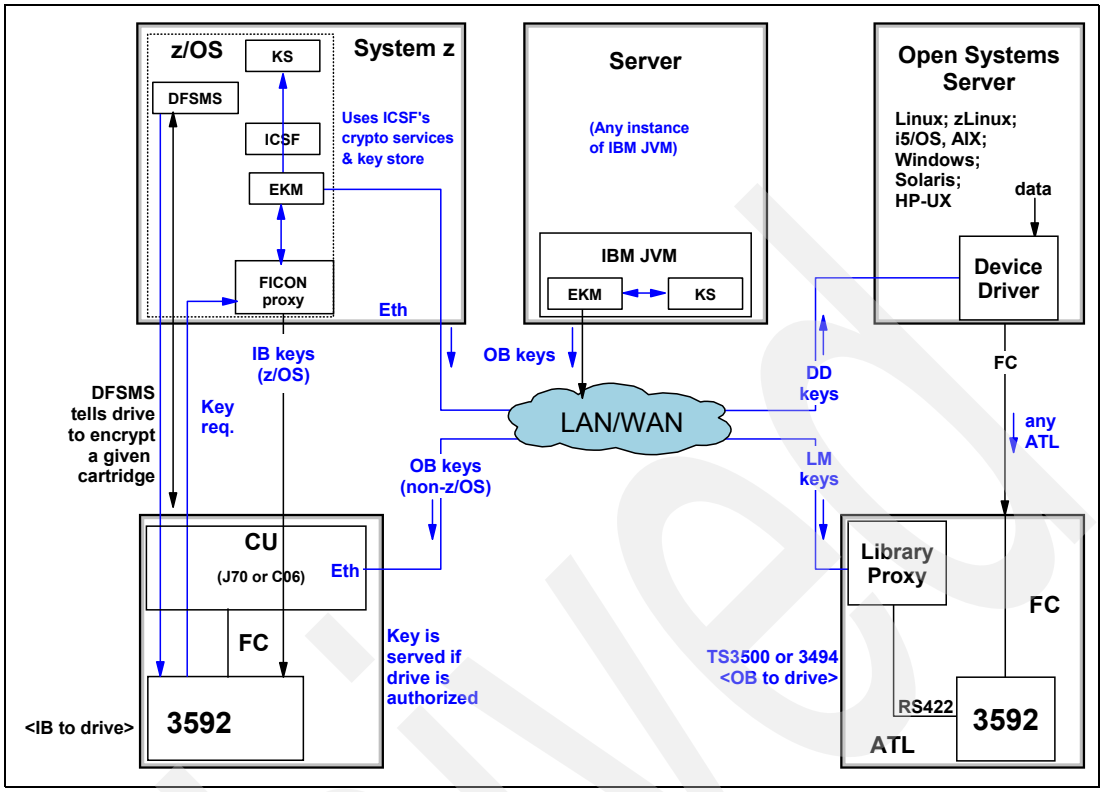


Figure 2-13 Encryption in a mixed environment

3

IBM System Storage tape and tape automation for encryption

A wide variety of IBM tape products support encryption. In this chapter, we introduce and describe the IBM tape drives, the IBM tape libraries, and the IBM Tape Virtualization solution that support tape encryption in Open Systems and System z environments. We will discuss the characteristics of each product and how it supports tape encryption.

We describe the following products:

- IBM System Storage TS1120 and TS1130 Tape Drives
- IBM System Storage TS1120 Tape Controller Model C06
- IBM Virtualization Engine TS7700
- IBM Linear Tape-Open (LTO) Ultrium 4 Tape Drives
- IBM LTO Tape Libraries:
  - IBM TS2900 Tape Autoloader
  - IBM TS3100 Tape Library Express Model
  - IBM TS3200 Tape Library Express Model
  - IBM TS3310 Tape Library
  - IBM TS3400 Tape Library
- IBM System Storage TS3500 Tape Library
- IBM TotalStorage 3494 Tape Library

For more information about tape drives and libraries, refer to:

- *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946
- *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation,* SG24-6789
- *IBM TotalStorage 3494 Tape Library: A Practical Guide to Tape Drives and Tape Automation,* SG24-4632
- *IBM System Storage TS1120 and TS1130 Tape Drives and TS1120 Controller (3592 Models J1A, E05, E06, EU6, J70 and C06) Introduction and Planning Guide*, GA32-0555.

3.1 IBM System Storage TS1130 and TS1120 Tape Drive

The IBM System Storage TS1130 Tape Drive (machine type 3592, Model E06) represents the third generation of IBM 3592 Tape Drives. Made available in September 2008. The TS1130 is designed for applications that require high capacity and fast access to data across a wide range of environments. The IBM TS1130 Tape Drive features dual 4 Gbps Fibre Channel (FC) interfaces, has a native data rate of up to 160 MBps, and a native physical capacity of up to 1000 GB on the JB cartridge.

The IBM System Storage TS1120 Tape Drive (machine type 3592, Model E05) represents the second generation of IBM 3592 Tape Drives. Announced and made available in October 2005, the TS1120 is designed for applications that require high capacity and fast access to data across a wide range of environments. The IBM TS1120 Tape Drive features dual 4 Gbps Fibre Channel (FC) interfaces, has a native data rate of up to 100 MBps, and a native physical capacity of up to 700 GB on the JB cartridge. Figure 3-1 shows the front view of the IBM TS1120 Tape Drive. The 3592-E05 may be upgraded to a TS1130 drive model 3592-EU6. Section 3.1.2, “TS1120 characteristics” on page 47 discusses this upgrade.

Figure 3-1 TS1130 Model E05

Similar to the previous 3592-J1A, and 3592-E05, the 3592-E06 includes an RS-422 library interface port for communication with the IBM TS3400 Tape Library, IBM TS3500 Tape Library, or the IBM 3494 Tape Library. It directly attaches to Open Systems servers through a Fibre Channel interface. The 3592-E05 is supported for attachment to ESCON and FICON channels on System z servers through the following tape subsystems:

- TS1120 Tape Controller Model C06
- TS7700 Virtualization Engine (FICON only)[1][1]
- IBM 3592-J70 Tape Controller[1]
- IBM 3494 Models B10 and B20 Virtual Tape Server[1] (VTS) in J1A Emulation mode

Important: To use tape encryption on a System z server, the TS1120 has to attach to the host through a TS1120 Model C06 Tape Controller, the IBM 3592-J70 Controller, or the TS7700 Virtualization Engine.

The IBM TS1130 Tape Drive and IBM TS1120 Tape Drive can be installed in a rack, inside an IBM TS3400 Tape Library, inside an IBM TS3500 Tape Library, or in an already-installed IBM 3494 Tape Library, frame models L22, D22, or D24.

[1] Withdrawn from marketing

3.1.1 Tape data encryption support

The IBM TS1130 Tape Drive and IBM TS1120 Tape Drive supports encryption of data on a tape cartridge. All IBM TS1130 Tape Drives and IBM T1120 Tape Drives with a serial number of 13-65000 or higher are encryption-capable. For currently installed TS1120 drives with a serial number lower than 13-6500, a chargeable upgrade is available.

The encryption capability is implemented through tape drive hardware, and microcode additions and changes. All 3592 media, including WORM cartridges, can be encrypted. In addition, two applications are designed to support encryption: Tivoli Key Lifecycle Manager (TKLM) and Encryption Key Manager (EKM). Both TKLM and EKM uses standard key repositories on supported platforms. Either TKLM or EKM is required on a supported server to interface with the tape drive for encryption in a System-Managed Encryption or Library-Managed Encryption implementation. Refer to Chapter 2, "IBM tape encryption methods" on page 23. For a detailed discussion of the EKM program and the three encryption methods available with the IBM TS1120 and IBM TS1130 Tape Drive.

With encryption enabled, the access time to data on the tape drive increases with the bulk of the time spent in OPEN processing when writing from load point. Also, the tape drive unload time increases. This is because of the time necessary to retrieve, read, and write the encryption key.

Note: When attaching to a System z server, the IBM TS1120 Tape Controller Model C06 or the IBM 3592 Tape Controller Model J70 is required to support tape data encryption.

3.1.2 TS1120 characteristics

The IBM TS1120 Tape Drive maintains the same form factor and reliability specification of the previous 3592 Model J1A, as well as the features and technology enhancements introduced with the Model J1A. In addition, the 3592-E05 offers several enhancements over the previous model.

Features introduced with the 3592-J1A and incorporated in the 3592-E05 include:

- Digital speed matching
- Channel calibration
- High resolution tape directory
- Virtual Backhitch (Recursive Accumulating Backhitchless Flush or Non-Volatile Caching)
- Cartridge memory (CM)
- Streaming Lossless Data Compression (SLDC)
- Single Field Replaceable Unit (FRU)
- Error detection and reporting
- Statistical Analysis and Reporting System (SARS)
- Large internal buffer of 512 MB (3592 Model J1A is 128 MB)

Recording format

The TS1120 reads and writes 16 data tracks simultaneously as opposed to the 3592 Model J1A that reads and writes eight tracks at a time. TS1120 uses the Enterprise Format 2 (EFMT2 or E2) for data that is not encrypted or the Enterprise Encrypted Format 2 (EEFMT2 or EE2) for encrypted data, but can also read and write Enterprise Format 1 (EFMT1 or E1) used by the IBM 3592 Model J1A Tape Drive.

J1A emulation

The TS1120 has an emulation mode that enables it to emulate the previous 3592 Model J1A. If you attach a mix of TS1120 and 3592 Model J1A Tape Drives to a TS7700 Virtualization Engine, a VTS release 2.32.745.*xx* (or later), or a 3592 Model J70 or TS1120 Model C06 Tape Controller, the 3592-E05 drives will automatically operate in J1A Emulation mode, even when set to operate as native E05 drives. In this mode, the 3592-E05 drives read and write only in E1 format at the J1A performance and capacity ratings.

Note: The IBM TS1120 Tape Drive cannot be encryption-enabled while running in J1A Emulation mode.

Upgrading TS1120 to TS1130

If your TS1120 is still under warranty or a service contract you can upgrade it to a 3592-EU6.Refer to Figure 3-2. This involves replacing the drive brick (on the left side) but preserving the drive canister and electronics (on the right side).

Figure 3-2 3592 Drive and canister assembly)

Advantages of upgrading TS1120 to TS1130 (3592-EU6)

Advantages include:

- The 3592-EU6 retains the same serial number as it had as a 3592-E05 (reconfiguring your EKM or TKLM is not necessary).
- Capacity and performance of the 3592-EU6 are the same as a 3592-E06.
- Media formats supported are the same as the 3592-E06.

Advantages of buying a new TS1130 instead of upgrading an existing TS1120

Advantages include:

- New warranty
- Ethernet service port
- Old TS1120 can still write E1 format cartridges if necessary

Host attachment

Each IBM TS1120 Tape Drive comes with two independent 4 Gbps Fibre Channel ports for attachment to multiple servers or a single server with redundancy. The IBM TS1120 Tape

Drive attempts to connect at 4 Gbps but auto-negotiates down to 2 Gbps and then 1 Gbps if the system or switch to which it is connected cannot support 4 Gb.

In an open systems environment, you can connect different systems to the two Fibre Channel ports and use the drive alternatively from each system, but you cannot share a drive between an open systems and a System z environment. Note the following information:

- Open Systems attachment

  A TS1120 can attach to IBM System i®, i5, iSeries®, AS/400®, System p®, p5, pSeries®, RS/6000®, RS/6000 SP systems, System z9® and System z servers, System x® and xSeries®, Netfinity®, and non-IBM servers, workstations, and personal computers that support Fibre Channel interfaces.

- System z attachment

  In a System z environment, the IBM TS1120 Tape Drives do not attach directly to the host. They either attach as back-end drives to a TS7700 Virtualization Engine or a VTS Model B10 or B20, or they attach to a TS1120 Model C06 or an 3592 Model J70 Tape Controller.

For the latest details about specific hardware, software, and Fibre Channel support for the IBM TS1120 Tape Drive, refer to the following Web site:

`http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_interop.pdf`

For the latest information about applications and the levels that support TS1120 Tape Drives, refer to the Independent Software Vendor (ISV) matrixes at:

`http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_isv_matrix.pdf`

For supported host bus adapters, refer to:

`http://www.ibm.com/systems/support/storage/config/hba/index.wss`

For additional information about TS1120, refer to *IBM System Storage TS1120 Tape Drive and Controller Introduction and Planning Guide,* GA32-0555.

3.1.3 TS1130 characteristics

The IBM TS1130 Tape Drive maintains the same form factor and reliability specification of the previous TS1120, as well as the features and technology enhancements introduced with the TS1120. In addition, the 3592-E06 offers several enhancements over the previous model. If you have a TS1120 that is still under warranty or a service contract, it may be upgraded to a 3592-EU6. This drive will have all the characteristics of a 3592-E06 with the exception that it is not eligible for any further upgrade and does not contain an Ethernet service port.

The TS1130 characteristics include:

- 160 MBps (up to 350 MBps with compression)
- 1000 GB uncompressed using JB or JX media
- 650 GB uncompressed using JA or JW media
- 128 GB uncompressed using JJ or JR media
- Reads but does not write E1 formatted media
- MES Upgrade from TS1120 to 3592-EU6

Recording Format

The TS1130 reads and writes 16 data tracks simultaneously as opposed to the 3592 Model J1A that reads and writes eight tracks at a time. TS1130 uses the Enterprise Format 3 (EFMT3 or E3) for data that is not encrypted or the Enterprise Encrypted Format 3 (EEFMT3 or EE3) for encrypted data, Enterprise Format 2 (EFMT2 or E2) for data that is not encrypted

or the Enterprise Encrypted Format 2 (EEFMT2 or EE2) for encrypted data, but can also read Enterprise Format 1 (EFMT1 or E1) used by the IBM 3592 Model J1A Tape Drive.

J1A emulation

The TS1130 does not do J1A emulation. The TS1130 can still read E1 formatted media.

Host attachment

Each IBM TS1130 Tape Drive comes with two independent 4 Gbps Fibre Channel ports for attachment to multiple servers or a single server with redundancy. The IBM TS1130 Tape Drive attempts to connect at 4 Gbps but auto-negotiates down to 2 Gbps and then 1 Gbps if the system or switch to which it is connected cannot support 4 Gb.

In an Open Systems environment, you can connect different systems to the two Fibre Channel ports and use the drive alternatively from each system, but you cannot share a drive between an Open Systems and a System z environment. A better approach is to connect each port to an independent fabric for redundancy and zone the fabric so both open systems hosts can access both ports. When combined with IBM device driver on most platforms this will provide both load balancing and path failover. For more information, see the *IBM TotalStorage Tape Device Drivers Installation and User's Guide*, GC35-0154, available at the following ftp site:

`ftp://ftp.software.ibm.com/storage/devdrvr/`

- Open Systems attachment

  A TS1130 can attach to IBM System i, i5, iSeries, AS/400, System p, p5, pSeries, RS/6000, RS/6000 SP systems, System z9 and System z servers, System x and xSeries, Netfinity, and servers that are not IBM, workstations, and personal computers that support Fibre Channel interfaces.

- System z attachment

  In a System z environment, the IBM TS1130 Tape Drives do not attach directly to the host. They either attach as back-end drives to a TS7700 Virtualization Engine or a VTS Model B10 or B20, or they attach to a TS1120 Model C06 or an 3592 Model J70 Tape Controller.

For the latest details about specific hardware, software, and Fibre Channel support for the IBM TS1120 Tape Drive, refer to the following Web site:

`http://www.ibm.com/systems/storage/tape/ts1130/index.html`

For the latest information about applications and the levels that support TS1130 Tape Drives, refer to the Independent Software Vendor (ISV) matrixes at:

`http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_isv_matrix.pdf`

For supported host bus adapters, refer to:

`http://www.ibm.com/systems/support/storage/config/hba/index.wss`

For additional information about TS1130, refer to *IBM System Storage TS1120 and TS1130 Tape Drives and TS1120 Controller Introduction and Planning Guide 3592 Models J1A, E05, E06, EU6, J70 and C06,* GA32-0555.

3.1.4 3592 cartridges and media

The TS1130 uses Enterprise Format 3 (E3) recording technology. The TS1130 also reads and writes Enterprise Format 2 (E2) and reads Enterprise Format 1 (E1). A JA cartridge formatted in E3 has an uncompressed capacity of 640GB. The TS1120 uses Enterprise

Format 2 (E2) recording technology. A JA cartridge formatted in E2 has an uncompressed capacity of 500 GB. The 3592 Model J1A uses Enterprise Format 1 (E1). A JA cartridge formatted in E1 has an uncompressed capacity of 300 GB. When emulating the J1A, the TS1120 also uses the E1 format to provide a native capacity of 300 GB on a data cartridge. Though the 3592-J1A cannot read or write using E2 or E3, it recognizes E2 or E3 and rejects cartridges formatted in E2 or E3 with an error message indicating that E2 or E3 is not supported. It also can reformat E2 or E3 formatted cartridges using E1.

Media types

The TS1120 and TS1130 use six media cartridge types (JA, JB, JJ, JR, JW, and JX) and the 3592-J1A uses four media types (JA, JJ, JR, and JW). All six cartridge types contain the same dual coat advanced particle media. Capacity on four of these media types depends on whether it is used by model 3592-J1A, 3592-E05, 3592-EU6 or 3592-E06.

Table 3-1 shows the media types and capacity options available with 3592 Tape Drives.

Table 3-1 IBM TotalStorage Enterprise 3592 media types

Name	Media type	Usable length	Native capacity 3592-J1A (E1 format)	Native capacity 3592-E05 emulating J1A (E1 format)	Native capacity 3592-E05 (E2 format)	Native capacity 3592-E06 or 3592-EU6 (E3 format)	Data Facility Storage Management Subsystem (DFSMS)
Data	JA	570 m	300 GB	300 GB	500 GB	640 GB	MEDIA5
Extended Data	JB	785 m	N/A	N/A	700 GB	1000 GB	MEDIA9
Economy	JJ	204 m	60 GB	60 GB	100 GB	128 GB	MEDIA7
WORM	JW	570 m	300 GB	300 GB	500 GB	640 GB	MEDIA6
Economy WORM	JR	204 m	60 GB	60 GB	100 GB	128 GB	MEDIA8
Extended WORM	JX	785 m	N/A	N/A	700 GB	1000 GB	MEDIA10

Figure 3-3 on page 52 shows four of the six media types from left to right: Economy WORM, WORM, Economy, and Data. The WORM cartridges pictured on the left have a platinum color shell, and the Read/Write (R/W) cartridges on the right have a black shell. The write-protect tab, door, and label for the full length cartridges (both WORM and R/W) are dark blue. The write protect tab, door, and label for the economy (short length) cartridges are light blue.

Figure 3-3 IBM System Storage Enterprise 3592 WORM and R/W cartridges

Labels

The 3592 cartridges use a new media label to describe the cartridge type. Figure 3-4 shows a 3592 cartridge with a JA label.

Figure 3-4 View of the 3592 cartridge label

The VOLSER consists of six characters, which are left-aligned on the label. Fewer than six characters are possible, but hardly ever used. The media type is indicated by the seventh and eighth characters. For optimal library performance make sure your labels adhere to the guidelines found in *Label Specification for IBM 3592 Cartridges when used in IBM Libraries:*

`http://www.storage.ibm.com/media/tapecartridges/index.html`

Under Enterprise storage media, select 3592 tape cartridges. Under Related information, select Barcode Label Specification for use with 3592 Tape Media. Under Content, select the PDF file to access the document. You can also contact your IBM Marketing Representative for this specification.

3592 WORM functionality

The IBM 3592 Write Once Read Many (WORM) data cartridges provide unalterable, non-rewritable tape media for long-term record retention. WORM characteristics include:

- WORM cartridges are available with 1000 GB, 640 GB, or 128 GB native capacity for E3 format (3592-E06 or 3592-EU6), with 700 GB, 500 GB, or 100 GB native capacity for E2 format (3592-E05) and with 300 GB or 60 GB native capacity for E1 format.
- Non-reversible screws are used to secure the media housing.

- WORM and R/W cartridges can be intermixed within the same IBM TotalStorage Enterprise Automated Tape Library 3494, IBM System Storage TS3400 or TS3500 Tape Library, or StorageTek™ Automated Cartridge System (ACS™) solutions.
- When the drive senses that a cartridge is a WORM cartridge, the microcode prohibits changing or altering user data that is already written on the tape. The licensed internal code keeps track of the last point on the tape to which data can be appended by means of an overwrite-protection pointer stored in the cartridge memory (CM).
- Each WORM cartridge is identified using a unique cartridge identifier (UCID).

The WORM cartridge is geometrically identical to a R/W cartridge and uses the same rewritable media formulation. The servo format, which is mastered onto the tape at manufacturing, is different for WORM cartridge types, however. The WORM aspect comes not from any inherent non-reversible media characteristic, such as permanent WORM on optical media, CD-R, or ablative optical WORM, but rather by the way that the 3592 drive's microcode handles a WORM cartridge. The drive microcode does not allow writing over the data or erasure of previously written user data, such as records or file marks, however, appending new data following existing data is supported.

Final destruction of WORM cartridges

A WORM cartridge cannot be reused after it is written to, and thus, when it is no longer of use, it needs to be destroyed. If the WORM cartridge has sensitive data, it needs to be bulk-erased (which erases everything on the tape, including the mastered servo pattern, rendering it useless), before it is sent out to the landfill or the incinerator.

Tape encryption is fully supported for WORM cartridges. You can rekey WORM cartridges, because the wrapped key structures are not stored in the data area.

Note: Instead of physically deleting or destroying the data, consider rekeying WORM cartridges and deleting the key after the rekeying operation.

3.2 IBM System Storage TS1120 Tape Controller

The IBM System Storage TS1120 Tape Controller is the replacement to the IBM 3592 Model J70 Tape Controller. The IBM TS1120 Tape Controller is designed to attach to ESCON and FICON channels on System z servers or through a FICON/FC switch with appropriate levels of system software. Figure 3-5 shows a front view of the TS1120 Model C06.

Figure 3-5 IBM System Storage TS1120 Tape Controller

The TS1120 Tape Controller is supported in the following configurations:

- TS3500

  Attachment is the same as the 3592-J70 using the 3953 Tape Frame Model F05. An intermix of these controllers is allowed in the 3953-F05 expansion frames.

- IBM 3494 Tape Library

  The IBM TS1120 Tape Controller resides in an IBM 3952 Tape Frame Model F05.

- TS3400 Tape Library

  The IBM TS1120 Tape Controller resides in an industry standard 19 inch rack.

- Silo

  The IBM TS1120 Tape Controller resides in a rack or in a 3952-F05 frame (replaces 3590-C10 frame). This controller is then connected to the 3592 drives residing in a 3592-C20 frame.

- Stand-alone

  The IBM TS1120 Tape Controller resides in a rack or in a 3952-F05 frame. This controller is then connected to the 3592 drives residing in a rack.

Note: The IBM TS1120 Tape Controller supports 3592-J1A or TS1120 Tape Drives, but not IBM 3590 Tape Drives.

3.2.1 IBM TS1120 Tape Controller characteristics

The TS1120-C06 offers ESCON and FICON attachment to 3592-J1A and 3592-E05 drives. IBM 3592-J1A and TS1120 Tape Drives can be intermixed behind a single controller, but the 3592-E05 drives operate in 3592-J1A Emulation mode.

Note: To support tape data encryption, the IBM TS1120 Tape Drive must run in E05 mode, not in J1A Emulation mode. Therefore,To use tape data encryption, do not intermix 3592-J1A and TS1120 Tape Drives behind the same Model C06 Tape Controller or Model J70 Tape Controller. With System Managed Storage (SMS) tape support, intermixing encryption-enabled and non-encryption-enabled 3592 Model E05 drives also is not supported behind the same control unit.

Enhancements over the 3592-J70 that have been incorporated into the IBM TS1120 Tape Controller include:

- Up to four 4 Gbps FICON attachments or 2 Gbps for 3592 Model J70 controllers
- Up to eight ESCON attachments
- Support for an intermix of ESCON and FICON attachments
- Up to sixteen attached 3592-E05 (or 3592-J1A) Tape Drives
- Two 4 Gbps Fibre Channel adapters for attaching 3592 Tape Drives or switches
- Support for 3592 drive hot swap capabilities
- Support for capacity scaling and segmentation with the 3592 Tape Drives

- Support for WORM capabilities with the 3592 Tape Drives
- Support for call home communication
- Support for outboard search interface, for increased performance of certain applications.Currently, DFSMShsm audit is the only application written to take advantage of this support.

In a system-managed (SMStape) environment, an intermix of encryption-enabled and non-encryption-enabled TS1120 Model E05 drives is not supported behind the same IBM TS1120 Tape Controller.

3.2.2 IBM TS1120 Tape Controller encryption support

The IBM TS1120 Tape Controller and its predecessor, the 3592-J70, support System-Managed Encryption (SME). SME requires an Encryption Key Manager (EKM) to manage and provide keys.

When the IBM TS1120 Tape Controller attaches to z/OS, two ways of connecting to the EKM are supported: The controller can communicate with the EKM residing within z/OS through the FICON or ESCON path, or it can communicate with the EKM through a TCP/IP connection.

When the controller communicates with the EKM through the FICON or ESCON path, we call this *in-band encryption*, because data and keys travel through the same path. When the controller externally connects to an EKM through TCP/IP, we call this *out-of-band encryption*, because data and keys are transferred through separate paths. When using out-of-band encryption, the EKM can reside on any supported platform. Operating systems z/VM, z/VSE, and z/TPF require out-of-band encryption.

If TS1120 drives attached to a TS1120 controller reside in a TS3500, TS3400, or 3494 Tape Library, you can encryption-enable the drives through the library's user interface. A service support representative (SSR) has to encryption-enable stand-alone drives and drives installed in a Silo.

For details about System-Managed Encryption, refer to 2.3.1, "System-Managed Encryption" on page 33.

3.2.3 Installation with an IBM TS3500 Tape Library

To support the TS1130, TS1120 or 3592-J1A drives in an IBM TS3500 Tape Library, the IBM TS1120-C06 Tape Controller is installed in an external frame, the 3953 Model F05 Frame. The two versions of the 3953 Tape Frame are the base frame and the expansion frame. Both frames have the same F05 model number and can contain one to three controllers depending on whether the frame is a base frame or an expansion frame as shown in Table 3-2.

Table 3-2 TS1120 Tape Controllers in 3953 Tape Frames

Frame	Attachments
3953 Tape Frame Model F05 (base)	Zero to one IBM TS1120 Tape Controllers
3953 Tape Frame Model F05 (expansion)	One to three IBM TS1120 Tape Controllers

A fully configured 3953 Tape System (3953-F05 Frames and 3953-L05 Library Manager) can have up to sixteen subsystems attached (tape controllers, TS7700 Virtualization Engines, or

Virtual Tape Servers (VTSs)). If the maximum of two TS7700 Virtualization Engines (or VTSs) is attached, you can only have fourteen TS1120 controllers.

Figure 3-6 shows a sample of a TS1120-C06 installation and configuration in a 3953-F05 base frame with two 3953-F05 expansion frames.

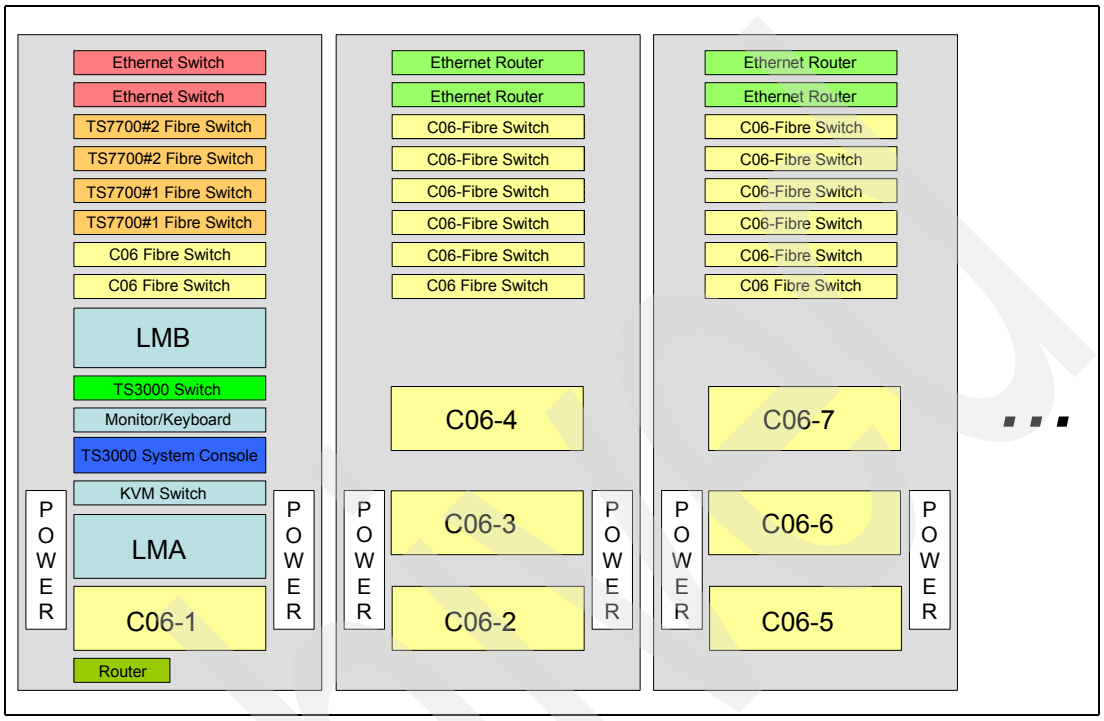


Figure 3-6 Sample configuration of IBM TS1120 Model C06 Tape Controller in 3953 frames

Drive attachment

The TS1120-C06 can attach up to sixteen TS1130, TS1120, or 3592-J1A Tape Drives in a TS3500 library. For this attachment, there must be at least one Fibre Channel switch per tape controller in a 3953 Model F05 Frame to route data between the controller and its associated tape drives. For reliability, two Fibre Channel switches can be associated with one controller. The tape drives connected to a particular controller do not need to reside in the same frame. They can be spread across multiple frames in the IBM TS3500 Tape Library frames as shown in Figure 3-7 on page 57. In this picture, one IBM TS1120-C06 Tape Controller has 16 tape drives attached: four tape drives in each of four separate frames. Another controller also has 16 drives attached, 12 of which reside in one frame and 4 in another frame.

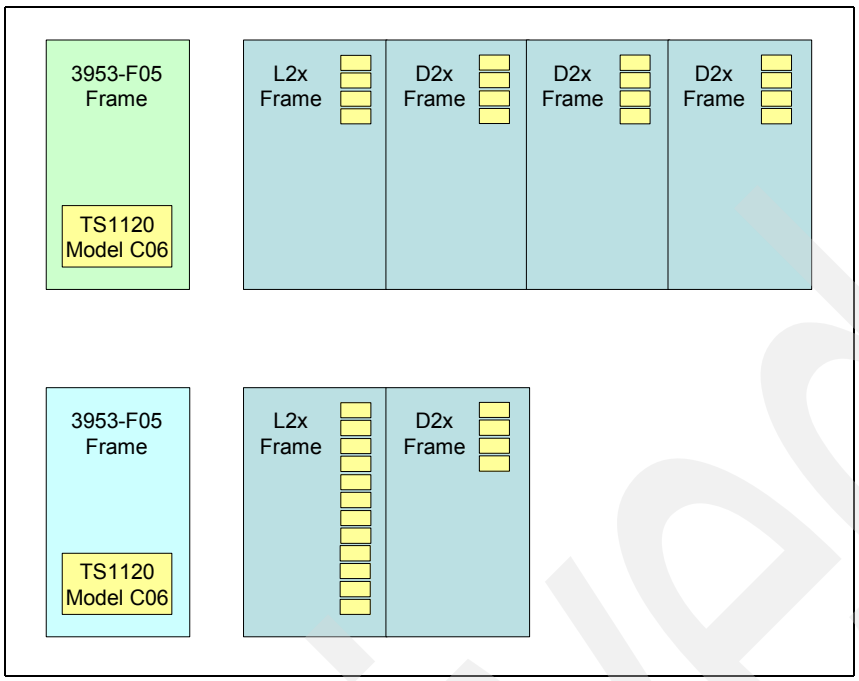


Figure 3-7 Examples for drive attachment to an IBM TS1120 Tape Controller

The TS1120-C06 supports an intermix of 3592-J1A and 3592-E05; however, this intermix results in the 3592-E05 running in J1A Emulation mode at J1A capacity and performance ratings.

3.2.4 Installation with an IBM TS3400 Tape Library

The TS1120 controller supports attachment of up to seven TS3400 libraries. Because each TS3400 can house up to two TS1130 or TS1120 drives, you can attach a maximum of fourteen drives installed in TS3400 libraries to one IBM TS1120 Model C06 Tape Controller. If an IBM TS1120 Tape Controller attaches to drives in a TS3400, all other drives attached to this controller also must reside in TS3400s.

Both the controller and the libraries need to be rack-installed. The TS3400 Tape Library does not support 3592-J1A drives.

Figure 3-8 on page 58 shows the maximum configuration of TS3400 Tape Libraries attached to an IBM TS1120 Tape Controller.

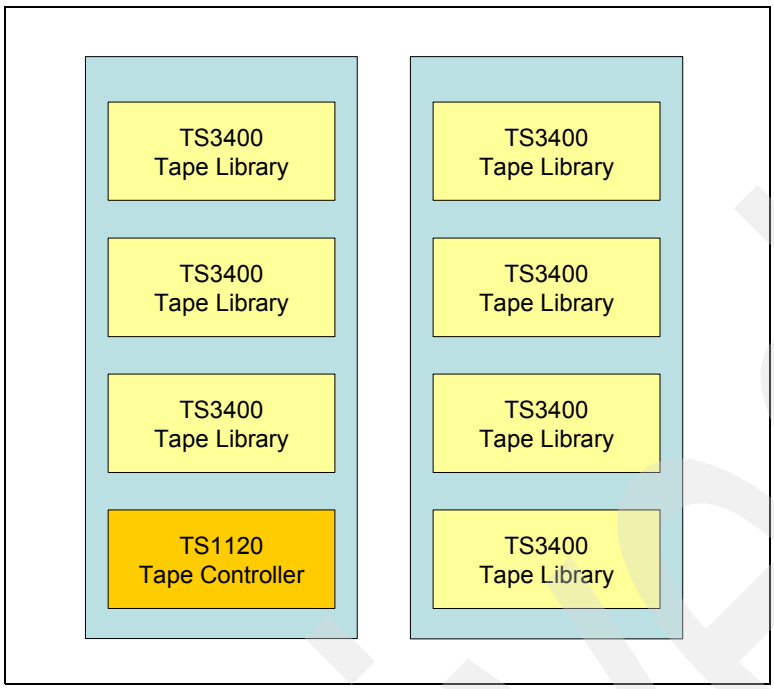


Figure 3-8 Maximum configuration of TS3400 attached to IBM TS1120 Tape Controller

3.2.5 Installation with an IBM 3494 Tape Library

When using a C06 controller with an IBM 3494 Tape Library, the controller resides in a 3952 Tape Frame that is detached from the library.

There are two versions of the 3952 Tape Frame for the TS1120 Model C06: the 3494 attachment frame and the silo-attached frame. A maximum of three IBM TS1120 Model C06 Tape Controllers can be installed in the frame, as detailed in Table 3-3.

Table 3-3 TS1120 Tape Controllers in 3952 Tape Frames

Frame	Attachments
3952 Tape Frame Model F05 (3494 attachment)	One to three IBM TS1120 Tape Controllers
3952 Tape Frame Model F05 (silo attachment)	One to three IBM TS1120 Tape Controllers

The TS11200 Model C06 connects to the library through a 3494 D24 or 3494 D22 frame. The 3494 D24 frame or 3494 D22 frame contains the Fibre Channel switches that the controller uses to communicate with the TS1130, TS1120 or 3592-J1A drives, the Ethernet router through which the controller communicates with the Library Manager, and up to 12 IBM TS1130, TS1120 or 3592-J1A Tape Drives. Additional drives, up to a total of 16 per TS1120 controller, can be connected in an adjacent 3494 D22 frame or 3494 L22 frame.

Table 3-4 on page 59 lists the frame capacities for drives and controllers.

Table 3-4 The 3494 maximum frame capacities for drives and controllers

Frame	Number of drives and tape controllers
3494 Model L22	Up to four 3592 Tape Drives and no controller **Note:** If a Model L22 frame is installed with the adjacent frame feature FC4065, FC4075, FC4085, or FC4086, up to four 3592 or TS1120 drives are supported in the L22 frame.
3494 Model D22	Up to twelve 3592 Tape Drives and no controller **Note:** If a Model D22 frame is installed with the adjacent frame feature FC4065, FC4075, FC4085, or FC4086, the maximum number of attached 3592 drives is eight.
3494 Model D24	Up to eight 3592 Tape Drives and one 3592 Model J70 or 3590 Model A60 controller

3.2.6 IBM TotalStorage 3592 Model J70 Tape Controller

The Model J70 controller is the predecessor of the TS1120 Model C06. It has been withdrawn from marketing, but existing installations can upgrade the microcode level of the J70 to support tape data encryption when encryption-enabled TS1120 Tape Drives are attached. See Figure 3-9.

The 3592 Model J70 controller supports TS1130, TS1120, 3592-J1A, and 3590 drives in a 3494 Tape Library and TS1130, TS1120 and 3592-J1A drives in a TS3500 library.

To use tape encryption, all tape drives behind the J70 must be TS1130 Model E06, TS1130 Model EU6 or TS1120 Model E05 drives, and in the system-managed (SMStape) environment, intermixing encryption-enabled and non-encryption-enabled TS1120 Model E05 drives is not supported behind the same tape controller.

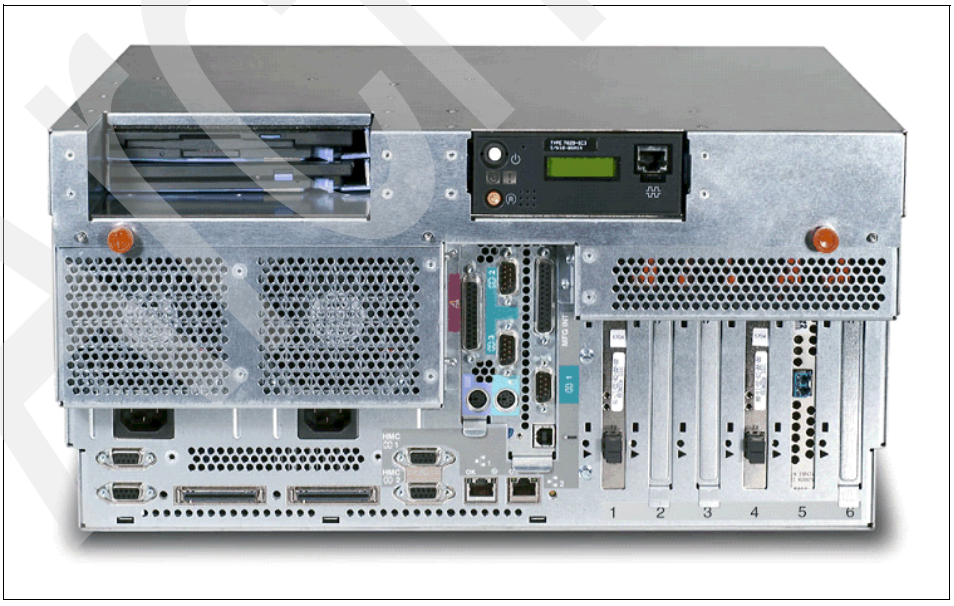


Figure 3-9 IBM 3592 Model J70 Tape Controller

3.3 IBM Virtualization Engine TS7700

The IBM System Storage Virtualization Engine TS7700 is a member of the IBM TS7000 Virtualization Family. It represents the fourth generation of IBM Tape Virtualization for mainframe systems and replaced the highly successful IBM TotalStorage Virtual Tape Server (VTS).

The TS7700 Virtualization Engine is designed to provide improved performance and capacity to help lower the total cost of ownership for tape processing. It introduces a new modular, scalable, high-performing architecture for mainframe tape virtualization. It integrates the advanced performance, capacity, and data integrity design of the IBM TS1120 and IBM TS1130 Tape Drives, with high-performance disk and a new advanced IBM System p server to form a storage hierarchy managed by robust storage management firmware with extensive self-management capabilities.

The two types of TS7700 at the time of this writing are the TS7720 and the TS7740. The TS7720 is a disk-only virtual tape system with up to 70 TB of disk cache. Because it does not attach to physical tape it does not take advantage of tape encryption. The TS7740 has up to 14 TB of disk cache and attaches to both IBM TS1120 or TS1130 Tape Drives.

The TS7700 Virtualization Engine integrates the following components into the virtual tape solution:

- One IBM Virtualization Engine TS7740 Server Model V06
- One IBM Virtualization Engine TS7740 Cache Controller Model CC6
- Up to three IBM Virtualization Engine TS7740 Cache Drawers Model CX6

Figure 3-10 shows the IBM Virtualization Engine TS7700 and a schematic layout of its components.

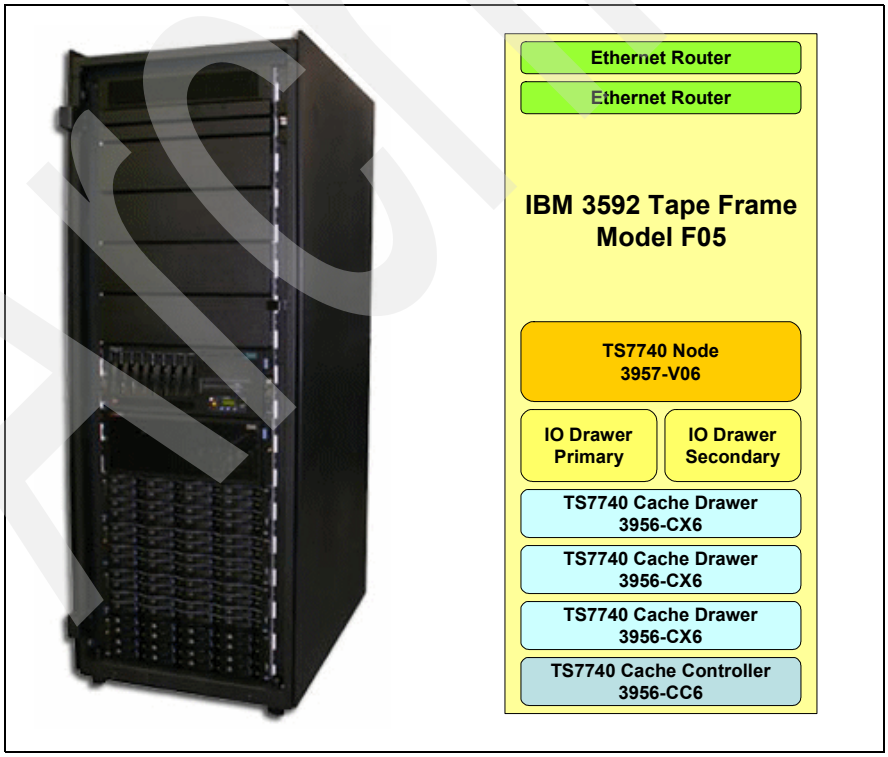


Figure 3-10 IBM Virtualization Engine TS7700

The TS7700 Virtualization Engine attaches to System z hosts through FICON connections, either directly or through FICON directors. It supports back-end drives in a TS3500 or 3494 tape library. When the TS7700 attaches to a TS3500 library, an external library manager is no longer required as of licensed internal code 8.5.0.xx

TS7700 characteristics

The TS7700 offers a new standards-based management interface and enhanced statistical reporting, compared to the VTS.

Important characteristics of theTS7700 are summarized here:

- The TS7700 Virtualization Engine utilizes outboard policy management to manage physical volume pools, cache management to control selective dual copy, dual copy across a grid network, and copy mode control.
- The TS7740 Server provides host connections of up to four FICON channels and connections to the tape library and tape drives for back-end tape processing.
- A TS7700 with Grid Communication features can be interconnected with up to two other TS7700s to provide peer-to-peer copy capability between Virtualization Engines for tape using TCP/IP network connections.
- The TS7740 Cache, comprised of the TS7740 Model CC6 and the TS7740 Model CX6, provides from 1 TB to 14 TB of uncompressed tape volume cache (TVC) capacity.
- Each TS7700 supports up to a maximum of 256 3490E virtual tape drives and up to 1,000,000 logical volumes, each with a maximum capacity of 1.2 GB to 12 GB (assuming 3:1 compression and using the 400 to 4000 MB volume sizes).
- The TS7700 offers a new standards-based Management Interface (MI) and enhanced statistical reporting, compared to the VTS.
- On demand features for cache capacity and performance allow for a lower cost entry configuration with the option to grow with minimal impact on operations.
- The Copy Export Function allows for export of secondary copies of volumes for Disaster Recovery purposes. The export operates on physical cartridge pools. The primary copy of exported volumes stays resident in the TS7700.
- The TS7700 supports the TS3500 and the 3494 tape libraries. When it attaches to a TS3500 tape library, an external Library Manager is required.
- You can attach from four to sixteen TS1130, TS1120 or 3592-J1A Tape Drives to a TS7700. The drives can reside either in a TS3500 or a 3494 tape library. The TS7700 also supports a mix of TS1130 and TS1120 or TS1120 and 3592-J1A drives. In a mixed TS1120 and 3592-J1A configuration, the TS1120 drives run in J1A Emulation mode. TS1120 drives running in J1A Emulation mode do not support encryption.
- The TS7700 supports System-Managed Encryption. To utilize the encryption capability of TS1120 and TS1130 drives, all drives attached to the TS7700 must be TS1130 or encryption-enabled TS1120 drives.

The following operating systems support attachment of the TS7700 Virtualization Engine:

- IBM z/OS V1.7 or later
- IBM z/VM V5.1 or later
- IBM z/VSE V3.1.2 or later
- IBM z/TPF 1.1 or later
- IBM TPF 4.1 or later

Earlier releases might support the basic functionality of the TS7700 Virtualization Engine.

For detailed information about the IBM Virtualization Engine TS7700, refer to *IBM System Storage Virtualization Engine TS7700*, SG24-7312.

Encryption support

Encryption on the TS7700 is controlled on a storage pool basis. DFSMS *Storage Group* and *Management Class* constructs are used to control the use of primary and secondary storage pools for logical volumes, through mapping in the Library Manager, resulting in an indirect form of encryption policy management. The storage pools, which were originally created for the management of physical media, have been enhanced to include encryption characteristics. You can set up storage pools for encryption through the TS7700 Management Interface (MI) and then direct primary and secondary copies of volumes to these pools by using the Storage Group and Management Class constructs.

In z/OS, automatic class selection (ACS) routines assign Storage Group and Management Class to volumes dynamically. In non-z/OS environments, you can set up encryption policies by assigning Storage Group and Management Class statically to ranges of volume serials using the Library Manager user interface.

Encryption key labels are assigned using the Management Interface on a per-storage-pool basis. For more information, refer to:

- *IBM Virtualization Engine TS7700: Tape Virtualization for System z hosts*, SG24-7312
- *IBM Virtualization Engine TS7700 Series Encryption Overview*, which is available at:

  `ftp://ftp.software.ibm.com/storage/Encryption/Docs/TS7700_Encryption_Support_V11.pdf`

3.4 IBM LTO Ultrium tape drives and libraries

In this section, we give an overview of LTO Ultrium tape drive and media technology and describe IBM LTO Ultrium tape drives and automated tape libraries. We describe the following products:

- IBM System Storage TS2240 Tape Drive Express Model
- IBM System Storage TS2340 Tape Drive Express Model
- IBM System Storage TS1040 Tape Drive
- IBM System Storage TS2900 Tape Autoloader
- IBM System Storage TS3100 Tape Library
- IBM System Storage TS3200 Tape Library
- IBM System Storage TS3310 Tape Library

The System Storage TS3500 Tape Library also supports IBM LTO Ultrium tape drives, but it is not exclusively an LTO tape library. It supports the installation of IBM LTO Ultrium, IBM System Storage TS1120 Tape Drives and IBM System Storage TS1130 Tape Drives. Using IBM TS1120 or IBM TS1130 Tape Drives, the TS3500 attaches not only to Open Systems, but also to System z hosts. We discuss the TS3500 Tape Library in a separate section (3.6, “IBM System Storage TS3500 Tape Library” on page 77).

Note: For the remainder of this book, we use the term LTO as a generic term for various generations of the LTO Ultrium tape drives. We use LTO4 as a generic term for IBM LTO Ultrium Generation 4 drives. These are the IBM System Storage TS2240 tape drive, IBM System Storage TS2340 tape drive, the IBM System Storage TS1040 tape drive, the IBM LTO Ultrium 4 tape drives installed in the System Storage TS2900 Autoloader, and the System Storage TS3100, TS3200, and TS3310 tape libraries.

3.4.1 LTO overview

The LTO Program was formed in 1997 by IBM, Hewlett-Packard (HP), and Seagate. The three companies, HP, IBM, and Quantum (the successor to Seagate), jointly oversee the development and road map of Linear Tape-Open (LTO) technology.

The LTO technology objective was to establish new open-format specifications for high capacity, high performance tape storage products for use in the midrange and network server computing environments and to enable superior tape product options.

LTO program cooperation goes beyond the initial three companies. LTO format specifications have been made available to all who want to participate through standard licensing provisions. LTO program technology has attracted a number of other industry leaders, so that LTO-specified products (tape drives and tape storage cartridges) will reach the market from multiple manufacturers, not just the Technology Provider Companies. This is critical to meeting an open market objective and is accomplished through open licensing of the technology.

Cooperation is also evident in the LTO program requirement that all products produced by licensees are technically certified annually. The primary objective of this certification is to help determine whether LTO format cartridges will be interchangeable across drives produced by different LTO Ultrium manufacturers. In other words, LTO compliant media from any vendor can be read and written in LTO compliant drives from any vendor.

All three consortium members (IBM, HP, and Quantum) are shipping LTO Ultrium products, and numerous other licensees are shipping hardware and media.

The Linear Tape-Open organization Web site is:

http://www.lto.org

For more information about LTO technology, refer to the *IBM System Storage Tape Libraries Guide for Open Systems*, SG24-5946.

The IBM LTO Web site is:

http://www.ibm.com/storage/lto

The LTO Ultrium road map (Figure 3-11 on page 64) shows the evolution of LTO technology. At the time of writing, IBM Ultrium generation 3 and 4 products are offered. The information in the road map is given as an indication of future developments by the three consortium members and is subject to change.

	Generation 1	Generation 2	Generation 3	Generation 4	Generation 5	Generation 6
Capacity (Native)	100 GB	200 GB	400 GB	800 GB	1.6 TB	3.2 TB
Transfer Rate (Native)	Up to 20 MB/s	Up to 40 MB/s	Up to 80 MB/s	Up to 120 MB/s	Up to 180 MB/s	Up to 270 MB/s
WORM	No	No	Yes	Yes	Yes	Yes
Encryption	No	No	No	Yes	Yes	Yes

Figure 3-11 LTO Ultrium road map

Important: Hewlett-Packard, IBM, and Quantum (the successor to Seagate) reserve the right to change the information in this migration path without notice.

3.4.2 LTO media

Each generation of LTO Ultrium tape drives uses its own cartridge. LTO drives generally provide backward read compatibility for the previous generations and read/write compatibility for the previous generation. For example, LTO4 drives can read and write in LTO3 format on LTO3 media. They can also read the LTO2 format from LTO2 media, but cannot write in LTO2 format. The generations include:

- LTO1 was the first generation of the LTO technology with an uncompressed tape capacity of 100 GB per cartridge.
- LTO2 is the second generation of the LTO technology with an uncompressed tape capacity of 200 GB per cartridge.
- LTO3 is the third generation of the LTO technology with an uncompressed tape capacity of 400 GB per cartridge. A WORM (write-once, read-many) version of the LTO3 cartridge is also available.
- LTO4 is the fourth generation of the LTO technology with an uncompressed tape capacity of 800 GB per cartridge. A WORM (write-once, read-many) version of the LTO4 cartridge is also available. LTO4 is the first LTO generation that supports encryption. Encryption on LTO drives requires the use of LTO4 media.

LTO cartridges are color-coded. The LTO Ultrium 1 data cartridge is black, the LTO Ultrium 2 data cartridge is purple, the LTO Ultrium 3 data cartridge is steel blue, and the LTO Ultrium 4 data cartridge is green. The third generation IBM WORM cartridge is a two-tone cartridge with a steel-blue top and a platinum (silver) bottom and the fourth generation WORM is a two-tone cartridge with a steel-green top and a platinum (silver) bottom.

WORM tape format

Beginning with LTO3, Write Once Read Many (WORM) functionality provides for non-erasable, non-rewritable operation with tape media and is designed for long-term tamper resistant record retention.

The IBM LTO3 specification for WORM includes the use of low-level encoding in the Cartridge Memory (CM), which is also mastered into the servo pattern as part of the manufacturing process. This encoding is designed to prevent tampering.

Data can be appended at the end of a WORM cartridge to which data was previously written, allowing the full use of the high capacity tape media.

LTO3 WORM cartridges can be used with any LTO3 tape drive with the appropriate microcode and firmware. LTO3 non-WORM and WORM cartridges can coexist in the same library.

The same description holds for the LTO4 WORM cartridges. They can be used by any LTO4 Tape Drive and can coexist with non-WORM cartridges. Additionally, the LTO4 drive can read and write WORM and non-WORM LTO3 cartridges.

Figure 3-12 shows IBM LTO3 and LTO4 media. The two-tone cartridges on the picture are LTO3 WORM media.

Figure 3-12 IBM LTO Ultrium 3 and IBM LTO Ultrium 4 media

Labels

The LTO cartridge label uses the barcode symbology of USS-39. A description and definition is available from the Automatic Identification Manufacturers (AIM) specification Uniform Symbol Specification (USS-39) and the ANSI MH10.8M-1993 ANSI Barcode specification.

The barcode string consists of a start character, eight alphanumeric characters, and the stop character. Quiet zones precede and follow the start and stop characters. The first six characters can be any combination of uppercase A-Z or 0-9 (for example, `ABC123`) to identify the cartridge volume. The last two characters are determined by the LTO cartridge media type (that is, “L” for LTO and “1” for tape cartridge generation or drive manufacturer unique identifier).

Note: No characters other than uppercase alpha A-Z or numeric 0-9 are allowed.

Human-readable characters are allowed provided that there is no conflict or interference with the automation code. Users can specify the format, colors, and location of the human-readable characters.

For optimal library performance make sure your labels adhere to the guidelines found in *Label Specification for IBM 3592 Cartridges when used in IBM Libraries*, located at:

http://www.storage.ibm.com/media/tapecartridges/index.html

Under Enterprise storage media, select 3592 tape cartridges. Under Related information, select Barcode Label Specification for use with 3592 Tape Media. Under Content, select the .pdf file to access the document. You can also contact your IBM Marketing Representative for this specification.

Figure 3-13 shows a barcode label for an LTO1 data cartridge.

Figure 3-13 LTO Ultrium 1 barcode label

3.4.3 IBM System Storage TS2240 Tape Drive Express Model

The IBM TS2240 Tape Drive is an external stand-alone or rack-mountable half high LTO4 drive. It is the entry point for the family of IBM LTO tape products and incorporates the latest generation of LTO technology. The TS2240 is suited to handle the backup, save and restore, and archival data storage needs of a wide range of small systems. See Figure 3-14 on page 67.

IBM TS2240 increases the native data rate to up to 120 MBps. With the use of the LTO4 data cartridge, it doubles the tape cartridge capacity to 800 GB uncompressed capacity (1600 GB with 2:1 compression).

The IBM TS2240 Tape Drive uses a 3 Gbps Serial-Attached SCSI (SAS) interface for connections to a wide spectrum of Open Systems servers. The TS2240 models attach to IBM System p, IBM System i, IBM System x, Microsoft® Windows, HP-UX, Sun Solaris, UNIX, and Linux servers.

The TS2240 is encryption-capable and supports Application-Managed Encryption on AIX, Windows Server 2003, Linux, and Solaris. Encryption requires the latest device drivers, which are available on the FTP download site:

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

Figure 3-14 IBM System Storage TS2240 Tape Drive Express Model

For more information about IBM TS2240 Tape Drive, see *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946.

3.4.4 IBM System Storage TS2340 Tape Drive Express Model

The IBM TS2340 Tape Drive is an external stand-alone or rack-mountable LTO4 drive. It is the entry point for the family of IBM LTO tape products and incorporates the latest generation of LTO technology. The TS2340 is suited to handle the backup, save and restore, and archival data storage needs of a wide range of small systems. See Figure 3-15.

IBM TS2340 increases the native data rate to up to 120 MBps. With the use of the LTO4 data cartridge, it doubles the tape cartridge capacity to 800 GB uncompressed capacity (1600 GB with 2:1 compression).

The IBM TS2340 Tape Drive Model L43 uses a SCSI Ultra160 LVD attachment. The Model S43 uses a 3 Gbps Serial-Attached SCSI (SAS) interface for connections to a wide spectrum of Open Systems servers. The TS2340 models attach to IBM System p, IBM System i, IBM System x, Microsoft Windows, HP-UX, Sun Solaris, UNIX, and Linux servers.

The TS2340 Model S43 with SAS interface is encryption-capable and supports Application-Managed Encryption on AIX, Windows Server 2003, Linux, and Solaris. Encryption requires the latest device drivers, which are available on the FTP download site:

`ftp://ftp.software.ibm.com/storage/devdrvr/`

Note: The IBM TS2340 Tape Drive is available as Model L43 with LVD/SCSI or as Model S43 with SAS interface. Only SAS-attached TS2340 Tape Drives support encryption.

Figure 3-15 IBM System Storage TS2340 Tape Drive Express Model

For more information about IBM TS2340 Tape Drive, see *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946.

3.4.5 IBM System Storage TS1040 Tape Drive

The IBM System Storage TS1040 is the IBM LTO Ultrium 4 Tape Drive for installation in the IBM TS3500 Tape Library. The drive mounts in the TS3500 Tape Library Models L53 or D53 and in previous Models L52, L32, D52, or D32.

The TS1040 increases the native data rate to up to 120 MBps. With the use of the LTO4 data cartridge, it doubles the tape cartridge capacity to 800 GB uncompressed capacity (1600 GB with 2:1 compression) in comparison to its predecessor, the TS1030 Tape Drive.

The drive includes a 4-Gbps Fibre Channel interface attachment.

The IBM TS1040 Tape Drive is encryption-capable and supports Library-Managed Encryption (LME) and System-Managed Encryption (SME) on a variety of Open Systems platforms. For a list of supported operating systems and host bus adapters (HBAs), refer to:

http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts3500_interop.pdf

The TS1040 also supports Application-Managed Encryption (AME) on AIX, Windows Server 2003, Linux, Solaris, and HP-UX.

Figure 3-16 shows the IBM TS1040 Tape Drive.

Figure 3-16 IBM System Storage TS1040 Tape Drive

3.4.6 IBM System Storage TS2900 Tape Autoloader

The IBM TS2900 Tape Autoloader is a single drive entry level desktop or rack-mounted unit (requiring one rack unit in an industry standard 19-inch rack). One half-high IBM LTO3 or LTO4 drive may be mounted in the TS2900. The TS2900 is positioned between a standalone tape drive and the TS3100 Tape Library. The TS2900 is well-suited to handle the backup, save and restore, and archival data storage needs of small to medium-sized environments. Up to nine cartridges may be mounted in the Autoloader at a time. See Figure 3-17.

Figure 3-17 IBM System Storage TS2900 Tape Autoloader

Encryption settings

The TS2900 supports Application-Managed Encryption by default when an LTO4 drive is installed. An additional feature, Transparent LTO Encryption, is required for System-Managed Encryption or Library-Managed Encryption. See Figure 3-18.

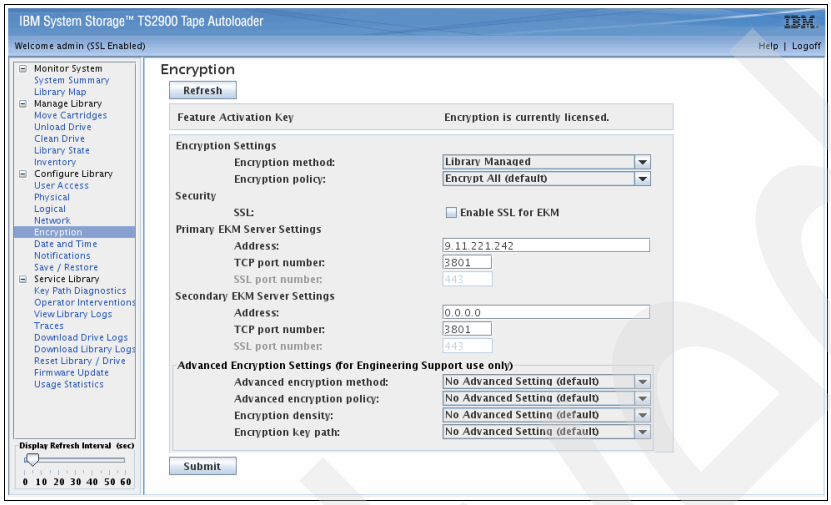


Figure 3-18 IBM TS2900 Tape Autoloader Encryption Settings

3.4.7 IBM System Storage TS3100 Tape Library

The IBM TS3100 Tape Library is a single or dual drive entry level desktop or rack-mounted unit (requiring two rack units in an industry standard 19 inch rack). It is the entry point for the family of IBM LTO tape library products and incorporates the latest generation of LTO technology. The TS3100 is well-suited to handle the backup, save and restore, and archival data storage needs of small to medium-sized environments. See Figure 3-21 on page 72.

The TS3100 supports either one IBM LTO3 full-high tape drive with a native capacity of 400 GB, two IBM LTO3 half-high tape drives with a native capacity of 400 GB, one IBM LTO4 Tape Drive with a native capacity of 800 GB, or up to two IBM LTO4 half-high tape drives with a native capacity of 800 GB. The IBM LTO4 Tape Drives are available with Fibre Channel (FC), 3 Gbps Serial Attached SCSI (SAS), or SCSI Ultra160 LVD attachment interface.

Note: Half High drives do not have Fibre Channel attachment. The TS3100 can be ordered with LTO3 drives with LVD/SCSI or FC interface or with LTO4 drives with LVD/SCSI, SAS, or FC interface. Only LTO4 drives with Fibre Channel or SAS attachment interface support encryption.

A total of 24 cartridges can be stored in two removable magazines (23 cartridges with I/O Station enabled). A single dedicated mail slot (I/O Station) is available for importing and exporting cartridges. Using LTO4 cartridges, the library has an uncompressed capacity of 19.2 TB (38.4 TB with 2:1 compression).

The barcode reader and the remote management unit (RMU) are standard features. The RMU provides an Ethernet port so that the library can be configured as a TCP/IP device in the network. Library status can be sent to the network as Simple Network Management Protocol (SNMP) traps. The IBM System Storage Tape Library Specialist enables network access through a Web browser. You can access all library Operator Panel functions using the Tape Library Specialist. The Specialist also provides the ability to configure logical libraries up to

the number of tape drives, providing a maximum capability of two logical libraries for the TS3100 with two half-high drives.

The TS3100 Tape Library attaches to IBM System p, IBM System i, IBM System x, Microsoft Windows, HP-UX, Sun Solaris, UNIX, and Linux servers.

The IBM TS3100 supports Library-Managed Encryption, System-Managed Encryption, and Application-Managed Encryption on SAS and Fibre Channel LTO4 drives using LTO4 media. Not all environments support all three encryption methods. For details about platform-specific support, refer to Chapter 4, “Planning for software and hardware” on page 91.

Three policy settings are available for Library-Managed Encryption on a TS3100:

- Encrypt All

  All cartridges in the library will be encrypted.

- Internal Label - Selective Encryption

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports these Internal Label Encryption Policies (ILEP).

- Internal Label - Encrypt All

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports these Internal Label Encryption Policies (ILEP).

You set these policies through the Web interface.

Figure 3-19 IBM System Storage TS3100 Tape Library Express Model

For more information about IBM TS3100 Tape Library, refer to *IBM System Storage Tape Libraries Guide for Open Systems,* SG24-5946.

3.4.8 IBM System Storage TS3200 Tape Library

The IBM TS3200 Tape Library is a midrange desktop or a rack-mounted unit (requiring four rack units in an industry standard 19-inch rack).

The TS3200 supports either two IBM LTO3 full-high tape drives with a native capacity of 400 GB, four IBM LTO3 half-high tape drives with a native capacity of 400 GB, two IBM LTO4 tape drives with a native capacity of 800 GB, four IBM LTO4 half-high tape drives with a native capacity of 800 GB, or a mix of IBM LTO3 and LTO4 full-high tape drives. The IBM LTO4 tape drives are available with Fibre Channel, 3 Gbps Serial Attached SCSI (SAS), or SCSI Ultra160 LVD attachment interface.

Note: Half High drives do not have Fibre Channel attachment. The TS3200 can be ordered with LTO3 drives with LVD/SCSI or FC interface or with LTO4 drives with LVD/SCSI, SAS, or FC interface. Only LTO4 drives with Fibre Channel or SAS attachment interface support encryption.

A total of 48 cartridges (45 with I/O Station enabled) can be stored in four removable magazines. Using LTO4 cartridges, the library has an uncompressed capacity of 38.4 TB (76.8 TB with 2:1 compression). A three-cartridge I/O Station is available for importing and exporting cartridges.

The barcode reader and the remote management unit (RMU) are standard features. The RMU provides an Ethernet port so that the library can be configured as a TCP/IP device in the network. Library status can be sent to the network as Simple Network Management Protocol (SNMP) traps. The IBM System Storage Tape Library Specialist enables network access through a Web browser. You can access all library operator panel functions by using the Tape Library Specialist. The Specialist also provides the ability to configure logical libraries up to the number of tape drives, providing a maximum capability of four logical libraries for the TS3200 with four half-high drives.

The IBM TS3200 tape library attaches to IBM System p, IBM System i, IBM System x, Microsoft Windows, HP-UX, Sun Solaris, UNIX, and Linux servers.

The IBM TS3200 supports Library-Managed Encryption, System-Managed Encryption, and Application-Managed Encryption on SAS and Fibre Channel LTO4 drives using LTO4 media. See Figure 3-20.

Figure 3-20 IBM System Storage TS3200 Tape Library Express Model

Three policy settings are available for Library-Managed Encryption on a TS3200:

- Encrypt All

  All cartridges in the library will be encrypted. If you have two or more drives and partition the library into two logical libraries, you can set separate encryption policies for the two logical libraries.

- Internal Label - Selective Encryption

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

- Internal Label - Encrypt All

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

You set these policies through the Web interface, as shown in Figure 3-21 on page 72.

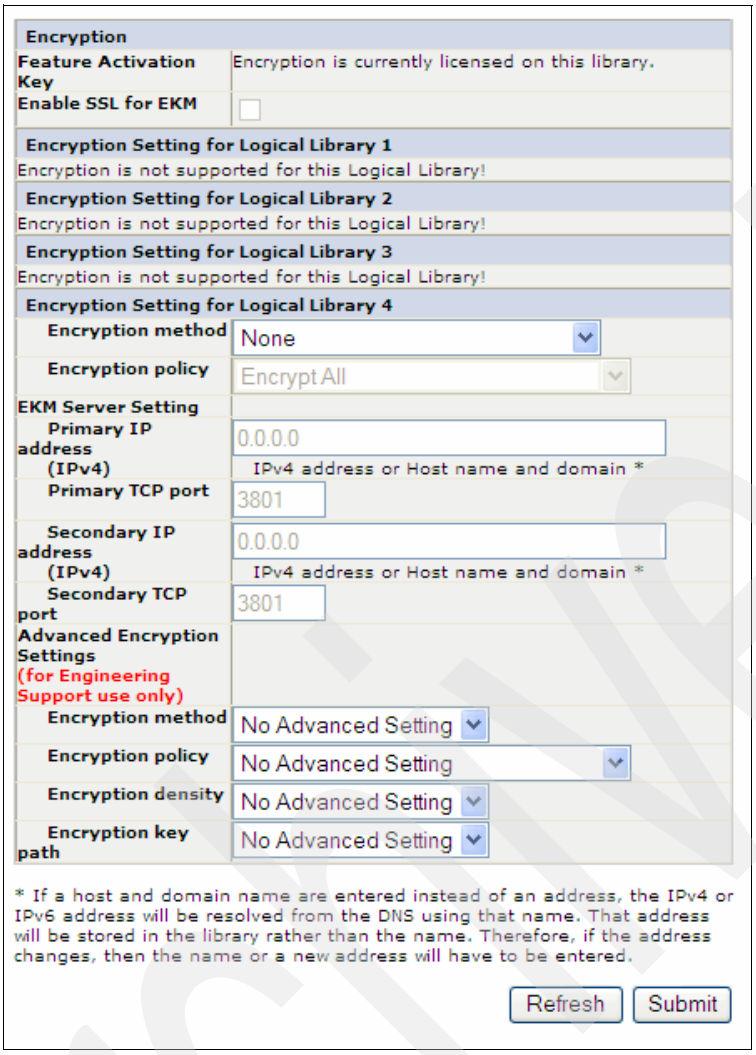


Figure 3-21 Sample TS3200 Encryption configuration panel

Not all environments support all three encryption methods. For details about platform-specific support, refer to Chapter 4, “Planning for software and hardware” on page 91.

For more information about IBM TS3200 Tape Library, refer to the *IBM System Storage Tape Libraries Guide for Open Systems*, SG24-5946.

3.4.9 IBM System Storage TS3310 Tape Library

The IBM TS3310 Tape Library is a modular, highly scalable IBM LTO library. It is designed to address the tape requirements of companies with rapid data growth. The TS3310 expands vertically when you add expansion modules.

The IBM TS3310 Tape Library offers a broad range of configuration options. The smallest configuration includes a base unit model L5B with one or two tape drives, either IBM LTO3, LTO4, or a mix of LTO3 and LTO4, 30 storage slots, and six I/O slots. The base library module contains all of the necessary robotics and intelligence to manage the library system. You can order the base models either as a deskside unit or for installation in an industry standard 19-inch rack, where it requires four rack units.

As your need for tape backup expands, you can add up to four expansion modules TS3310 Model E9U to the base configuration. Each of these modules is nine rack-units high and adds space for cartridges and tape drives to your TS3310 configuration. Configurations with more than one expansion module require a rack for installation.

A fully configured rack-mounted library is 41 rack-units high and offers space for up to 18 LTO drives and 396 cartridges. You can configure up to 54 I/O slots in this configuration. Using LTO4 drives and LTO4 media results in an uncompressed capacity of 316.8 TB (633.6 TB with 2:1 compression).

The IBM TS3310 Tape Library provides the ability to configure the number of logical libraries up to the number of tape drives, which provides a maximum capability of 18 logical libraries for the IBM TS3310.

Available as a standard feature, a Remote Management Unit (RMU) provides an Ethernet port so that the library can be configured as a TCP/IP device in the network. Library status can be sent to the network as Simple Network Management Protocol (SNMP) traps. The TS3310 also supports an embedded SMI-S agent for monitoring the library using tools like TotalStorage Productivity Center. The IBM System Storage Tape Library Specialist enables network access (with a Web browser) to the library for more detailed status and for updating the firmware of the library. All library Operator Panel functions can be accessed using the IBM System Storage Tape Library Specialist.

The TS3310 tape library attaches to IBM System p, IBM System i, IBM System x, Microsoft Windows, HP-UX, Sun Solaris, UNIX, and Linux servers.

The IBM TS3310 supports Application-Managed Encryption (AME), System-Managed Encryption (SME) and Library-Managed Encryption (LME) on SAS and Fibre Channel LTO4 drives using LTO4 media.

Three policy settings are available for Library-Managed Encryption on a TS3310:

- Encrypt All

  All cartridges in a non-partitioned TS3310 library will be encrypted. If you have partitioned a TS3310 library into two or more logical libraries, you can set the encryption policy separately for each logical library.

- Internal Label - Selective Encryption

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

- Internal Label - Encrypt All

  The LTO4 drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

You set these policies through the Web interface.

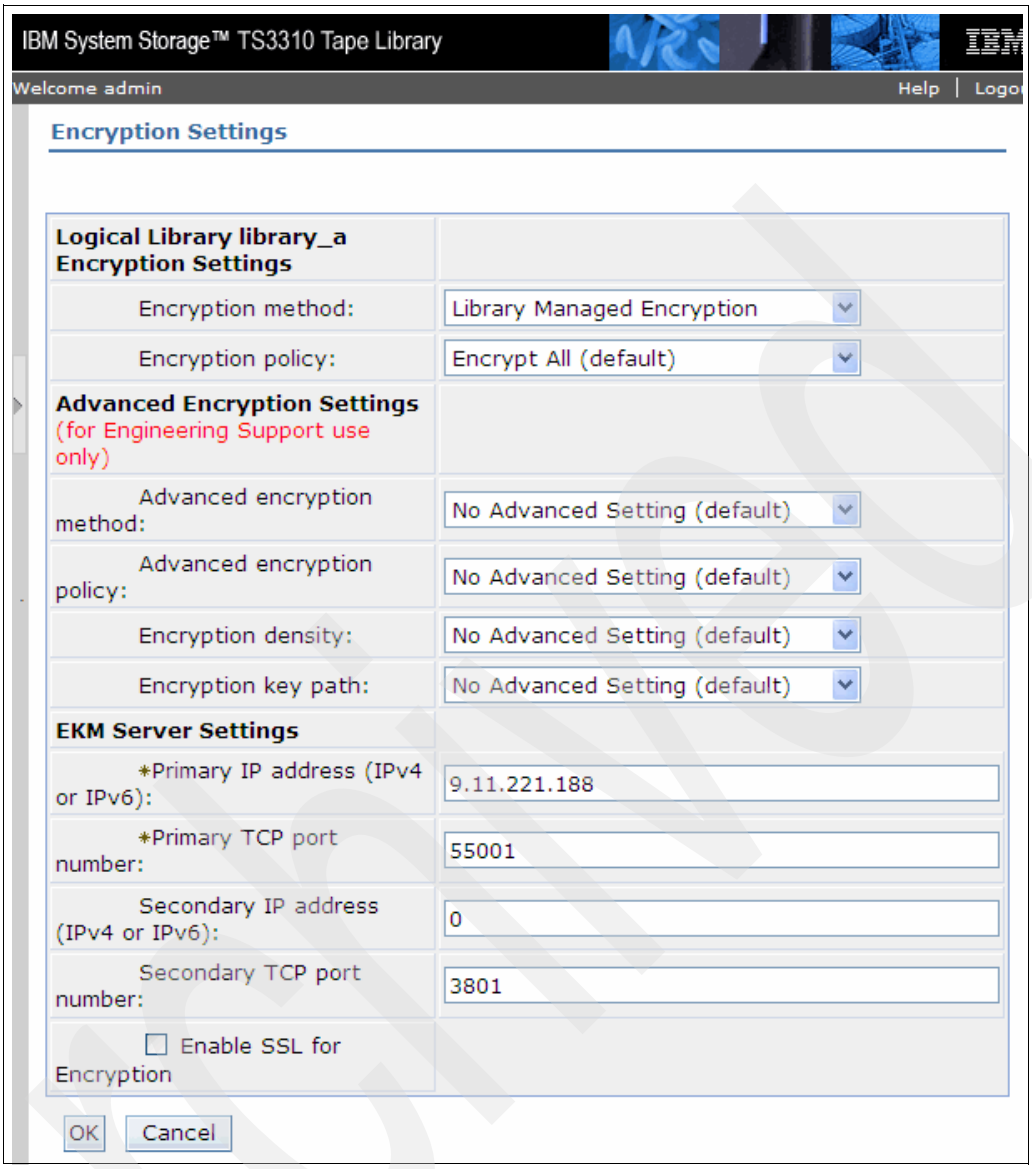


Figure 3-22 TS3310 Encryption configuration interface

Not all environments support all of these three encryption methods. For details on platform-specific support, refer to Chapter 4, “Planning for software and hardware” on page 91.

For more information about IBM TS3310 Tape Library, refer to *IBM System Storage Tape Libraries Guide for Open Systems*, SG24-5946.

Figure 3-23 on page 75 shows a TS3310 configuration with the base unit model L5B and one expansion unit model E9U.

Figure 3-23 IBM System Storage TS3310 Tape Library

3.5 IBM System Storage TS3400 Tape Library

The IBM System Storage TS3400 Tape Library is designed to offer high performance drive technology and automation in Open Systems and System z environments. The TS3400 is a five unit external desktop or rack-mountable tape library that supports up to two IBM System Storage TS1120 Tape Drives or IBM System Storage TS1130 Tape Drives. The library does not support the predecessor of the TS1120, the IBM 3592 Model J1A Tape Drive.

The TS3400 is an excellent tape storage solution for organizations already using TS1130 or TS1120 Tape Drives in their data centers that want to use the same technology in remote locations. The TS3400 is also designed for organizations that have limited physical space in their IT environments. Installed in a standard 19-inch rack, it provides up to 18 TB of uncompressed tape storage in a 5U space. Figure 3-24 shows the IBM TS3400 Tape Library.

Figure 3-24 IBM System Storage TS3400 Tape Library

Characteristics

The TS3400 supports one or two IBM TS1130 or IBM TS1120 Tape Drives. It has two removable cartridge magazines providing 18 data cartridge slots, including a three slot I/O Station. You can configure the slots of the I/O Station either as normal cartridge slots or as I/O slots. The total native storage capacity is 18 TB when using the 1000 GB data cartridges and all 18 slots are configured as slots for data cartridges.

IBM multipath architecture allows for partitioning of the TS3400 when two IBM System Storage TS1120 or TS1130 Tape drives are installed. You can partition the library into two partitions to share it between different applications.

The TS3400 attaches through the Fibre Channel ports of the IBM TS1120 or TS1130 Tape Drives to Open Systems hosts or to an ESCON/FICON tape controller in a System z environment. Each TS1120 or TS1130 has two independent 4 Gbps switched fabric Fibre Channel ports.

Remote management through a Web browser allows you to communicate directly with the library and perform a wide range of user, operator, and administrator tasks without being at the operator panel.

System z attachment requires an IBM TS1120 Tape Controller (refer to 3.2, “IBM System Storage TS1120 Tape Controller” on page 53). Up to seven TS3400 tape libraries with a maximum of 14 TS1120 or TS1130 drives can attach to a single TS1120 Model C06. Up to four IBM TS1120 or TS1130 Tape Drives can be direct-attached to the IBM TS1120 Tape Controller without the use of an external switch.

When the TS3400 attaches to an IBM TS1120 Tape Controller, you can choose between System Mode and Auto Mode. In System Mode, cartridges are fed to the drive one after the other under the attaching system’s command, which continues until all storage cells are processed. Currently, only z/OS supports System Mode. In Auto Mode, the TS3400 acts like an autoloader, and cartridges are automatically fed into the drive one after another until all storage cells are processed.

Both the TS3400 libraries and the IBM TS1120 Tape Controller must be rack-installed. TS1120 or TS1130Tape Drives attached to an IBM TS1120 Tape Controller cannot be shared with Open Systems hosts. If a TS3400 attaches to an IBM TS1120 Tape Controller, all drives attached to this controller must reside in a TS3400 library. You cannot share an IBM TS1120 Tape Controller between drives in a TS3400 and rack-mounted drives or drives in a TS3500 or 3494 tape library.

Standard features and capabilities of the IBM System Storage TS3400 Tape Library are summarized in the following list:

- Control path and data path failover
- Two removable cartridge magazines with nine slots each
- Configurable three slot I/O Station
- Barcode reader
- Dual power supplies
- Remote management unit
- Open Systems and System z attachment
- Sequential or random access mode selectable for Open Systems-attached library
- System Mode or Auto Mode selectable for System z-attached library
- Support for tape encryption

Supported platforms

Support for the TS3400 library is provided on z/OS 1.6 or later, z/VM V5.2.0 or later, z/VSE V3.1.2 or later, and z/TPF V1.1 or later. The z/VM, z/VSE, and z/TPF support the TS3400 as an autoloader in Auto Mode. You might require later operating system versions to support tape encryption. Refer to Chapter 4, “Planning for software and hardware” on page 91.

A wide variety of Open Systems platforms supports the IBM TS3400 Tape Library. For additional information, refer to the System Storage Interoperation Center (SSIC):

`http://www.ibm.com/systems/support/storage/config/ssic/`

Encryption support

The IBM System Storage TS3400 Tape Library supports the IBM System Storage TS1120 or TS1130 Tape Drive built-in encryption capabilities. The TS3400 Tape Library supports Library-Managed Encryption (LME), System-Managed Encryption (SME), and Application-Managed Encryption (AME).

Three policy settings are available for Library-Managed Encryption on a TS3400:

- Encrypt All

  All cartridges in a non-partitioned TS3400 library will be encrypted. If you have partitioned a TS3400 library into two logical libraries, you can set the policy separately for each logical library.

- Internal Label -Selective Encryption

  The IBM TS1120 or TS1130 Tape Drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

- Internal Label - Encrypt All

  The IBM TS1120 or TS1130 Tape Drive automatically derives the encryption policy and key information from the metadata written on the tape volume by the application. NetBackup is currently the only backup software that supports Internal Label Encryption Policies (ILEP).

In a System z environment, SME is the only option. You set the policies through the Web interface.

3.6 IBM System Storage TS3500 Tape Library

The IBM TS3500 Tape Library (machine type 3584) is designed for medium to large automated tape storage and backup solutions and is part of a whole family of tape libraries for small to large automated tape storage and backup solutions. Originally delivered in 2000 at the same time as Linear Tape Open (LTO) Ultrium technology, the TS3500 offers a robust enterprise library solution available for midrange and high-end Open Systems. Since its introduction, the library has been enhanced to accommodate different drive types and operating platforms, more recently including the attachment of System z (mainframe) hosts and tape controllers. Combining reliable, automated tape handling and storage with reliable, high-performance IBM TS1130, TS1120 drives and LTO tape, the IBM TS3500 Tape Library offers outstanding retrieval performance with typical cartridge move times of less than three seconds.

The IBM TS3500 Tape Library can be partitioned into multiple logical libraries, making it an excellent choice for consolidating tape workloads from multiple heterogeneous Open Systems servers, and enables the support for System z attachment in the same library.

In addition, the IBM TS3500 Tape Library provides outstanding reliability and redundancy through the provision of redundant power supplies in each frame, an optional second cartridge accessor, control and data path failover, and dual grippers within each cartridge accessor. Both library and drive firmware can now be upgraded nondisruptively, that is, without interrupting the normal operations of the library.

Figure 3-25 on page 78 show the maximum configuration of 16 frames and the minimum configuration of one frame of an IBM TS3500 Tape Library, which can contain from one to 192

TS1130, TS1120, or LTO Tape Drives; and from 58 to 6,260 (frames for TS1120, or TS1130 only) or 6,887 (frames for LTO only) cartridge storage cells.

Figure 3-25 Maximum and minimum IBM TS3500 Tape Library configuration

The IBM TS3500 Tape Library provides:

- Modular, scalable, automated tape library, combining IBM tape and automation for Open Systems and mainframe hosts, using a variety of IBM drive types
- Attachment to IBM System z, System i, iSeries, AS/400, System p, pSeries, RS/6000, IBM System x, Netfinity, Sun, Hewlett-Packard, and other IBM servers (that are not IBM)
- Connectivity using FICON, ESCON, Fibre Channel, Low Voltage Differential (LVD) SCSI, and High Voltage Differential (HVD) SCSI
- IBM Multipath Architecture designed to support redundant control paths, mixed drive configurations, and library sharing between multiple applications
- Tape data encryption

3.6.1 TS3500 frames

Seven different frames are currently available to build an IBM TS3500 Tape Library. Each frame is identified by a three character model number (L23, L53, D23, D53, S24™, S54, or HA1) that describes the nature of the frame. Libraries are built of modules, as follows:

- Each library requires a base frame (model L*xx*) to which optional expansion frames (model D*xx or Sxx*) can be added. Only one base frame is permitted in each library configuration.
- Base and expansion Dxx frames support one of either:
  - LTO drives (model *x*53)
  - 3592 drives, 3592-J1A, TS1120 and TS1130 (model *x*23)
- Storage frames (model Sxx) do not support drives but do support higher cartridge density.
- Optional second accessor is made available through the addition of model HA1 frames.

All currently available frame models can be intermixed with each other and installed frame models with the provision that there is only one base frame in each library. Installed frame models include the L22, L32, L52, D22, D32, and D52.

A mix of 3592 and LTO drives within one library is supported, because frames for 3592 and LTO drives can be mixed. TS3500 frames are dedicated to either 3592 or to LTO. Therefore, you cannot mix these drive types in a single frame.

The following sections introduce the available frame models.

IBM System Storage TS3500 Model L23 frame for TS1120 or TS1130

The L23 frame can be installed on its own as a complete library enclosure or up to 15 model D23s or D53s can be attached to it. The L23 frame provides the major library components for the whole library, whether the library consists of a single frame or multiple frames. Expansion frames must be added to the right of the L23 frame. In addition, model S24 or S54 storage only frames may be attached.

The L23 frame provides cartridge slots for 3592 media and support for up to twelve IBM TS1130, TS1120, or 3592-J1A (collectively referred to as 3592) Tape Drives. The number of available 3592 cartridge storage slots ranges from 58 to 260. The minimum configuration provides 58 slots available for actual use, although all 260 slots are already physically installed. You can enable additional slots for use (up to the total of 260) by ordering Capacity On Demand features. The Intermediate Capacity feature gives you a total of 117 usable cartridge slots. This Intermediate Capacity feature is required to add a Full Capacity feature, which gives you the capacity of 260 cartridge slots. The Full Capacity feature is required to add an I/O slot feature or to attach the optional expansion frame models D23 or D53.

You can install a maximum of twelve 3592 drives in an L23 frame. If you install additional drives in an L23 frame, the fifth and the ninth drive reduce the number of available cartridge slots.

Note: ALMS is Required when installing the TS1130 Tape Drive in a TS3500 tape library.

Each L23 has a standard 16-slot 3592 cartridge I/O Station for convenient importing cartridges into the library or exporting cartridges from the library. Optionally, you can order 16 additional I/O slots for 3592 or LTO media if the library contains frames with LTO drives. Additional I/O slots reduce the number of available cartridge slots.

IBM System Storage TS3500 Model D23 frame for TS1130

The D23 frame is an expansion frame and cannot be installed on its own. It must be connected to a L23 or L53 frame and optionally to other expansion frames.

The D23 frame offers space for a maximum of 400 3592 media. Up to 12 TS1130, TS1120 or 3592-J1A (collectively referred to as 3592) drives can be installed in this frame. If one or more tape drives are installed in the D23, the Enhanced Frame Control Assembly feature is also required. This feature provides the hardware and firmware required to support IBM 3592 drives within the D23 and provides a redundant line feed for the L23 or L53 accessor.

The installation of the first, fifth, and ninth 3592 drive reduces the number of available cartridge slots.

Note: ALMS is Required when installing the TS1130 Tape Drive in a TS3500 tape library.

In a library with 3592 drives only, you can order an additional 64 slot I/O Station for 3592 media for the D23 frame. The additional I/O Station reduces the number of available cartridge slots.

IBM System Storage TS3500 Model S24 frame for TS1130

The S24 frame is an expansion frame and cannot be installed on its own. It must be connected to a L23 or L53 frame and optionally to other expansion frames.

The S24 frame offers space for a maximum of 1000 3592 media. By default it comes with space for 600 3592 cartridges. The ALMS feature is required to support the S24 frame. The High Density Capacity On Demand feature is required to unlock access to the remaining 400 slots.

IBM System Storage TS3500 Model L53 frame for LTO

The L53 frame provides cartridge slots for LTO media and supports up to 12 LTO4 or LTO3 4 Gbps Fibre Channel tape drives.

The number of available LTO cartridge storage slots ranges from 64 to 287. The minimum configuration provides 64 slots available for actual use, although all 287 slots are already physically installed. You can enable additional slots for use (up to the total of 287) by ordering Capacity on Demand features. The Intermediate Capacity feature gives you a total of 129 usable cartridge slots. This Intermediate Capacity feature is required in order to add a Full Capacity feature, which gives you the capacity of 287 cartridge slots. The Full Capacity feature is required to add an I/O Slot feature or to attach the optional expansion frame models D23 or D53. In addition, models S24 or S54 storage-only frames may be attached if the ALMS feature is installed.

You can install a maximum of 12 LTO drives in an L53 frame. If you install additional drives in an L53 frame, the fifth and the ninth drive reduce the number of available cartridge slots.

Each L23 has a standard 16-slot 3592 cartridge I/O Station for conveniently importing cartridges into the library or exporting cartridges from the library. Optionally, you can order 16 additional I/O slots for LTO media or 3592 media if the library contains frames with 3592, TS1120 or TS1130 drives. Additional I/O slots reduce the number of available cartridge slots.

IBM System Storage TS3500 Model D53 frame for LTO

The D53 frame is an expansion frame and cannot be installed on its own. It must be connected to a L23 or L53 frame and optionally to other expansion frames.

The D53 frame offers space for a maximum of 440 LTO media. Up to 12 TS1030 or TS1040 LTO 4 Gbps Fibre Channel tape drives can be installed in this frame. If one or more tape drives are installed in the D53, the Enhanced Frame Control Assembly feature is also required. This feature provides the hardware and firmware required to support IBM LTO drives within the D53 and provides a redundant line feed for the L23 or L53 accessor.

The installation of the first, fifth, and ninth LTO drive reduces the number of available cartridge slots.

In a library with LTO drives only, you may order an additional 64 slot I/O Station for LTO media for the D23 frame. The additional I/O Station reduces the number of available cartridge slots.

IBM System Storage TS3500 Model S54 frame for LTO

The S54 frame is an expansion frame and cannot be installed on its own. It must be connected to a L23 or L53 frame and optionally to other expansion frames.

The S54 frame offers space for a maximum of 1320 LTO media. By default it comes with space for 660 LTO cartridges.The ALMS feature is required to support the S54 frame. The High Density Capacity on Demand feature is required to unlock access to the remaining 660 slots.

IBM System Storage Model HA1 Frame

The TS3500 HA1 frame adds a second accessor to a TS3500 tape library in a high availability configuration. The HA1 frame is installed left of the TS3500 Model L*xx* frame and acts as a service bay for the left accessor. A high availability configuration with an HA1 frame requires a driveless TS3500 Model D*xx* expansion frame as the rightmost frame in this configuration. This expansion frame acts as the service bay for the right accessor.

3.6.2 TS3500 characteristics

In the following sections, we describe:

- Advanced Library Management System (ALMS)
- Logical library partitions
- System z attachment through 3953 Library Manager
- Path failover
- TS3500 Tape Library Specialist
- High Density Frames
- Encryption support

Advanced Library Management System

The Advanced Library Management System (ALMS), which is an optional extension in existing libraries and required for new installs, to the IBM patented Multipath Architecture (FC1690), provides enhanced flexibility and capabilities for partitioning the IBM TS3500 Tape Library. The Advanced Library Management System (ALMS) virtualizes the SCSI element addresses while maintaining the approach of the multipath architecture and using SCSI Medium Changer commands. Without ALMS, everything is based on the SCSI element address (location-centric) and partitioning is based on real cartridge slots and drive slots. With ALMS, there is no affinity between a real slot address and a SCSI element address reported to the server and used by the server. Instead, there is now an affinity with the VOLSER (volume serial numbers on the barcode label of the cartridge).

With ALMS, the TS3500 virtualizes the location of cartridges (called *SCSI element addresses*). Without ALMS, the Storage Element Address maps directly to a specific storage slot after the library is configured. With ALMS enabled, each Storage Element Address is no longer associated with a specific storage slot. Instead, storage slots are virtualized by dynamically associating element addresses to them as required. An element address is associated to a storage slot selected by the library as cartridges are moved and inventoried. If a Storage Element is empty because of a move, that source element address becomes unassociated.

Capabilities of ALMS include:

- Dynamic partitioning (storage slot pooling and flexible drive assignment)
- Flexible drive assignment
- Sharing of the physical cartridge slots by logical partitions
- Ability to add or remove storage capacity with minimal or no disruption to host applications
- The ability to configure drive or L frame storage capacity without taking the library offline
- Virtual I/O slots to automatically manage the movement of cartridges between I/O slots and storage slots
- Cartridge Assignment Policy (CAP)
- Tape System Reporter
- Native SMI-S Support

The IBM TS3500 Tape Library is compliant with the SCSI Medium Changer standard whether ALMS is enabled or not; when enabled, ALMS is completely transparent to the application. The SCSI Medium Changer can be thought of as a l*ocation-centric* interface. The application controlling a SCSI Medium Changer device specifies a source and a destination location for each request to move a cartridge. The traditional SCSI library does not have control of the cartridge locations; instead, the SCSI library just acts on behalf of the server.

IBM Tape System Reporter is a windows application that monitors multiple TS3500 libraries and allows report generation. It also tracks history for cartridges and drives in the library allowing you to monitor tape and drive encryption over time. For more information refer to *IBM System Storage TS3500 Tape Library with ALMS Tape System Reporter User's Guide (GA32-0589)*

ALMS is an optional feature for existing IBM TS3500 Tape Libraries; however, it is a mandatory feature for new installs, when the library when you attach the library to a System z server or when you install the IBM 3584 Model HA1.

Note: Though ALMS is not an absolute requirement for encryption, we strongly recommend that you order ALMS.

Logical library partitions in a TS3500

The Multipath Architecture of the IBM TS3500 Tape Library provides the capability for sharing library robotics by partitioning the library into logical partitions. A single IBM TS3500 Tape Library can have Open Systems-attached partitions and System z-attached partitions, but logical partitioning differs between Open Systems environments and System z environments.

Logical library partitions in an Open Systems environment

In an Open Systems environment, you can partition the library into as many logical partitions as there are drives installed, that is, up to a maximum of 192 logical libraries. Each logical library has its own separate and distinct drives, storage slots, and control paths. I/O slots are shared on a first-come-first-served basis. The ALMS Virtual I/O function can be used to manage sharing the I/O station. All logical libraries share the robotics of the TS3500 library. The virtualization of the library accessor allows homogeneous and heterogeneous servers and applications to share the library robotics independently of each other. Drives can be shared between logical library partitions in an Open Systems environment. Cartridges are not shared among logical libraries.

Logical library partitions in a System z environment

For System z attachment, the configuration rules for logical partitioning are different from those for Open Systems hosts. System z-attached TS1130, TS1120 or 3592-J1A drives in a TS3500 require an external 3953 Library Manager. This 3953 Library Manager controls a separate logical library partition (with its own separate and distinct drives, storage slots, and control paths) in the IBM TS3500 Tape Library. Up to four 3953 Library Managers can attach to a TS3500, but each Library Manager requires a separate logical library partition of its own.

Note: When using TS7740 licensed internal code 8.5.0.xx and later a separate 3593 Library Manager is no longer required to attach to a TS3500 to a TS7740. This function is integrated into the TS7740.

Each one of these 3953 logical Library Manager partitions can have up to three logical libraries defined to it (identical to logical libraries of a 3494), two of which can be TS7700 Virtualization Engines or VTSs. The 3953 Library Manager supports the configuration of 16 subsystems per 3953 tape system. A subsystem is a TS7700 Virtualization Engine, a Virtual

Tape Server (VTS), or a TS1120 Model C06 or 3592 Model J70 controller. If two virtual systems are defined, the remaining 14 subsystems can be 14 TS1120-C06 or 3592-J70 tape controllers dedicated to native drive attachment (native library).

This configuration of logical Library Manager partitions and further subsystem attachment allows a TS3500 Tape Library that is fully configured with four 3953 tape systems to house up to eight TS7700 Virtualization Engines. With the latest version of the TS7700 a separate 3953 is no longer required as the library manger function is now part of the TS7700.

Figure 3-26 illustrates the attachment of a TS3500 library to System z hosts. The illustration shows all of the drives belonging to a logical library in contiguous positions in the physical library. This configuration is not a requirement. You can distribute the drives of a logical library across several frames. In fact, drives belonging to a Library Manager-attached logical library can share the same frame with drives belonging to an Open Systems partition.

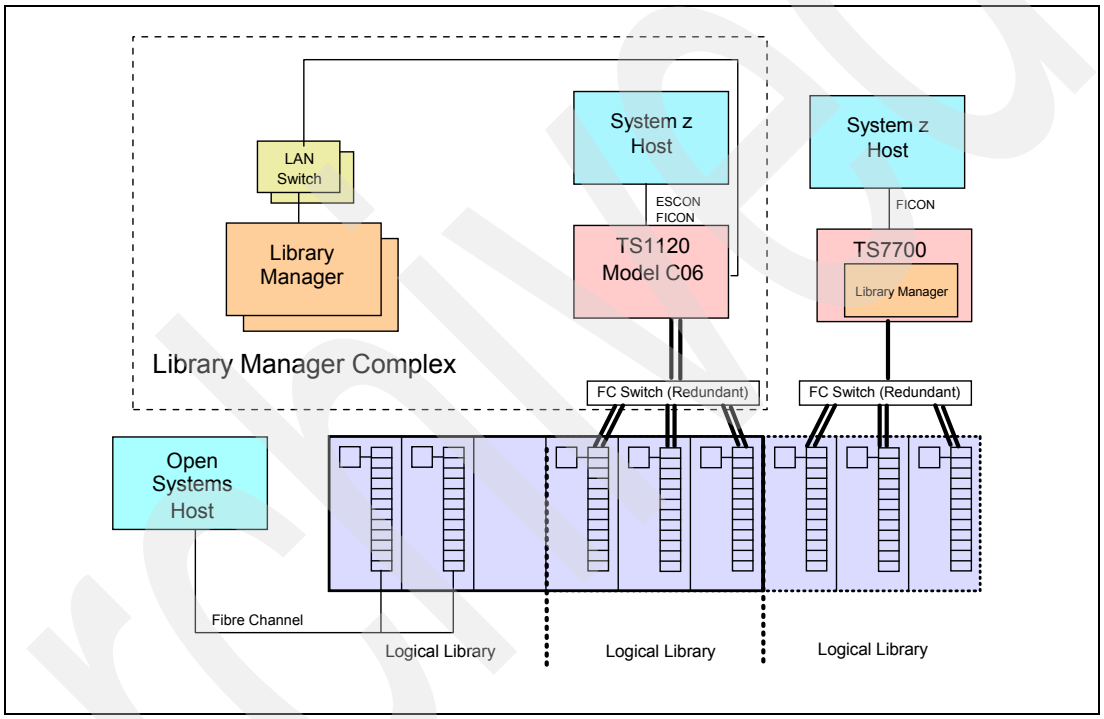


Figure 3-26 TS3500 attached to System z

System z attachment with IBM 3953 Tape System

To attach System z to the IBM System Storage TS3500 Tape Library, a number of specific hardware elements are required. Here is a summary of equipment that is either required or installed according to client system requirements:

- IBM TotalStorage 3953 Tape Frame Model F05

  This is a special frame, which is independent of the TS3500 library, that is used to house the Library Manager, switches, and controllers for mainframe-attached tape. It has:

  - TS3000 System Console
  - Keyboard-Video-Mouse (KVM) switch
  - Ethernet switches for the internal communications between the Library Manager, tape controllers, IBM TS7700, and the TSSC
- IBM TotalStorage 3953 Library Manager Model L05
- IBM TS3500 Models L23 and D23 frames

- IBM TS1130, TS1120 or 3592-J1A Tape Drives

  Note: IBM 3592-J1A Tape Drives are not encryption-capable; you must have TS1130 or TS1120 Tape Drives to use tape encryption.

- IBM TS7700 Virtualization Engine or 3494 Virtual Tape Server models B10 or B20

  Note: When using TS7740 licensed internal code 8.5.0.xx and later a separate 3593 Library Manager is no longer required to attach to a TS3500 to a TS7740. This function is integrated into the TS7740

- IBM System Storage TS1120 Model C06 or 3592 Model J70 tape controller

  Fibre Channel switches to attach the IBM 3592 Tape Drives to the controllers

For detailed information about the IBM 3953 Tape System, refer to the *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789.

Control path failover and data path failover

The TS3500 supports control path failover (CPF) and data path failover (DPF). The optional path failover feature includes both CPF and DPF.

Control path failover

In the TS3500 Tape Library, a *control path* is a logical path into the library, which uses the physical channel path to the tape drives, through which a server sends SCSI move medium commands to control the logical library. The control path is set up as Logical Unit Number (LUN) 1 on a drive; LUN 0 addresses the server-attached drive.

Alternate path support, which is currently available for AIX, Linux, Solaris, HP-UX, and Windows hosts, configures multiple physical control paths to the same logical library within the device driver and provides automatic failover to an alternate control path when a permanent error occurs on one path. This is transparent to the running application.

Note: Best practice is to have control path drives in different library frames for increased redundancy. See Figure 3-27 on page 85.

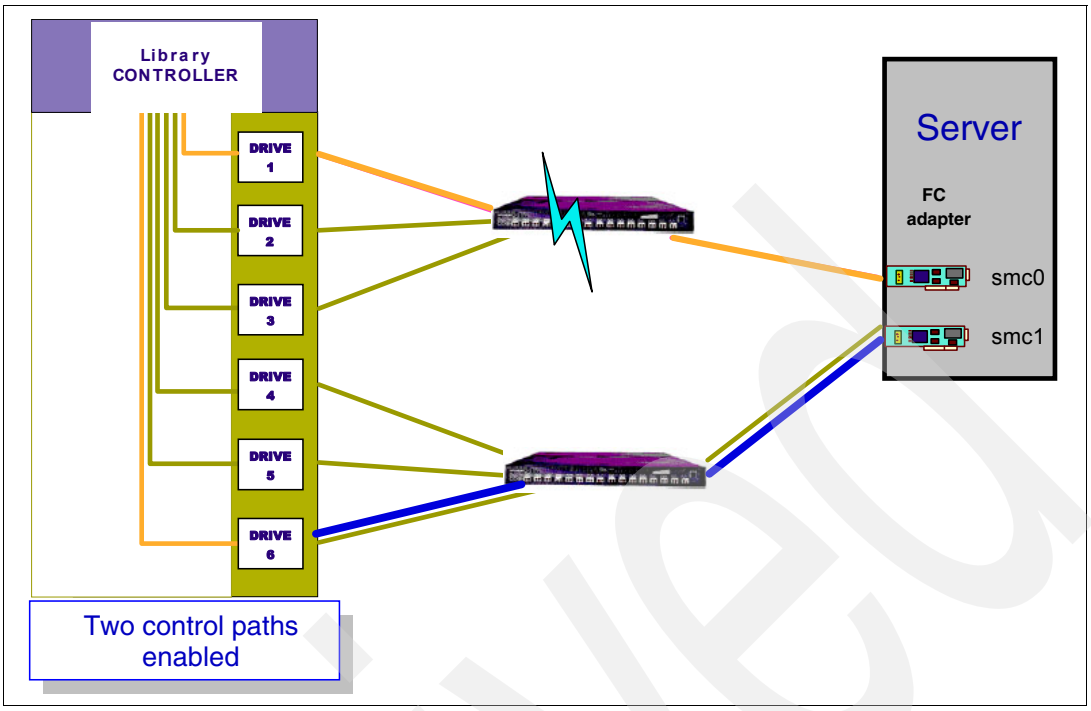


Figure 3-27 Redundant control paths to the library controller

Data path failover

Data path failover and load balancing exclusively support native Fibre Channel Ultrium and IBM TS1130, TS1120 or 3592-J1A Tape Drives in the IBM TS3500 Tape Library using the IBM device driver. Data path failover is supported for AIX, Linux, HP, Solaris, and Windows hosts. Load balancing is supported for AIX, HP-UX, Linux, Solaris, and Windows.

Refer to the *IBM Tape Device Driver Installation and User's Guide*, GC27-2130, for current support and implementation details.

Data path failover provides a failover mechanism in the IBM device driver so that you can configure multiple redundant paths in a SAN environment. If a path or component fails, the failover mechanism will automatically retry the current operation using an alternate, preconfigured path without stopping the current job in progress. When accessing a tape drive device that has been configured with alternate pathing across multiple host ports, the IBM device driver automatically selects a path through the HBA that has the fewest open tape devices and assigns that path to the application. This autonomic self-optimizing capability is called *load balancing*.

Data path failover and load balancing support for AIX or for IBM 3592 Tape Drives do not require the Path Failover feature.

IBM TS3500 Tape Library Specialist

The IBM TS3500 Tape Library's Web interface, which is also known as the IBM TotalStorage UltraScalable Tape Library Specialist, enables operators and administrators to manage storage devices from any location in an enterprise. The IBM TS3500 Tape Library Specialist enables direct communication with an IBM TS3500 Tape Library and provides a full range of user, operator, and administrator tasks, which can be executed remotely. For programmatic remote access to the tape library the TS3500 includes Storage Management Interface-Specification (SMI-S) support. Monitor or superuser role is required to access the SMI-S interface.

Firmware for the library and drives can be updated nondisruptively using the Web user interface.

Individual Web login IDs and passwords

For the IBM TS3500 Tape Library, the Web user interface (UI) supports a list of users who can access various areas of the Web user interface. An administrator can create up to nineteen additional user IDs. Each user has a 30-character name, a 15-character login ID, a 15-character password, and an access level. The access level defines the level of Web access that the user is allowed.

Four access levels are available:

Monitor	Can view all physical and logical library data.
Service	Can perform only service-related functions, such as update firmware, download logs, and view vital product data (VPD).
Superuser	Can perform all tasks of a monitor or service role, plus change library settings and perform library operations. This role cannot change the passwords of other users or enable or disable security.
Administrator	Can perform all user management tasks.

High-density frames

The IBM TS3500 now supports high density frames this allows more cartridges to be stored inside the library in the same physical footprint in the data center. The ALMS feature is required to add HD frames to your TS3500

HD slots contain tape cartridges in a tiered architecture. The cartridge immediately accessible in the HD slot is a Tier 1 cartridge. Behind that is Tier 2 and so on. The maximum tier in an LTO Ultirum (Model S54) HD slot is Tier 5. The maximum tier in a 3592 (Model S24) HD slot is Tier 4 because the 3592 tape cartridge is slightly longer than the LTO cartridge. The single-deep slots on the door-side of HD frames are referred to as Tier 0 slots. Figure 3-28 shows a top-down view of one row of an HD S54 frame with cartridges in Tiers 0 (door side), 1, 2, and 3.

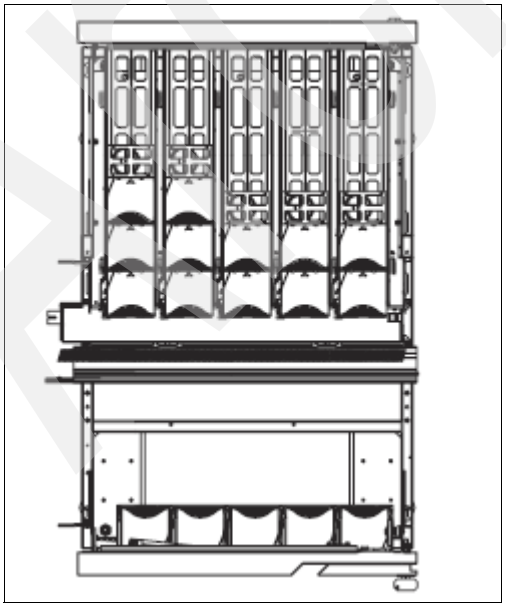


Figure 3-28 Top down view of an HD Model S54 frame with Tier 0 at the bottom

As you can see, access to higher tiers is slower because of having to move cartridges out of the way. Depending on your usage, this could be significant.

Host platforms and device drivers

The IBM TS3500 Tape Library is supported on a variety of operating systems. For a current list of host software versions and release levels that support the TS3500 Tape Library, see:

http://www.ibm.com/servers/storage/tape/compatibility/pdf/ts3500_interop.pdf

For information about TS3500 Tape Library attachment to a System z environment, refer to the *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789.

Encryption support

The TS3500 Tape Library supports Library-Managed Encryption (LME), System-Managed Encryption (SME), and Application-Managed Encryption (AME).

LME on a TS3500 includes support for Barcode Encryption Policy (BEP) and Internal Label Encryption Policies (ILEP).

When using BEP, policies are based on ranges of cartridge volume serial numbers. You can specify the volume serial ranges for the volumes that are to be encrypted. Alternatively, you can exclude ranges of volume serial numbers from encryption. Library-Managed Encryption also allows for encryption of all volumes in a library, independent of barcodes.

Note: New with the November 7, 2008 Firmware the TS3500 may be configured with independent Key Managers for each logical library. This firmware requires the ALMS feature be installed and enabled.

For certain backup applications, such as Symantec NetBackup, encryption policies based on volume serials are not appropriate. When ILEP is configured, the TS1120, TS1130 or LTO Ultrium 4 Tape Drive automatically derives the encryption policy and key information from the metadata that is written on the tape volume by the application. NetBackup is currently the only backup software that supports ILEP. When you select ILEP, you may choose one of the following polices, which you set through the Tape Library Specialist Web interface:

- Internal Label - Selective Encryption
- Internal Label - Encrypt All

In a System z environment, only SME is supported.

Note: SME and LME require an Encryption Key Manager (EKM) or Tivoli Lifecycle Key Manager (TKLM). For LME, you enter the IP addresses of the primary and optionally backup EKMs or TKLMs on the TS3500 library through the TS3500 Specialist. These settings may refer to the physical library or be configured on a per logical library basis. Each logical library may be configured with its own EKM, TKLM, TKLMs or EKMs.

For more information, refer to the *Operator's Guide* for your tape library.

3.7 IBM TotalStorage 3494 Tape Library

An already-installed IBM 3494 Tape Library can consist of up to 16 tape library frames and two additional high-availability tape frames (3494-HA1). IBM 3490E, IBM 3590, IBM 3592-J1A Tape Drives, TS1120, TS1130 Tape Drives can coexist within a single 3494 library. For a detailed description of the IBM 3494 Tape Library, its components and features, refer to the *IBM 3494 Tape Library: A Practical Guide to Enterprise-Class Tape Drives and Tape Automation*, SG24-4632.

The tape library frame models of the IBM 3494 Tape Library have been withdrawn from Marketing. You may still order 3494 Model D22 and D24 drive frames to expand an existing 3494 library. The replacement for the IBM 3494 Tape Library is the IBM System Storage TS3500 Tape Library together with the IBM 3953 Tape System.

Note: Effective December 2006, IBM withdrew from marketing all Model L*xx* Control Unit Frames of the IBM 3494 Tape Library. You may still add selected drive unit frames to already-installed IBM 3494 Tape Libraries, but you cannot install new libraries. The TS3500 Tape Library with the IBM 3953 Tape System replaces the IBM 3494.

Figure 3-29 shows a 3494 Tape Library configuration consisting of a library frame, a driveless storage frame, and two drive frames.

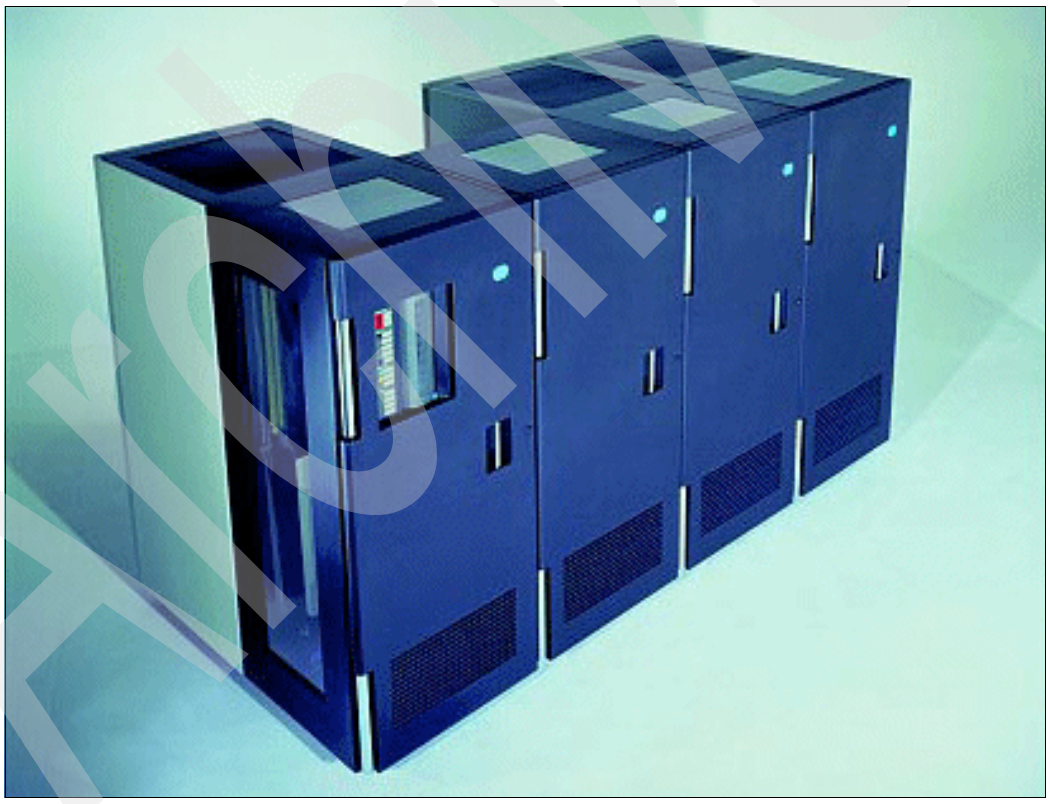


Figure 3-29 IBM TotalStorage 3494 Tape Library

Encryption support

The IBM 3494 Tape Library supports Library-Managed Encryption (LME), System-Managed Encryption (SME), and Application-Managed Encryption (AME).

LME on a 3494 Tape Library includes support for Barcode Encryption Policy (BEP) and Internal Label Encryption Policies (ILEP).

When using BEP, policies are based on ranges of cartridge volume serial numbers. You can specify the volume serial ranges for the volumes that are to be encrypted. Alternatively, you may exclude ranges of volume serial numbers from encryption. LME also allows for encryption of all volumes in a library, independent of barcodes.

For certain backup applications, such as Symantec NetBackup, encryption policies based on volume serials are not appropriate. When ILEP is configured, the IBMTS1130 or IBM TS1120 Tape Drive automatically derives the encryption policy and key information from the metadata that is written on the tape volume by the application. NetBackup is currently the only backup software that supports ILEP. When you select ILEP, you may select one of the following policies, which you set through the Tape Library Specialist Web interface:

- Internal Label - Selective Encryption
- Internal Label - Encrypt All

In a System z environment, only SME is supported.

4

Planning for software and hardware

In the previous chapters, we introduced the basics of encryption, how IBM tape data encryption functions, and how Tivoli Key Lifecycle Manager (TKLM) or Encryption Key Manager (EKM) works. We also covered the tape drives, tape control units, and tape libraries that support encryption.

This chapter describes planning considerations for the tape hardware and the operating systems that you should consider before implementing IBM tape data encryption.

If you plan to use System-Managed Encryption (SME) or Library-Managed Encryption (LME), you should have TKLM or EKM implemented before you can encrypt any tapes. However, many people plan for their tape hardware and their operating system first and then plan TKLM or EKM implementation. These two major parts of your tape data encryption implementation might be handled by different people in your organization.

Because the platform on which the tape drives reside can be different from the platform where TKLM or EKM is implemented, we have separated out EKM planning and placed it in Chapter 5, “Planning for EKM and its keystores” on page 139 Information about TKLM planning has been placed in Chapter 9, “Planning for TKLM and its keystores” on page 317.

4.1 Encryption planning

Encryption is not your typical tape or library upgrade. Significant new function and infrastructure must be implemented with an encryption solution. Planning is vital to a smooth rollout of an encryption solution into your existing environment. Before you tackle this chapter, if this is your first experience with IBM tape drives or libraries, make sure you have read Chapter 3, “IBM System Storage tape and tape automation for encryption” on page 45.

Then, even if you have had experience with IBM Encryption Facility for z/OS or other vendor cryptology products, you should read 1.4, “Concepts of tape data encryption” on page 9 and Chapter 2, “IBM tape encryption methods” on page 23 to understand how the Encryption Key Manager (EKM) or Tivoli Key Lifecycle Manager (TKLM) component ties your operating system platforms together with your keystores.

After reading those basic topics, you are ready to use this chapter to plan the implementation of the IBM TS1120 or TS1130 tape data encryption, or 3592 Encryption solution; and the Linear Tape-Open (LTO) 4 or LTO4 Encryption solution.

4.2 Planning assumptions

Note: We discuss tape data encryption planning assuming that your tape drives and tape libraries have already been installed without encryption.

Most of you are familiar with IBM tape drive or tape library implementations and upgrades of the past. For those of you who are implementing new IBM tape drives or tape libraries, refer to several existing IBM Redbooks publications for planning and implementation details:

- *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789

  This book describes the TS1120 tape drive in a TS3500 tape library using z/OS or other System z platforms, most concepts should map to the TS1130 as well.

- *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946

  This book describes the TS1120, TS1130 and the LTO4 tape drive in Open Systems implementations. This book also discusses Open Systems IBM tape libraries: the TS3500, 3494, TS3400, TS3310, TS3200, TS3100, TS2900.

- *IBM TotalStorage 3494 Tape Library: A Practical Guide to Tape Drives and Tape Automation*, SG24-4632

  This book explains how to plan for and how to install the tape products and library in enterprise platforms. It considers day-to-day operations and integration with other products and applications and also provides information about data migration and operational considerations.

- *IBM System Storage Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312

  This book describes the TS7700 Virtualization Engine using 3592, TS1120, TS1130 tape drives in a TS3500 tape library or a 3494 tape library.

The major part of this chapter focuses on the encryption aspects of this new solution, because the underlying tape and application processing basics, with which you are familiar, do not change.

4.3 Encryption planning quick-reference

The tables in this section compare encryption on the 3592 drive family to encryption on the LTO4 drive:

- Table 4-1 compares encryption characteristics.
- Table 4-2 on page 94 compares drive, library, and controller prerequisites for encryption.
- Table 4-3 on page 95 compares available encryption methods.

On Open Systems platforms, the 3592 and the LTO4 are almost identical in the encryption methods that they support and the operating system software requirements. However, the LTO4, and TS1130 support can require later software levels.

Table 4-1 compares encryption characteristics of the 3592 drive family and IBM LTO4 drive.

Table 4-1 Encryption implementation characteristics comparison

Description	3592 Tape drive family (3592-E05, 3592-EU6, 3592-E06)	LTO4 Tape drive
Encryption standard	AES (256-bit)	AES (256-bit)
Encryption process for data	Symmetric AES (256)	Symmetric AES (256)
Encryption key model	Wrapped key	Direct key
Encryption type for data keys	Public-private key (Asymmetric)	None
Data keys used	Unique data key for each cartridge	Keylist: A list or range of data keys used, pregenerated in keystore
Data keys stored?	Wrapped (that is, encrypted) data keys (2) stored on cartridge, called EEDKs	Stored in keystore
Rewriteable media required	3592 JJ, JA, or JB cartridges	Ultrium 4 media only
WORM media required	3592 JR, JW, or JX cartridges	Ultrium 4 WORM media only
Rekeying support	z/OS, TS3500, and 3494	No

Table 4-2 compares tape drive, library, and controller prerequisites for tape data encryption.

Table 4-2 Drive, library, and controller encryption prerequisites

Description	3592 tape drive family	LTO4 tape drive
Tape drives		
Drive machine type and model	3592-E05 3592-E06 3592-EU6	1. TS1040 (3588-F4A) 2. Feature code (FC) 8144 or FC8145 of the TS3310, TS3200, or TS3100 tape library 3. TS2340 (3580-S43) 4. TS2240 (3580E4S 5. TS2900 (3572-SH4) With Feature Code 5901 (Transparent LTO Encryption)
Encryption-capable	FC9592 or FC5592 for 3592-E05. Standard on 3592-E06 and 3592-EU6	Standard
Encryption-enabled and encryption method set	FC9596 or FC5596 on 3592-E05 drive, FC9595 or FC5595 on controller, or with library menus	Via library menus
Tape controllers (System z only)		
3592-J70 tape controller with drives in a TS3500 or 3494	Yes, FC9595 or FC5595. Also, FC5593 for out-of-band support	N/A
TS1120 Model C06 tape controller (3592-C06) with drives in TS3500 or 3494	Yes, FC9595 or FC5595. Also, FC5593 for out-of-band support	N/A
TS1120 Model C06 tape controller (3592-C06) with drives in TS3400	Yes, FC9595 or FC5595 and feature FC5247	N/A
TS7700 Virtualization Engine (3957-V06)	Yes, FC9900	N/A
Tape libraries		
TS3500 Library (3584-L*xx*): ► Frames for drives ► z/OS, z/VM, z/VSE, and z/TPF attach	Yes, FC9900 TS1130 requires ALMS (Depending on configuration FC 1690, 1692, 1693 or 1694) 3584-L22, L23, D22, and D23 Requires 3953-F05 with 3953-L05 Library Manager	Yes, FC9900 and FC1604 3584-L53, L52, L32, D53, D52, and D32 N/A
3494 library: ► Frames for drives	Yes, FC9900 3494-L22, D22, and D24	No N/A
TS3400 library (3577-L5U)	Yes, FC9900	No

Description	3592 tape drive family	LTO4 tape drive
TS3310 library (3576-E9U and 3576-L5B)	No	Yes, FC9900 and FC5900
TS3200 library (3573-L4U)	No	Yes, FC9900 and FC5900
TS3100 library (3573-L2U)	No	Yes, FC9900 and FC5900
TS2900 autoloader (3572-SH4)	No	Yes, FC5901

Table 4-3 compares the Encryption Methods available for each tape drive environment.

Table 4-3 Encryption methods comparison

Encryption method or platform	3592 tape drives	LTO4 tape drive
Application-Managed Encryption (AME)		
Tivoli Storage Manager	TS1120 Release 5.3.4 TS1130 Release 5.4.3 or 5.5.1	5.3.5.1
System-Managed Encryption (SME)		
IBM z/OS (controller-attached drives)	z/OS V1R7[a] or later	No
IBM z/OS (TS7700 Virtualization Engine)	z/OS V1R7[a] or later	No
IBM z/VM (controller-attached drives)	z/VM V5.1 and V5.2	No
IBM z/VM (TS7700 Virtualization Engine)	z/VM 4.4.0 or later	No
IBM z/VSE (controller-attached drives)	z/VSE V3.1 and V4.1	No
IBM z/VSE (TS7700 Virtualization Engine)	z/VSE 3.1.2 or later	No
IBM z/TPF (controller-attached drives)	z/TPF V1.1	No
IBM z/TPF (TS7700 Virtualization Engine)	TPF 4.1 and z/TPF V1.1	No
IBM AIX	AIX V5.2 or later, Atape device driver for TS1130 use 11.2.9.0 or later	AIX V5.2 or later, Atape 10.4.7.0 device driver
Sun Solaris	IBMTape device driver TS1130 IBMTape.4.1.8.7 or later	IBMTape.4.1.5.0 device driver or later
IBM System i5®	No	No
Linux on System z	Lin_Tape device driver	Lin_Tape device driver
Linux on other servers	Lin_Tape device driver	Lin_Tape device driver
Hewlett-Packard UNIX (HP-UX)	No	No
Windows	IBMTape device driver For TS1130 use 6.1.9.8 or later. At the time of this writing there are no WHQL drivers for the TS1130	IBMTape device driver

Encryption method or platform	3592 tape drives	LTO4 tape drive
Library-Managed Encryption (LME)		
Tape Libraries providing this support	TS3500, 3494, and TS3400	TS3500, TS3310, TS3200, TS3100, TS2900
IBM z/OS, z/VM, z/VSE, z/TPF	No	No
IBM AIX	AIX V5.2 or later	AIX 5L™ V5.1,V5.2, or V5.3
Sun Solaris	Sun Solaris 8, 9, or 10 TS1130 use IBMTape.4.1.8.7 or later	Sun Solaris 8, 9, or 10
IBM System i5	TS1120 i5/OS V5.2 or later TS1130 i5/OS v5.3 or later	i5/OS V5.3 or later
Linux on System z	SLES9, SLES10, RHEL4, RHEL5	SLES9, SLES10, or RHEL4, RHEL5
Linux on other servers	SLES9, SLES10, RHEL4, RHEL10, TS1130 requires lin_tape 1.19.0 10	SLES9, SLES10, RHEL4, RHEL5
HP-UX	64-bit HP-UX 11.0, 11i v1 v2, v3, or later atdd.84 or later	64-bit HP-UX 11i v1, 11i v2, 11i.v3 or later atdd.84 or later
Windows	Windows Server 2000, Windows 2003 Server, Windows 2008 Server For TS1130 use 6.1.9.8 or later. At the time of this writing there are no WHQL drivers for the TS1130	Windows Server 2003 (build 3790 or later), Windows 2008 Server use 6.1.9.5 or later

a. Certain earlier releases that are no longer in service also provide support.

4.4 Choosing encryption methods

When starting to plan encryption management, there are several important considerations to determine what solutions are available and what will fit in your environment. The important steps for your planning considerations are to identify the available tape data encryption solutions for your environment, as follows:

1. Identify which type of server is writing and reading the tape data.

   Are all of the servers of one type or one operating system, or do you have multiple operating systems that will be encrypting tape?

2. Identify encryption methods that are available for your server environments and choose the encryption methods that you will use.

3. Identify which server or servers will host the Encryption Key Manager (EKM) or Tivoli Key Lifecycle Manager (TKLM) component. Identifying the server or servers is really a separate decision from where the actual tape encryption takes place, although most

people will probably implement the EKM or TKLM on the same platform where the tape drives reside. Other important considerations are keystore and security requirements.

- We discuss EKM and keystore planning in Chapter 5, “Planning for EKM and its keystores” on page 139.
- We discuss TKLM and keystore planning in Chapter 9, “Planning for TKLM and its keystores” on page 317.

Depending upon your environment and your operating system platforms, you may use one or more methods of encryption. You do not have to use the same method of encryption for all implementations.

4.4.1 Encryption method comparison

Table 4-4 lists information about encryption policy implementation and encryption key management for each encryption methods, which are Application-Managed Encryption (AME), System-Managed Encryption (SME), and Library-Managed Encryption (LME).

Table 4-4 IBM tape data encryption methods, policies, and key management

Encryption method	Where is the policy defined	Key management
AME	Tivoli Storage Manager	Tivoli Storage Manager
SME	For 3592 drive family: ► DFSMS (z/OS) For 3592or LTO4 drives: ► Atape Device Driver (AIX) ► IBMTape Device Driver (Sun) ► IBMTape Device Driver (Linux) ► IBMTape Device Driver (Windows) ► No HP-UX support	Either: ► Encryption Key Manager (EKM) R1 or later for TS1120 EKM R2 or later for LTO4 EKM R2.1 or later for TS1130 ► Tivoli Key Lifecycle Manager (TKLM)
LME	For 3592 drive family: ► TS3500 Web Interface ► 3494 Web Interface or 3494 Library Manager User Interface ► TS3400 Web Interface For LTO4 drives: ► TS3500 Web Interface ► TS3310 Web Interface ► TS3200 Web Interface ► TS3100 Web Interface ► TS2900 Web Interface	Either: ► Encryption Key Manager (EKM) R1 or later for TS1120 and EKM R2 or later for LTO4 EKM R2.1 or later for TS1130 ► Tivoli Key Lifecycle Manager (TKLM)

4.4.2 System z encryption methods

In System z environments (z/OS, z/VM, z/VSE, and z/TPF), you must always use *System-Managed Encryption*. Application-Managed and Library-Managed Encryption are *not* supported for these operating systems.

4.4.3 Open Systems encryption methods

In the Open Systems environments (including Linux on System z), you usually have a choice of the method of encryption to use. On most operating systems, all three encryption methods are available: AME, SME, and LME. Table 4-5 compares several of the differences and considerations for Open Systems solutions.

Table 4-5 Comparison of Open Systems encryption methods

Method	Policy granularity	Advantages	Disadvantages
AME	Encryption policy is configured at the application GUI. Granularity is application-dependent.	Fewer new responsibilities for storage administrators	Key management is not centralized. Only available currently in Tivoli Storage Manager
SME (using device drivers)	Encryption is configured (on/off) at the host for each device driver instance, for example, the host-to-drive relationship.	Centralized enterprise-class key management Broadest library and non-library coverage	Requires ISV support for IBM tape drive device drivers
LME	Encryption is configured (on/off) at the library GUI for each logical grouping of drives (for example, all drives in a TS3500 logical library). One of: ► Encrypt with default EKM keys ► Barcode Encryption Policy (BEP) for VOLSER ranges of cartridges associated with logical grouping of drives ► Internal Label Encryption Policy (ILEP) currently supported by NetBackup	Centralized enterprise-class key management Broadest application and operating system (OS) coverage	Not available for drives outside an IBM tape library

Note: LME BEP is supported only on the TS3500 and 3494 tape libraries. The following LME policy settings are supported on all tape libraries:

- Encrypt All
- Internal Label - Selective Encryption
- Internal Label - Encrypt All

Note the following considerations about the encryption methods:

- Application-Managed Encryption

  This method might be the most advantageous when a single application is the primary user of tape. For example, all of the tape processing in an Open Systems environment is

related to a single software application, such as a backup and restore application (Tivoli Storage Manager). In this case, having the backup and restore application manage the keys might be the most convenient solution. When you consider implementing an encryption management plan at the application layer using Tivoli Storage Manager, also consider an important point, which is that this software has to be updated on every server that provides data to be encrypted.

- System-Managed Encryption and Library-Managed Encryption

  These methods are perhaps the most logical approaches in environments where tape assets are shared across multiple applications. This is because the transparency of encryption offered through the use of the EKM or TKLM. As with Application-Managed Encryption, updates might be required for certain aspects of the overall system, such as device drivers, operating systems, DFSMS, or controllers, to fully enable encryption.

4.4.4 Decision time

You have to decide which encryption methods are best for your environments. However, in general, we expect most clients to use System-Managed Encryption for their z/OS, z/VM, z/VSE, and z/TPF operating systems (SME is really the only choice) and Library-Managed Encryption for their Open Systems operating systems (including Linux on System z).

In 4.5, “Solutions available by operating system” on page 99, we indicate which encryption methods are available for each operating system platform and tape hardware combination and the required hardware and software prerequisites.

4.5 Solutions available by operating system

In this section, we discuss the support available for each operating system.

4.5.1 The z/OS solution components

This section summarizes the requirements for tape encryption in a z/OS environment.

Operating system support

z/OS supports System-Managed Encryption (SME) through DFSMS on the TS1120 and TS1130 tape drive. The TS1120 and TS1130 tape drive, when encryption-enabled, is supported for attachment to ESCON and FICON channels on System z servers with the 3592-J70 tape controller, the TS1120 tape controller (3952-C06), or the TS7700 Virtualization Engine. This information is summarized in Table 4-6.

Table 4-6 z/OS encryption support

Tape library	TS3500	3494	TS3400	TS3310	TS3200	TS3100
TS1120, TS1130 tape drives attached to controller	SME	SME	SME	No	No	No
TS1120, TS1130 tape drives attached to TS7700	SME	SME	No	No	No	No
LTO4 tape drive	No	No	No	No	No	No

System-Managed Encryption on z/OS requires:

- z/OS and z/OS.e V1R4, or later. and program temporary fixes (PTFs) required for updates to z/OS and DFSMS

  Note: z/OS V1R4, V1R5, V1R6 and V1R7 are no longer in service.

- An Encryption Key Manager component that is available to the z/OS system

z/OS support of the TS1120 tape drive with encryption is available on z/OS V1R4, V1R5, V1R6, V1R7, and V1R8, and is included in the base support for z/OS V1R9 and later.

z/OS support of the TS1130 tape drive with encryption is available on z/OS V1R7 and later. PTFs are required for updates to DFSMS.

Refer to these enabling APARs:

- OA18111 for z/OS V1R4 and V1R5
- OA17562 for z/OS V1R6 and V1R7
- OA15685 for z/OS V1R8
- OA22118 for TS1130 device support for new function
- OA22119 for TS1130 new function

Note: Before you encryption-enable the TS1120 drives, these software levels must be installed. If you enable the drives prior to having the software support, the drives will not come up in z/OS because they now have device-type bytes that are not recognizable by the previous software levels. Although compatibility support maintenance of these systems recognizes the drive, it treats the drive as a non-encryption-enabled 3592-E05 drive.

Before using the encryption-enabled TS1120 or TS1130 tape drive in a sysplex, OAMplex, RMMplex, or HSMplex, ensure that the appropriate support is available and installed at all of the release levels used in the plex. In addition, as appropriate for your environment and release level, determine what coexistence PTFs are required for your environment. For additional information about the support provided, refer to the 3592 preventive service planning (PSP) bucket. We also discuss additional planning details in 4.8.2, “TS1120 and TS1130 compatibility considerations” on page 130.

Note: RACF APAR OA13030 is required on z/OS 1.4, 1.5, 1.6, and 1.7 for greater than 1024 modulus support. This APAR also provides additional support to the RACDCERT command with respect to IBM Integrated Cryptographic Service Facility (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 to 1.7, you must verify that the PTF for this APAR has been applied.

Tape drives

Table 4-7 on page 101 describes tape drive combinations and feature codes.

Table 4-7 Tape drive requirements for z/OS

Tape drive	Machine types and models	Type of update
TS1130	3592-E06	New TS1130
TS1130	3592-EU6	TS1130 upgraded from a TS1120
TS1120	3592-E05	See TS1120 prerequisites, encryption-capable, and encryption-enabled.

Tape libraries

Table 4-8 describes the tape library requirements for z/OS.

Table 4-8 Tape Library requirements for z/OS

Tape library	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22, L23,D22, and D23	ALMS (Depending on model FC 1690, 1692, 1693, or 1694) For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html Library Firmware 8160 or later
TS3500 with TS1120 drives	3584-L22, L23, D22, and D23	For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-L22, D22, and D24	Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-L22, D22, and D24	Firmware updates shipped with FC5596 and FC9596 on the 3592-E05 drive
TS3400 with TS1130 drives	3577-L5U	Library Firmware 0032.0000 or later. Order FC9900 for encryption configuration documentation.
TS3400 with TS1120 drives	3577-L5U	Order FC9900 for encryption configuration documentation.

Control units

Table 4-9 describes control unit combinations and feature code prerequisites.

Table 4-9 Control unit requirements for z/OS

Control unit	Machine types and models	Type of update
3592-J70 controller	3592-J70	Firmware update shipped with 3592-J70 FC5595 or FC9595.
TS1120 tape controller (3952-C06)	3592-C06	Firmware update shipped with 3592-C06 FC5595 or FC9595.
Router for EKM or TKLM Attach (only required on tape controllers for out-of-band support)		FC5593 or FC9593
3494-L*xx*	3494-L*xx*	Firmware shipped for 3494 Library Manager with 3592-J70 or TS1120-C06 FC5595 or FC9595.

Control unit	Machine types and models	Type of update
3953-L05	3953-L05	Firmware shipped for TS3500 Library Manager with 3592-J70 or TS1120-C06 FC5595 or FC9595.
TS3400	3577-L5U	Order FC9900 for Encryption Configuration documentation.

TS7700 Virtualization Engine

Table 4-10 describes the tape library requirements for the TS7700 Virtualization Engine. FC9900 is required on the 3957-V06 component of the TS7700 system to enable encryption within the TS7700. FC0521 provides the latest firmware updates.

Table 4-10 TS7700 Virtualization Engine tape library requirements for z/OS

Tape library	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22, L23, D22, and D23	ALMS (Depending on model FC 1690, 1692, 1693, or 1694) For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html Library Firmware 8160 or later TS7700 microcode must be at 8.5.0.xx or later.
TS3500 with TS1120 drives	3584-L22, L23, D22, and D23	For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-D22	Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-D22	Might require FC5596 and FC9596 on the 3592-E05 drives if the Library Manager is not at microcode level 535 or later.

4.5.2 z/VM, z/VSE, and z/TPF solution components for TS1120 drives

z/VM, z/VSE, and z/TPF support out-of-band System-Managed Encryption for TS1120 drives through the 3592 Model J70 or TS1120 Model C06 tape controllers. Table 4-11 summarizes the supported tape encryption methods.

Table 4-11 z/VM, z/VSE, and z/TPF Encryption methods available

Tape library	TS3500	3494	TS3400
TS1120 and TS1130 tape drives attached to controller	SME	SME	SME

For z/OS prerequisites, refer to the tape libraries and control unit tables (Table 4-6 on page 99 through Table 4-9 on page 101).

Note System-Managed Encryption EKM does not run on z/VM, z/VSE, or z/TPF but is supported through the out-of-band tape controller connection to an EKM server running on z/OS or another EKM platform.

The z/VM support of System-Managed Encryption requires:

- z/VM 5.1 and later with PTFs for APAR VM64063.
- PTF for APAR VM64062 is also required for DFSMS/VM support for z/VSE guests.
- A 3592 Model J70 or TS1120 Model C06 tape controller *with routers* for out-of-band communication with EKM or TKLM.
- An EKM or TKLM available to the tape controller.
- One of the supported tape library and tape drive combinations from Table 4-11 on page 102.

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

- *z/VM CP Planning and Administration*, SC24-6083

  Search for the description of the overall usage of encryption support on z/VM.
- *z/VM CP Commands and Utilities*, SC24-6081

  This book contains specific details about each command.

The z/VSE support of System-Managed Encryption requires:

- z/VSE V4.1 with APAR DY46682 (UD53141 and UD53142)
- z/VSE V3.1 with APAR Dy46685 (UD53143, UD53144, and UD53146) and PK43473 (UK24398)
- PTF for APAR VM64062 is also required for DFSMS/VM support for z/VSE guests.
- DITTO with PK44172. With this APAR, DITTO/ESA for VSE supports tape encryption interactively and with standard VSE JCL in BATCH mode.
- A 3592 Model J70 or TS1120 Model C06 tape controller *with routers* for out-of-band communication with EKM or TKLM.
- An EKM or TKLM available to the tape controller.
- One of the supported tape library and tape drive combinations from Table 4-11 on page 102.

For more information, refer to the latest edition of *z/VSE V4R1.0 Administration*, SC33-8304. Refer to the chapter about implementing hardware-based tape encryption.

The z/TPF support of System-Managed Encryption requires:

- z/TPF with APAR PJ31479
- A 3592 Model J70 or TS1120 Model C06 tape controller *with routers* for out-of-band communication with EKM or TKLM.
- An EKM or TKLM available to the tape controller.
- One of the supported tape library and tape drive combinations from Table 4-11 on page 102.

For more information, refer to the latest edition of z/TPF Operations on the IBM TPF Product Information Center:

`http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp`

TS7700 Virtualization Engine for z/VM, z/VSE, and z/TPF

Support of encrypted tapes within the TS7700 for these operating systems is totally outboard within the TS7700. No operating system maintenance is required. At the TS7700, you can

designate the *Storage Group* and the *Management Class* to be used for a range of virtual VOLSERS. The Storage Group and Management Class can specify the *TS7700 storage pools* to be used, and encryption can be specified for a storage pool within the TS7700. Refer to Chapter 13, “Implementing TS7700 Tape Encryption” on page 421 for more information.

Table 4-12 describes the tape library requirements for the TS7700 Virtualization Engine. The FC9900 is required on the 3957-V06 component of the TS7700 system to enable encryption within the TS7700. FC0521 provides the latest firmware updates.

Table 4-12 TS7700 Virtualization Engine tape library requirements for z/OS

Tape library	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22, L23, D22, and D23	ALMS (depending on configuration FC 1690, 1692, 1693 or 1694) For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html Library Firmware 8160 or later TS7700 microcode must be at 8.5.0.xx or later
TS3500 with TS1120 drives	3584-L22, L23, D22, and D23	For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-D22	Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-D22	Might require FC5596 and FC9596 on the 3592-E05 drives if the Library Manager is not at microcode level 535 or later.

4.5.3 IBM System i encryption solution components

This section summarizes the i5/OS encryption support.

Operating system support

System i supports Library-Managed Encryption (LME) on the following combinations of tape drives and tape libraries. Application-Managed Encryption, while supported, currently has no applications that run on the System i5 platform.

Table 4-13 i5/OS Encryption support

Tape library	TS3500	3494	TS3400	TS3310	TS3200	TS3100
TS1130 tape drive	LME	LME	LME	No	No	No
TS1120 tape drive	LME	LME	LME	No	No	No
LTO4 tape drive	LME	No	No	LME	LME	LME

System i support of LME requires:

- i5/OS V5.2 or later for TS1120
- i5/OS v5.3 or later for TS1130
- i5/OS V5.3 or OS/400® V5.3 or later for LTO4
- An Encryption Key Manager component available to the library
- One of the supported tape drive and library combinations from Table 4-15 on page 105

Tape drives

Table 4-14 describes tape drive combinations and feature codes.

Table 4-14 Tape drive requirements for i5/OS

Tape drive	Machine types and models	Type of update
TS1130	3592-E06 3592-EU6	No features on the drive are required for AME, SME, or LME.
TS1120	3592-E05	See TS1120 prerequisites, encryption-capable and encryption-enabled.
LTO4	3588-F4A or features of the TS3310, TS3200, TS3100 tape libraries and TS2900 tape autoloader	No features on the drive are required for AME, SME, or LME.
TS2340 (stand-alone LTO4)	3580-S43	No features required for AME.
TS2240 (half high stand-alone LTO4)	3580-H4S	No features required for AME.

Tape libraries

Table 4-15 describes the tape library order options and firmware updates.

Table 4-15 Tape library requirements for i5/OS

Tape library and tape drive	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22 and L23 3584-D22 and D23	ALMS (Depending on configuration FC 1690, 1692, 1693, or 1694) Order FC9900. Library Firmware 8160 or later For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3500 with TS1120 drives	3584-L22 and L23 3584-D22 and D23	Order FC9900. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-L22 and D22	Order FC9900. Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-L22 and D22	Order FC9900. Firmware update FC0520
TS3400 with TS1130 drives	3577-L5U	Library Firmware 0032.0000 or later. Order FC9900 for Encryption Configuration documentation.
TS3400 with TS1120 drives	3577-L5U	Order FC9900 for Encryption Configuration documentation.

Tape library and tape drive	Machine types and models	Type of update
TS3500 with LTO4 drives	3584-L32, L52, and L53 3584-D32, D52, and D53	Order FC9900 and FC1604. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3310 with LTO4 drives	3576-L5B and E9U	Order FC9900 and FC5900, Transparent LTO Encryption.
TS3200 with LTO4 drives	3573-L4U	Order FC9900 and FC5900, Transparent LTO Encryption.
TS3100 with LTO4 drives	3573-L2U	Order FC9900 and FC5900, Transparent LTO Encryption

4.5.4 AIX solution components

This section summarizes the AIX support for tape data encryption.

Operating system support

IBM System p running AIX supports:

- System-Managed Encryption (SME) through the Atape device drivers
- Library-Managed Encryption (LME) in conjunction with the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape libraries and the TS2900 tape autoloader
- Application-Managed Encryption (AME) with Tivoli Storage Manager

This information is summarized for library and drive combinations in Table 4-16.

Table 4-16 AIX encryption support

Tape library	TS3500	3494	TS3400	TS3310	TS3200	TS3100	TS2900
TS1130 tape drive TS1120 tape drive	SME, LME, AME	SME, LME, AME	SME, LME, AME	No	No	No	No
LTO4 tape drive	SME, LME, AME	No	No	SME, LME, AME	SME, LME, AME	SME, LME, AME	SME, LME, AME

System-Managed Encryption with AIX requires:

- AIX V5.1, AIX V5.2, AIX V5.3, or later.
- An Encryption Key Manager component available to the AIX system.
- The Atape device driver supporting encryption must be installed, updated, and utilized. You may download it from:

  ftp://ftp.software.ibm.com/storage/devdrvr/AIX/

Device driver

For AIX System-Managed Encryption only, Table 4-17 describes device driver order updates.

Table 4-17 Device driver requirements for AIX

Device driver	Type of update
TS1130 tape drive TS1120 tape drive LTO Ultrium 4 tape drive	Included in the device driver Web download at: ftp://ftp.software.ibm.com/storage/devdrvr/AIX/

Library-Managed Encryption with AIX requires:

- AIX V5.1, AIX V5.2, AIX V5.3, or later.
- An Encryption Key Manager component available to library.
- A TS3500, 3494, or TS3400 tape library with TS1120 drives and 3592 media.
- A TS3500, TS3310, TS3200, or a TS3100 tape library with LTO4 drives and Ultrium 4 media.

Tape drives

Table 4-18 describes tape drive combinations and feature codes.

Table 4-18 AIX Tape drive requirements for encryption

Tape drive	Machine types and models	Type of update
TS1130	3592-E06 3592-EU6	No features on the drive are required for AME, SME, or LME.
TS1120	3592-E05	See TS1120 prerequisites, encryption-capable and encryption-enabled.
LTO4	3588-F4A	No features on the drive are required for AME, SME, or LME.
TS2340	3580-S43	No features required for Application-Managed Encryption
TS2240 (half high stand-alone LTO4)	3580-H4S	No features required for AME.

Tape libraries

Table 4-19 describes tape library order options and firmware updates.

Table 4-19 AIX tape library requirements

Tape library and tape drive	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22 and L23 3584-D22 and D23	ALMS (Depending on configuration FC 1690, 1692, 1693 or 1694) Order FC9900. Library Firmware 8160 or later For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html

Tape library and tape drive	Machine types and models	Type of update
TS3500 with TS1120 drives	3584-L22 and L23 3584-D22 and D23	Order FC9900 for Encryption Configuration documentation. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-L22 and D22	Order FC9900. Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-L22 and D22	Firmware update FC0520
TS3400 with TS1130 drives	3577-L5U	Library Firmware 0032.0000 or later. Order FC9900 for Encryption Configuration documentation.
TS3400 with TS1120 drives	3577-L5U	Order FC9900 for Encryption Configuration documentation.
TS3500 with LTO4 drives	3584-L32, L52, and L53 3584-D32, D52, and D53	Order FC9900 and FC1604, Transparent LTO Encryption. For firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3310 with LTO4 drives	3576-L5B and E9U	Order FC9900 and FC5900, Transparent LTO Encryption
TS3200 with LTO4 drives	3573-L4U	Order FC9900 and FC5900, Transparent LTO Encryption
TS3100 with LTO4 drives	3573-L2U	Order FC9900 and FC5900, Transparent LTO Encryption
TS2900 with LTO4 drives	3572-S4H	Order FC9900 and FC5901, Transparent LTO Encryption

4.5.5 Linux on System z

When the drives are connected to System z using Fibre Channel Protocol (FCP), Linux on System z supports:

- System-Managed Encryption (SME) through the lin_tape device drivers
- Library-Managed Encryption (LME) in conjunction with the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape libraries and TS2900 tape autoloader.
- Application-Managed Encryption (AME) with Tivoli Storage Manager

Table 4-20 on page 109 shows the encryption methods that are available on tape library and tape drive combinations.

Table 4-20 Linux on System z supported encryption methods

Tape library	TS3500	3494	TS3400	TS3310	TS3200	TS3100	TS2900
TS1120 and TS1130 tape drive	SME, LME, AME	SME, LME, AME	SME, LME, AME	No	No	No	No
LTO4 tape drive	SME, LME, AME	No	No	SME, LME, AME	SME, LME, AME	SME, LME, AME	SME, LME, AME

System-Managed Encryption with Linux on System z requires:

- One of the following Linux on System z distributions:
  - SUSE Linux Enterprise Server 9 (SLES 9) or later
  - Red Hat Enterprise Linux (RHEL 4) or later
- An Encryption Key Manager component available to the Linux system
- A lin_tape device driver supporting encryption must be installed, updated, and utilized. Download the lin_tape device drivers from the following Web site:

  `ftp://ftp.software.ibm.com/storage/devdrvr/Linux/`

  Download from the *latest* directory after looking at the Readme files to determine which driver is appropriate for you.

Library-Managed Encryption with Linux on System z requires:

- One of the following Linux on System z distributions:
  - SUSE Linux Enterprise Server 9 (SLES 9) or later
  - Red Hat Enterprise Linux (RHEL 4) or later
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the library
- A TS3500, 3494, or TS3400 tape library with TS1120 or TS1130 drives and 3592 media
- A TS3500, TS3310, TS3200, TS3100 tape library or a TS2900 tape autoloader with LTO4 drives and Ultrium 4 media.

4.5.6 Linux on System p, System x, and other Intel or AMD Opteron servers

System p, System x, and other Intel®-based or AMD™ Opteron™-based Linux servers support:

- System-Managed Encryption (SME) with lin_tape device drivers
- Library-Managed Encryption (LME) on the TS3500, 3494, TS3400, TS3310, TS3200 TS3100 tape libraries and TS2900 tape autoloader.
- Application-Managed Encryption (AME) with Tivoli Storage Manager

Table 4-21 on page 110 shows the encryption methods that are available on tape library and tape drive combinations.

Table 4-21 Linux-supported encryption methods

Tape library	TS3500	3494	TS3400	TS3310	TS3200	TS3100	TS2900
TS1120 and TS1130 tape drive	SME, LME, AME	SME, LME, AME	SME, LME, AME	No	No	No	No
LTO4 tape drive	SME, LME, AME	No	No	SME, LME, AME	SME, LME, AME	SME, LME, AME	SME, LME, AME

System-Managed Encryption with Linux requires:

- One of the following Linux distributions:
  - SUSE Linux Enterprise Server 9 or 10 (SLES 9 or SLES 10)
  - Red Hat Enterprise Linux 4 or 5 (REHL 4, REHL 5) or later
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the Linux system
- The lin_tape device drivers supporting encryption must be installed, updated, and utilized. Download the lin_tape device drivers from the following Web site:

  `ftp://ftp.software.ibm.com/storage/devdrvr/Linux`

Library-Managed Encryption requires:

- One of the following Linux distributions:
  - SUSE Linux Enterprise Server 9 (SLES 9) or later
  - Red Hat Enterprise Linux 4 (REHL 4) or later
  - Asianux 2.0
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the Linux system
- One of the tape drive and library combinations from Table 4-21

Tape drive

Table 4-22 describes the tape drive combinations and feature codes.

Table 4-22 Linux tape drive requirements for encryption

Tape drive	Machine types and models	Type of update
TS1130	3592-E06 3592-EU6	No features on the drive are required for AME, SME, or LME.
TS1120	3592-E05 new drive order	See TS1120 prerequisites, encryption-capable and encryption-enabled.
TS1120	3592-E05 field upgrade for encryption	
LTO4	3588-F4A or features of the TS3310, TS3200, and TS3100 tape libraries	No features on the drive are required for AME, SME, or LME.
TS2340	3580-S43	No features are required for application-managed encryption.
TS2240 (half high stand-alone LTO4)	3580-H4S	No features are required for AME.

Tape libraries

Table 4-23 describes the tape library order options and firmware updates.

Table 4-23 Linux tape library requirements

Tape library and tape drive	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22 and L23 3584-D22 and D23	ALMS (Depending on configuration FC 1690, 1692, 1693, or 1694) Order FC9900. Library Firmware 8160 or later For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3500 with TS1120 drives	3584-L22 and L23 3584-D22 and D23	For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
3494 with TS1130 drives	3494-L22 and D22	Order FC9900. Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-L22 and D22	Firmware update FC 0520
TS3400 with TS1130 drives	3577-L5U	Library Firmware 0032.0000 or later. Order FC9900 for Encryption Configuration documentation.
TS3400 with TS1120 drives	3577-L5U	Order FC9900 for Encryption Configuration documentation.
TS3500 with LTO4 drives	3584-L32, L52, and L53 3584-D32, D52, and D53	Order FC9900 and FC1604, Transparent LTO Encryption. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3310 with LTO4 drives	3576-L5B and E9U	Order FC5900, Transparent LTO Encryption
TS3200 with LTO4 drives	3573-L4U	Order FC5900, Transparent LTO Encryption
TS3100 with LTO4 drives	3573-L2U	Order FC5900, Transparent LTO Encryption
TS2900 with LTO4 drives	3572-S4H	Order FC9900 and FC5901, Transparent LTO Encryption

4.5.7 HP-UX, Sun, and Windows components

This section summarizes the encryption support of operating systems, and the tape drive and tape library requirements.

Operating system support

This section discusses support of HP-UX, Sun Solaris, and Windows systems.

HP-UX systems

HP-UX supports the following encryption methods:

- Library-Managed Encryption on the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape libraries and TS2900 tape autoloader
- Application-Managed Encryption with Tivoli Storage Manager

Library-Managed Encryption with HP-UX requires:

- HP-UX 11.0, 11i v1 and v2, or later for TS1130, TS1120, and LTO4 drives
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape library or TS2900 tape autoloader
- A TS3500, 3494, or TS3400 tape library with TS1130 or TS1120 drives and 3592 media
- A TS3500, TS3310, TS3200, TS3100 tape library or a TS2900 tape autoloader with LTO4 drives and Ultrium 4 media

Sun Solaris systems

Sun Solaris supports the following encryption methods:

- System-Managed Encryption through the IBMTape device drivers
- Library-Managed Encryption in conjunction with the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape libraries and TS2900 tape autoloader
- Application-Managed Encryption with Tivoli Storage Manager

System-Managed Encryption with Sun Solaris requires:

- Sun Solaris 8, 9, and 10
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the Sun Solaris system
- The IBMTape device drivers supporting encryption must be installed, updated, and utilized. The IBMTape device drivers can be downloaded from the following Web site: `ftp://ftp.software.ibm.com/storage/devdrvr/Solaris/`

Library-Managed Encryption with Sun Solaris requires:

- Sun Solaris 8, 9, and 10
- An Encryption Key Manager component available to the library
- A TS3500, 3494, or TS3400 tape library with TS1130 or TS1120 drives and 3592 media
- A TS3500, TS3310, TS3200, TS3100 tape library or a TS2900 tape autoloader with LTO4 drives and Ultrium 4 media

Windows systems

Windows supports:

- System-Managed Encryption (SME) through the IBMTape device drives
- Library-Managed Encryption (LME) in conjunction with the TS3500, 3494, TS3400, TS3310, TS3200, TS3100 tape library or a TS2900 tape autoloader.
- Application-Managed Encryption (AME) with Tivoli Storage Manager

System-Managed Encryption with Windows requires:

- Windows 2000 Server, Windows Server 2003 or Windows Server 2008
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the Windows system

- The IBMTape device drivers supporting encryption must be installed, updated, and utilized. Download the IBMTape device drivers from the following Web site:

  ftp://ftp.software.ibm.com/storage/devdrvr/Windows/

Library-Managed Encryption on Windows requires:

- Windows 2000 Server, Windows Server 2003, or Windows 2008
- An Encryption Key Manager or Tivoli Key Lifecycle Manager available to the library
- A TS3500, 3494, or TS3400 tape library with TS1130 or TS1120 drives and 3592 media
- A TS3500, TS3310, TS3200, TS3100 tape library or a TS2900 tape autoloader with LTO4 drives and Ultrium 4 media

Table 4-24 and Table 4-25 describe the tape drive and the tape library order options and firmware updates.

Tape drives

Table 4-24 lists tape drive combinations and feature codes for the operating systems.

Table 4-24 HP, Sun, and Windows tape drive requirements for encryption

Tape drive	Machine types and models	Type of update
TS1130	3592-E06 3592-EU6	No features on the drive are required for SME, LME, and AME.
TS1120	3592-E05 new drive order	See TS1120 prerequisites, encryption-capable and encryption-enabled.
TS1120	3592-E05 field upgrade for encryption	
LTO4	3588-F4A or features of the TS3310, TS3200, and TS3100 tape libraries	No features on the drive are required for SME, LME, and AME.
TS2340	3580-S43	No features are required for AME.
TS2240 (half high stand-alone LTO4)	3580-H4S	No features required for AME.

Tape libraries

Table 4-25 describes the tape library order options and firmware updates.

Table 4-25 HP, Sun, and Windows tape library requirements

Tape library with tape drive	Machine types and models	Type of update
TS3500 with TS1130 drives	3584-L22 and L23 3584-D22 and D23	ALMS (Depending on configuration FC 1690, 1692, 1693, or 1694) Order FC9900. Library Firmware 8160 or later For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3500 with TS1120 drives	3584-L22 and L23 3584-D22 and D23	Order FC9900. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html

Tape library with tape drive	Machine types and models	Type of update
3494 with TS1130 drives	3494-L22 and D22	Order FC9900. Library Manager must be at microcode level 536.00 or later.
3494 with TS1120 drives	3494-L22 and D22	Order FC9900. Firmware update FC0520
TS3400 with TS1130 drives	3577-L5U	Library Firmware 0032.0000 or later. Order FC9900 for Encryption Configuration documentation.
TS3400 with TS1120 drives	3577-L5U	Order FC9900 for Encryption Configuration documentation.
TS3500 with LTO4 drives	3584-L32, L52, and L53 3584-D32, D52, and D53	Order FC9900 and FC1604. For the firmware update, visit: http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
TS3310 with LTO4 drives	3576-L5B and E9U	Order FC9900 and FC5900, Transparent LTO Encryption
TS3200 with LTO4 drives	3573-L4U	Order FC9900 and FC5900, Transparent LTO Encryption
TS3100 with LTO4 drives	3573-L2U	Order FC9900 and FC5900, Transparent LTO Encryption
TS2900 with LTO4 drives	3572-S4H	Order FC9900 and FC5901, Transparent LTO Encryption

4.5.8 Tivoli Storage Manager

Currently, Tivoli Storage Manager is currently the only application that provides Application-Managed Encryption for TS1130,TS1120 and LTO4 tape drives.

AME is best where operating environments run an application that is already capable of generating and managing encryption policies and keys, such as Tivoli Storage Manager. Policies that specify when to use encryption are defined through the application interface. The policies and keys pass through the data path between the application layer and the TS1130, TS1120, or LTO4 tape drives. Encryption is the result of interaction between the application and the encryption-enabled tape drive and is transparent to the system and library layers.

The following tape drives and libraries are supported:

- TS1130 tape drives in a TS3500 tape library, 3494 tape library, TS3400 tape library, IBM TotalStorage 3952 Model C20 Tape Frame, or rack
- TS1120 tape drives in a TS3500 tape library, 3494 tape library, TS3400 tape library, IBM TotalStorage 3952 Model C20 Tape Frame, or rack
- LTO4 tape drives in a TS3500 tape library, TS3310 tape library, TS3200 tape library, TS3100 tape library, or TS2900 tape autoloader.

For details about setting up Application-Managed Encryption, refer to your Tivoli Storage Manager documentation or visit the following Web site for more information:

http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp

4.6 Ordering information

This section discusses features and microcode levels you might have to order and install to support your encryption solution. We discuss the following prerequisites:

- TS1130 tape drive
- TS1120 tape drive
- LTO4 tape drive
- Tape libraries
- Tape controllers
- TS7700 Virtualization Engine
- General software

4.6.1 TS1120 tape drive prerequisites

For any TS1120 drive to be able to support encryption, it must be *encryption-capable* before it can *enabled (encryption-enabled)* with the particular encryption method (AME, SME, or LME) to be used.

Encryption-capable

Prerequisites for the TS1120 tape drive are:

- A no-charge encryption-capable feature code for new TS1120 tape drives. All TS1120 drives shipped since 8 September 2006 (serial number 13-65000 or greater) require FC9592, which indicates that the drive is encryption-capable.
- If your TS1120 drive shipped prior to 6 September 2006 (serial number 13-50000 to 13-64999), you might have to order an optional chargeable encryption feature upgrade, FC5592, for the installed TS1120 tape drive. This feature code provides the encryption electronics and causes the TS1120 drive to be encryption-capable. The upgrade might contain refurbished parts.

Encryption-enabled

In most situations, the TS1120 tape drives can be encryption-enabled by using the library Web interfaces. In that case, no prerequisite feature codes are required. The actual procedures for encryption-enabling the drives are discussed in the implementation chapters.

The encryption-enabled environments and the prerequisite features (if any) to order are:

- For System z TS1120 tape drives attached to a 3592 Model J70 or a TS1120 Model C06 tape controller, read 4.6.2, “Tape controller prerequisites” on page 116.
- For Open Systems-attached TS1130 drives in a TS3500 tape library, ALMS is required depending on the library configuration FC 1690 for L22 libraries, FC1692, 1693 or 1694 depending on the level of capacity expansion installed. Library Firmware 8160 or later is required. You enable encryption for the drives through the Tape Library Web interface.
- For Open Systems-attached TS1130 drives in a TS3400 tape library, you enable encryption for the drives with the Tape Library Web interface. No prerequisite feature codes are required.
- For Open Systems-attached TS1120 drives in a TS3500 or TS3400 tape library, you enable encryption for the drives with the Tape Library Web interface. No prerequisite feature codes are required.
- For Open Systems-attached or TS7700-attached TS1130 tape drives (in a 3494 tape library with Library Manager licensed internal code level 536 or later is required), the Tape

Library Specialist or the 3494 Library Manager user interface provides the capability for you to enable encryption for the drives. No prerequisite feature codes are required.

- For Open Systems-attached or TS7700-attached TS1120 tape drives (in a 3494 tape library with Library Manager licensed internal code level 535 or later), the Tape Library Specialist or the 3494 Library Manager user interface provides the capability for you to enable encryption for the drives. No prerequisite feature codes are required.
- For Open Systems-attached or TS7700-attached TS1120 tape drives (in a 3494 tape library with a Library Manager licensed internal code level lower than 535), order FC9596 (Plant) or FC5596 (Field) for each drive to be enabled. These features provide instructions and procedures for the service support representative (SSR) to enable encryption for the drive and set the encryption method for the drive.
- For Open Systems-attached TS1120 or TS1130 tape drives in a rack or C20 Silo frame, order FC9596 (Plant) or FC5596 (Field) for each drive to be enabled. These features provide instructions and procedures for the SSR to enable encryption for the drive and set the encryption method for the drive.

4.6.2 Tape controller prerequisites

Support for the TS1120 or TS1130 encryption function installed in any IBM controller attachment requires a minimum level of firmware for the 3592-J70 tape controller, TS1120 tape controller (3592-C06), or 3590 A60 enterprise tape controller (even though the A60 does not support encryption or TS1130) where you intend to use tape cartridges from a common tape cartridge scratch pool.

The minimum level of firmware is 1.19.5.*x* for the 3592-J70 tape controller or 1.21.x.x for the TS1120 tape controller (3592-C06), and 1.16.1.11 for the 3590-A60 enterprise tape controller.

The level of firmware required for the TS1120 tape controller (3592-C06) and the 3592-J70 tape controller ships the first time that you order FC5595 or when you order FC9595 on a new controller. To obtain the microcode required for the 3590-A60 enterprise tape controller, order FC0520 (Function Enhancement Field).

For TS1120 or TS1130 drives attached to the System z platforms z/OS, z/VM, z/VSE, or z/TPF, a TS1120 Model C06 or a 3592 Model J70 tape controller is required. These tape controllers support encryption in the System z ESCON/FICON environment.

To configure and enable the encryption capability of the tape control unit (CU) and attached tape drives, order CU Encryption Configuration, FC9595 (Plant) or FC5595 (Field) on the Model J70 or C06 controllers. CU Encryption Configuration (Field FC5595) must also be ordered when an Encryption Configuration change is required on the tape controller or attached tape drives. One of these CU Encryption Configuration features must be installed whether in-band or out-of-band encryption support is implemented. This feature provides instructions and procedures for the SSR to enable encryption for the tape control unit and to enable encryption and set the System-Managed Encryption method for all of the TS1120 or TS1130 drives attached to the tape control unit.

Note: When one of the CU Encryption Configuration features is installed, it is not necessary to order the Encryption Configuration/Plant or Field (FC9596 or FC5596) on the TS1120 tape drives attached to the controller. The CU Encryption Configuration feature enables encryption for all encryption-capable TS1120 drives attached to the controller. Unless the TS1120 tape drives are installed in a client rack, the minimum level of Library Manager licensed internal code will also be shipped when FC9595 is ordered or the first time that FC5595 is ordered for the control unit.

Tape controller out-of-band support

For ESCON/FICON System z environments using out-of-band support for encryption (z/VM, z/VSE, and z/TPF), routers are required to allow the tape controller to communicate with the Encryption Key Managers (EKM) or Tivoli Key Lifecycle Manager (TKLM). Order FC5593 (Router for EKM Attach), which provides dual routers to allow redundant connections between the tape controller and the EKM or TKLM. The installation of features required for out-of-band support depends on the automation platform supporting the TS1120 or TS1130 tape drives:

- For TS1120 and TS1130 tape drives in the *TS3500 Tape Library*:

  Order one FC5593 on the 3953 Model F05 containing FC5505, Base Frame. This feature supports up to sixteen 3592 tape controllers in a TS3500/3953 Library Manager tape system.

- For TS1120 and TS1130 tape drives in the *3494 Tape Library*:

  Order one FC5593, Router for EKM attach, on the 3494 Library Manager (Model L10, L12, L14, L22, or L24). This feature supports up to seven 3592 tape controllers. FC5246, Dual Path Concentrator, is required on the Library Manager before FC5593 can be installed. A second FC5593, Router for EKM Attach, can be installed on the 3494 Library Manager to support up to a total of fourteen 3592 tape controllers.

  Note: The IBM 3494 Tape Library supports up to fifteen tape controllers. However, the *maximum quantity of two* FC5593s in the 3494 Library Manager supports only up to fourteen 3592 tape controllers.

- For TS1120 or TS1130 tape drives in the *TS3400 Tape Library*:

  Order FC5247, Enhanced Router, on the TS1120 Model C06 controller to which the TS1120 drives are attached. This feature is required when attaching TS1120 drives in the TS3400 library to a TS1120 Model C06 controller, even when no encryption is needed.

- For TS1120 or TS1130 tape drives in the *Storagetek 9310 Powerhorn Automated Tape Library*:

  - When the tape drives are attached to the 3952 Model J70, order FC5593, Router for EKM attach, on the 3590 Model C10 to support the first Model J70. One of FC4860, FC4862, FC9861, or FC9862 is required on the 3590 Model C10.

    To support a second 3592 Model J70 in the 3590 Model C10 frame, order FC5594, Attach Additional CU to Router, on the Model C10. FC5593 is required on the Model C10 to install FC5594.

  - When the tape drives are attached to the TS1120 Model C06, order FC5593 on the 3952 Model F05. FC7315 is required on the 3952 Model F05.

    To support more than one TS1120 Model C06 controller in the same 3952 Model F05 frame, order one of FC5594, Attach Additional CU to Router, for each additional Model C06 to use out-of-band support. FC5593 is required on the 3952 Model F05 to install FC5594. The maximum quantity for FC5594 on the 3952 Model F05 is two.

- For TS1120 or TS1130 tape drives in a client-supplied rack:

  Order one of FC5593 on the 3592 Model J70 or TS1120 Model C06 tape controller, which is contained in 3953 Model F05 containing FC5505, Base Frame. This feature supports up to sixteen 3592 or TS1120 tape controllers in a 3953 Tape System.

For the latest details about specific hardware, software, and Fibre Channel support for the IBM TS1120 Tape Drive, refer to the following Web site:

`http://www-03.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_interop.pdf`

4.6.3 LTO4 tape drive prerequisites

For an LTO4 drive to be able to support encryption, it must be an encryption-capable drive and then it must be enabled (encryption-enabled) with the particular encryption method (SME, LME, or AME) to be used:

- All LTO4 tape drives are encryption-capable (without any features) as long as they are using Ultrium 4 media, therefore, meeting LTO consortium specifications.
- The LTO4 tape drives will be encryption-enabled by using the library Web interfaces. No prerequisite feature codes are required for the drive. The procedures for enabling encryption for the drives are described in the implementation chapters.

4.6.4 Tape library prerequisites

You can configure your hardware setup to support encryption for your business in a variety of ways. We list the tape library and tape controller requirements next.

TS3500 Tape Library prerequisites

The IBM TS1120 and TS1130 Tape Drive can be installed and is supported in the TS3500 Tape Library Models L23, L22, D23, and D22. The IBM LTO4 Tape Drive can be installed and is supported in the TS3500 Tape Library Models L53, L52, L32, D53, D52, and D32. Support for the tape encryption function requires a minimum level of microcode firmware, and it is the client's responsibility to load, configure, and maintain the TS3500 Tape Library.

Important: You must order the no-charge FC9900 (Encryption Configuration) on one of the frames in the TS3500. We suggest ordering this feature for the L23, L22, L53, L52, or L32 frame for consistency, because this feature code is where many other library features are specified. This feature provides instructions and a license key for activating encryption on the TS3500 Tape Library. Client-initiated procedures have to be completed for enabling and configuring the TS3500 Tape Library to support encryption. No additional features are required for TS1120 encryption.

When using an TS1130, Automated Library Management System (ALMS) is required; depending on the library configuration, this may be FC 1690 for L32, L22 or L52 libraries, FC1692, 1693 or 1694 depending on the level of capacity expansion installed.

If you plan to use LME or SME encryption methods for LTO4 drives in the TS3500 Tape Library, you must also order FC1604, Transparent LTO Encryption. The AME encryption method for LTO4 drives does not require FC1604.

The 3494 Tape Library prerequisites

TS1120 and TS1130 encryption support for tape drives in a 3494 tape library requires a minimum level of firmware in the Library Manager. The required level of microcode firmware ships the first time that the client orders FC5595, or when the client orders FC9595 and the IBM 3592 Tape Controller is not located in a client-owned rack (FC4641). This feature applies only to controller-attached TS1130 or TS1120 drives.

FC5220, Ethernet LAN Adapter, is also required on the 3494-L*xx* frame in order to implement Library-Managed Encryption in the 3494. Most 3494s already have this feature code. This feature is not required if you plan to only use SME or AME methods in the 3494.

You must order no-charge FC9900, Encryption Configuration, on the L*xx* frame of the 3494 when you plan to use encryption. This feature provides instructions for activating encryption on the IBM 3494 Tape Library. This feature provides a level of microcode that supports a

capability known as Open Systems Library-Managed Encryption (OSLME). However, OSLME also provides the capability to enable encryption for TS1130 or TS1120 drives for SME and AME methods.

Client-initiated procedures need to be completed for enabling and configuring the 3494 tape library to support OSLME. One of FC5046 (PCI Library Manager-Field), FC5047 (LAN PCI Library Manager-Field), FC9046 (PCI Library Manager-Plant), or FC9047 (LAN PCI Library Manager-Plant) is a required prerequisite to provide a compatible Library Manager PC.

Installation of this firmware can also update the drive microcode firmware on any installed Open Systems-attached 3592 Model J1A tape drive, TS1120 or TS1130 tape drive. With this support, the encryption, WORM, and standard tape cartridges in all sizes (JA, JB, JJ, JR, JW, and JX media) can be intermixed within the same tape library and have one common scratch pool per media type, independent of recording technology and encryption.

The minimum levels of microcode firmware are:

- For OSLME, the minimum level is the Library Manager 535.*xx* code.
- For TS1130, the minimum level is the Library Manager 536.*xx* code.

TS3400 tape library prerequisites

For TS1120 and TS1130 encryption support of LME or SME encryption methods within the TS3400 tape library, order no-charge FC9900, Encryption Configuration, for machine type and model 3577-L5U.

This feature provides publication updates with information about enabling and configuring the IBM TS3400 Tape Library to support encryption. Client-initiated procedures need to be completed for enabling and configuring the IBM TS3400 Tape Library to support encryption with the TS1120 or TS1130 encryption-capable tape drive.

TS1120 drives in an IBM TS3400 Tape Library can either be Open Systems-attached or controller-attached.

The minimum firmware level for the TS3400 to support a TS1130 tape drive is 0032.0000

IBM TS3400 Tape Library attached to IBM TS1120 Tape Controller

The minimum machine code level required on the IBM TS1120 Tape Controller (3592-C06) to support attaching an IBM TS3400 Tape Library is 1.21.3.*xx* or later. You can use FC0520, Controller Licensed internal code Update, to order the most current level of machine code.

To support attaching an IBM TS3400 Tape Library and its drives to an IBM TS1120 Tape Controller, the TS3400 tape library (3577 Model L5U) requires the minimum machine code level 0009.0007 or later. The client is responsible for downloading and installing machine code updates to the IBM TS3400 Tape Library. Refer to the 3577 Sales Manual for more details. The machine code level just described is required for the IBM TS1120 Tape Controller.

To control IBM TS1120 Tape Drives or IBM TS1130 Tape Drives in an IBM TS3400 Tape Library, the IBM TS1120 Tape Controller must be installed in a client-supplied rack that usually contains the TS3400 libraries also. The TS3400 supports one or two TS1120 or TS1130 tape drives and up to eighteen 3592 tape cartridges. A TS1120 tape controller can attach up to seven TS3400 tape libraries. Each TS3400 tape library can only access cartridges in its own library. Only tape drives installed in TS3400 tape libraries can be connected to a TS1120 tape controller that has installed FC9014, Attach TS3400 to CU.

The IBM TS1120 Tape Controller (3592 Model C06) feature codes are:

- FC9014, Attach TS3400 to CU, is required.
- FC4641, Install Controller in Rack, is required.
- FC5247, Enhanced Router, is required to provide management interface connections to the TS3400 library and to the IBM TS3000 System Console (TSSC).
- FC3478, Dual Ported Fibre Adapter, is required on the IBM TS1120 Tape Controller for attachment of 3592 tape drives.

The IBM TS3400 Tape Library (3577 Model L5U) feature codes are:

- FC9014, Attach TS3400 to CU, is required and provides an Ethernet cable to connect the library to the enhanced router in the TS1120 tape controller configuration.
- FC7004, TS3400 Rack Mount Kit, is required.

TS3310 tape library prerequisites

To support any encryption, order no-charge FC9900, Encryption Configuration. If you plan to use LME or SME methods, also order chargeable FC5900, Transparent LTO Encryption. AME encryption method does not require FC5900.

TS3200 or TS3100 tape library prerequisites

Encryption supported on Ultrium 5 and Ultrium 4 Fibre Channel and SAS tape drives: The IBM System Storage TS3100 and TS3200 Tape Libraries support data encryption on the base drive with Ultrium 5 and Ultrium 4 media meeting LTO consortium specifications and Application Managed encryption(AME). System Managed(SME) and Library Managed Encryption(LME) are supported with the Transparent LTO Encryption feature (Feature #5900 or SEO #45E3081). IBM Tivoli Key Lifecycle Manager v1 (TKLM) is required for encryption key management with LTO Ultrium 5 drives.

For full functionality, the TS3100 and TS3200 Tape Libraries Driveless models require IBM LTO Ultrium Tape Drives. IBM Linear Tape-Open (LTO) Ultrium 5 Half-High and Full High 6-Gb SAS and 8-Gb Fibre Channel Drives, enhancing tape performance over the previous generation IBM LTO Ultrium 4 Tape Drives, with a native data transfer rate of up to 140 MB/sec. Ultrium 4 drive features are offered as 4 Gbps Fibre Channel in full-high format, or LVD Ultra160 SCSI or 3 Gbps Serial Attached SCSI (SAS) in either full-high or half-high formats. IBM Ultrium 4 Fibre Channel and SAS tape drives support encryption. Ultrium 3 drive features are offered as 3 Gbps Serial Attached SCSI (SAS) in half-high formats.

TS2900 tape autoloader prerequisites

To support any encryption, order no-charge FC9900, Encryption Configuration. If you plan to use LME or SME methods, also order chargeable FC5901, Transparent LTO Encryption. The AME methods do not require FC5901.

4.6.5 Other library and rack Open Systems installations

When you install encryption-capable TS1120 tape drives or TS1130 tape drives in either a 3592 Model C20 Silo Compatible Frame or in a 19-inch rack and if you intend to use tape cartridges from a common scratch pool, minimum microcode levels are required for any other 3592-J1A drive or any TS1120 drive without the Encryption capability. These microcode levels enable the non-encryption-capable drives to recognize the encrypted data format on a cartridge.

The minimum level of microcode firmware is:

- D3I0_851 for the IBM 3592-J1A Enterprise Tape Drive
- D3I1_919 for the IBM TS1120 Tape Drive (3592-E05)

The minimum level of microcode firmware with TS1130 E3 formatted media is:

- D3I0_C90 for the IBM 3592-J1A Enterprise Tape Drive
- D3I1_DA0 for the IBM TS1120 Tape Drive (3592-E05)

You may upgrade the drive microcode levels or order these microcode levels as FC0500 on the drive.

4.6.6 TS7700 Virtualization Engine prerequisites

This TS7700 Encryption function was initially introduced in the TS3500 tape library with the TS7700 Virtualization Engine microcode version 8.2.0.19 in March 2007. The TS7700 Encryption function is now also supported in the 3494 tape library. Current microcode requirements are:

- If the TS1130 drives are in a TS3500 library:
  - TS7700 Virtualization Engine licensed internal code level 8.5.0.xx or later
  - With 8.5.0.xx the 3943 Library Manager is no longer required.
- If the TS1120 drives are in a TS3500 library:
  - TS7700 Virtualization Engine licensed internal code level 8.2.0.19 or later
  - 3953 Library Manager licensed internal code level 534.31 or later
- If the TS1130 drives are in a 3494 library:
  - TS7700 Virtualization Engine microcode level 8.5.0.xx or later
  - 3494 Library Microcode level 536.0 or later
- If the TS1120 drives are in a 3494 library:
  - TS7700 Virtualization Engine microcode level 8.2.0 27 or later
  - 3494 Library Microcode level 534.31 or later

For encryption support in the TS7700 (SME only), order no-charge FC9900, Encryption Configuration, against the 3957-V06 component of the TS7700 system. This feature provides instructions and procedures for you to enable encryption microcode within the TS7700.

There is no host software maintenance required for the TS7700 encryption function, if software already supports the TS7700 without encryption.

4.6.7 General software prerequisites for encryption

In this section, we describe general information about the operating system software requirements. Specific information about the software release levels is discussed in the following sections where each operating system is described in detail. The following list describes support for encryption on environments and availability:

- Support for encryption on the *TS1120 or TS1130 in the z/OS* operating system environment is available through the 3592-J70 controller, the TS1120 tape controller (3592-C06), or through the TS7700 Virtualization Engine (3957-V06).
- Support for encryption on the *TS1120 or TS1130 in the z/VM, z/VSE,* or *z/TPF* operating system environments is available through out-of-band encryption through the 3592-J70 controller, the IBM TS1120 Tape Controller (3592-C06), or with outboard static encryption specifications in the TS7700 Virtualization Engine (3957-V06).

- Support for encryption on *TS1120 or TS1130 in Open Systems* environments is provided in i5/OS, AIX, HP-UX, Linux, Sun, Microsoft Windows Server 2000, Microsoft Windows 2003 Server or Microsoft Windows 2008 operating system environments.
- Support for encryption on *LTO4 in Open Systems* environments is provided in i5/OS, AIX, HP-UX, Linux, Sun, Microsoft Windows Server 2000, Microsoft Windows 2003 Server, or Microsoft Windows 2008 Server operating system environments.

The installation of a TS1120, TS1130 or a LTO4 drive with encryption can require code updates for System z, System i, System p, and supported Open Systems device drivers or storage management software. The IBM account team or IBM Business Partner must confirm that the client checks the appropriate preventive service planning (PSP) buckets for System z environments or the equivalent support levels required for their particular software environment prior to the installation of the TS1130, TS1120 with encryption or the LTO4 with encryption. A solutions assurance call is required at a minimum for the installation of the first new TS1130, TS1120 tape drive or LTO4 tape drive with encryption activated in an account.

To obtain an update of the Open Systems device drivers, use anonymous FTP from:

`ftp://ftp.software.ibm.com/storage/devdrvr/`

Then, look in the particular directories that represent your operating system environments.

For details about supported software versions and release levels for the TS1130 tape drive, and hardware support information, refer to the following Web site and follow the link to the System Storage Interoperation Center:

`http://www.ibm.com/systems/storage/tape/ts1130/index.html`

For details about supported software versions and release levels for the TS1120 tape drive, as well as hardware support information, refer to the following Web site:

`http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_interop.pdf`

4.6.8 TS1120 and TS1130 supported platforms

The TS1120 and TS1130 tape drives are supported in the widest range of mainframe and Open Systems environments:

- Mainframe-attached:
  - IBM System z running z/OS using ESCON or FICON channels
  - IBM System z running z/VM using ESCON or FICON channels
  - IBM System z running z/VSE using ESCON or FICON channels
  - IBM System z running z/TPF using ESCON or FICON channels

  Note: A tape controller is required for attachment to ESCON or FICON channels on IBM mainframe servers.

- Open Systems-attached:
  - IBM System i
  - IBM System p
  - IBM System x
  - Sun Solaris servers
  - Hewlett-Packard servers
  - Linux-based servers (including Linux on System z using FCP channels)
  - Intel-compatible servers Microsoft Windows 2000 Server, Windows Server 2003, Windows Server 2008

TS1120 and TS1130 tape drives are available in the TS3500, 3494, or TS3400 tape libraries. They are also available in silos and in rack-mounted configurations. The encryption solutions vary depending upon the location of the drives. Table 4-26 on page 124 shows the encryption options available for the TS1120 and TS1130 in the mainframe environments while Table 4-27 on page 124 shows the encryption options available for the TS1120 and TS1130 in Open Systems environments.

Table 4-26 Mainframe-attached TS1120 and TS1130 encryption solution options

Mainframe-attached IBM tape library	Mainframe-attached rack or silo
SME in-band (z/OS only)	SME in-band (z/OS only)
SME out-of-band (z/VM, z/VSE, and z/TPF)	SME out-of-band (z/VM/, z/VSE, and z/TPF)

Table 4-27 Open Systems-attached TS1120 and TS130 encryption solution options

Open Systems-attached IBM tape library	Open Systems-attached rack or silo
AME (Tivoli Storage Manager only) SME (AIX, Solaris, Windows, or Linux only) LME (TS3500, 3494, or TS3400 tape library)	AME (Tivoli Storage Manager only) SME (AIX, Solaris, Windows, or Linux only)

Additional information about TS1120 and TS1130 tape drives is in the announcement letter.

4.6.9 IBM LTO4 tape drive supported platforms

The IBM LTO4 tape drive is supported in a wide range of Open Systems environments:

- IBM System i
- IBM System p
- IBM System x
- Sun Solaris servers
- Hewlett-Packard servers
- Linux-based servers
- Intel-compatible servers Microsoft Windows 2000 Server, Windows Server 2003, or Windows Server 2008

Encryption-capable LTO4 tape drives are available in the TS3500, TS3310, TS3200, TS3100 tape libraries and the TS2900 tape autoloader. They are also available as a drive only with the TS2340 and TS2240. Table 4-28 shows the encryption options available for these environments.

Table 4-28 Open Systems-attached LTO4 encryption solution options

Open Systems-attached in IBM tape library	Open Systems-attached TS2340 drive
AME (Tivoli Storage Manager only) SME (AIX, Solaris, Windows, or Linux only) LME (TS3500, TS3310, TS3200, TS3100 tape libraries and TS2900 tape autoloader)[a]	AME (Tivoli Storage Manager only) SME (AIX, Solaris, Windows, or Linux only)

a. TS3310, TS3200, TS3100 and TS2900 do not support the Barcode Encryption Policy of LME. LME on the TS3310, TS3200, TS3100, TS2900 is all or nothing (entire partition or not).

For more information:

- Obtain information about the LTO4 tape drive in the announcement letter.
- Obtain an update of the Open Systems device drivers through anonymous FTP from:

  `ftp://ftp.software.ibm.com/storage/devdrvr/`

  Look under the specific directory that reflects your operating system environment.

- Refer to *IBM Tape Device Driver Installation and User's Guide,* GC27-2130, available at:
  ftp://ftp.software.ibm.com/storage/devdrvr/

4.7 Other planning considerations for tape data encryption

In this section, we describe other planning considerations for you to evaluate before implementing tape data encryption. You must consider many factors when you plan how to set up your encryption strategy.

4.7.1 In-band and out-of-band

System-Managed Encryption (SME) is the only encryption method available for z/OS environments and requires the Encryption Key Manager (EKM) or Tivoli Key Lifecycle Manager (TKLM) program. See Figure 4-1.

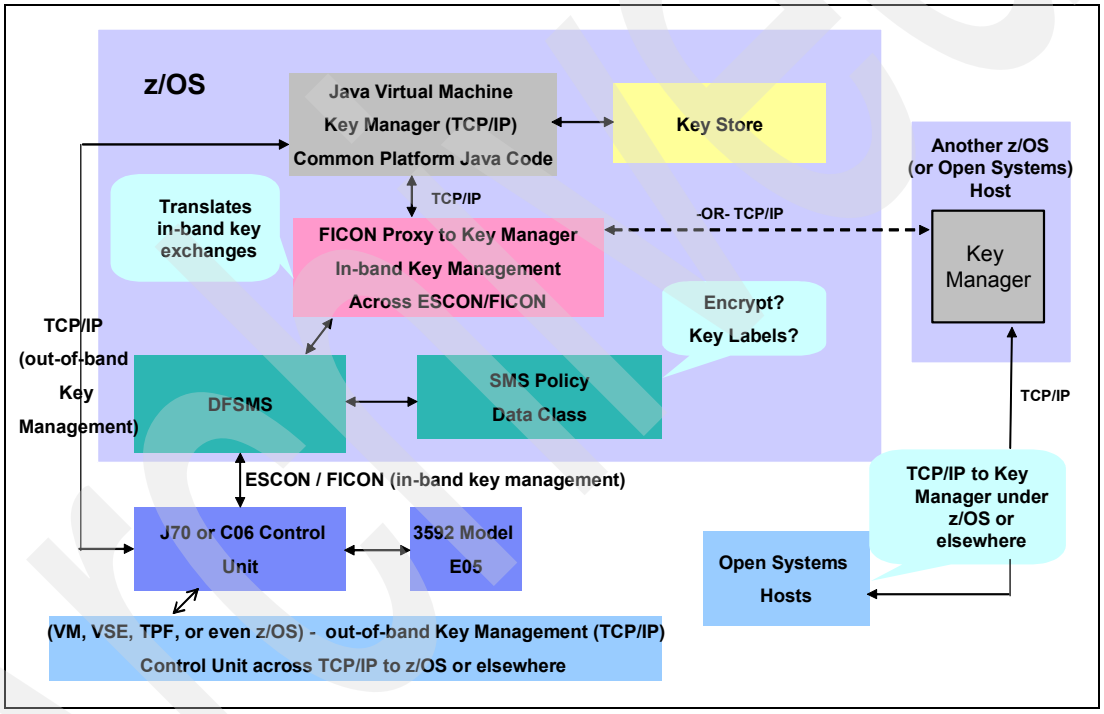


Figure 4-1 z/OS in-band and out-of-band centralized key management

SME on z/OS has two configuration options that are shown in Figure 4-1. Which type of setup makes sense for you?

The first key flow that we describe is in-band encryption where tape drive requests to the EKM or TKLM component travel over the ESCON/FICON channels to the server proxy that is TCP/IP-connected to the EKM or TKLM. The second key flow is out-of-band encryption where the tape controller establishes the communication to the EKM or TKLM server over TCP/IP connections between the tape controller and the EKM or TKLM server. A router is required for out-of-band and EKM support.

In-band encryption

The in-band z/OS proxy allows key management information to be exchanged with a tape drive over existing ESCON/FICON, instead of requiring the deployment of a secondary IP network. The reliability and physical security of the existing I/O attachments are significant reasons that many clients choose the in-band key management path to the EKM or TKLM. The z/OS proxy interface communicates with the tape drive (through the control unit) in the current data path and then uses a TCP/IP connection to communicate with the Encryption Key Manager. Because in-band encryption is managed by the I/O supervisor (IOS), it also allows you to display and alter EKM or TKLM primary and secondary server addresses from the operator console.

Out-of-band encryption

With out-of-band encryption, the EKM and TKLM server addresses are only visible on the Library Manager Console. One other consideration is that with out-of-band encryption, any z/OS image using a drive also has to use the EKM or TKLM for which that drive was set up. With in-band encryption, you can potentially have each z/OS image point to a different EKM or TKLM, with each pointing to a different keystore. This allows images sharing drives to be able to use different keystores. You might find this useful if you have to support a client or an application where each client requires its own keystore for security or regulatory needs.

For z/OS tapes, either method allows a client to configure whether to encrypt based on Data Class definitions. Furthermore for z/OS, a client can specify their key labels through Data Class or through the DD statement (JCL, dynamic allocation, or TSO ALLOCATE). In addition to this, EKM-assigned defaults can also be used if the key labels are not provided through z/OS. For tapes that will be encrypted or decrypted, clients must define and keep track of the key information to be used. Also, DFSMSrmm keeps track of the key labels that were used for a given cartridge.

4.7.2 Performance considerations

Unlike software encryption or encryption appliances, the TS1130, TS1120 or the LTO4 encryption solutions can encrypt data with minimal data transfer performance impact and without requiring additional equipment in the computing environment. You might be concerned if encryption will impact the data transfer performance of your applications or backup processing. Extensive testing shows little degradation to data transfer performance with encryption-enabled drives. The data rate claims of the drive remain unchanged.

With encryption enabled, when writing from loadpoint, the access time of the first write from the beginning of tape increases because of the time needed to retrieve, read, and write the encryption keys. In z/OS, this added time is detected in OPEN processing, the time between the mount message and the IEC705I "`Tape On`" message. Refer to Chapter 6, "Implementing EKM" on page 159 and Chapter 9, "Planning for TKLM and its keystores" on page 317 for more information. Reading an encrypted cartridge also increases the mount time because of the time necessary to retrieve the encryption keys.

4.7.3 Encryption with other backup applications

All backup applications currently supported on any of the IBM tape libraries can support Library-Managed Encryption, because the application is not involved in the encryption process. Applications include, for example Symantec NetBackup, EMC NetWorker, CommVault Galaxy, and BakBone NetVault.

4.7.4 ALMS and encryption in the TS3500 library

The Advanced Library Management System (ALMS) is required in a TS3500 with TS1130 drives.

Although the ALMS is not required for encryption in a TS3500 with TS1120 or LTO4 drives, best practices suggest using it. Encryption in a TS3500 tape library with ALMS is configured by the logical library. Encryption in a TS3500 tape library without ALMS is configured by the physical library.

TS3500 ALMS Encryption rules

For NON-ALMS TS3500 libraries, we enforce homogeneous encryption rules for TS1120 and earlier 3592 and all LTO drives, separately by drive type. The drive type is defined as 3592 or LTO. ALMS is REQUIRED for TS1130 drives. This section discusses the rules. For more information see *IBM System Storage TS3500 Tape Library Advanced Library Management System (ALMS) and Encryption*, by Anthony Abete at:

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101038`

Rule 1: TS3500 non-ALMS, TS1120 drives only library

All 3592 drives in the entire library must be encryption capable for encryption to be enabled. The entire physical library (all logical libraries if partitioned) must consist of encryption capable TS1120 (3592-E05) drives. If encryption is to be enabled, it must be enabled for all drives in the entire physical library, and the drives need to be all managed in the same manner, that is, all LME, all SME, or all AME.

Rule 2: TS3500 non-ALMS, LTO drives only library

The entire physical library (all logical libraries if partitioned) must consist of LTO4 drives before encryption can be enabled. No LTO 1, LTO 2, or LTO3 drives are allowed in the entire physical library. If the library is partitioned, all logical libraries must have encryption enabled in the same manner, that is, all LME, all SME, or all AME.

Rule 3: TS3500 non-ALMS, mixed TS1120 and LTO drives

In this environment, both drive types (LTO and 3592) are to be encryption-enabled. For non-ALMS TS3500 libraries, we enforce homogeneous encryption rules for all 3592 and all LTO drives, separately by drive type.

If you intend to enable encryption for both LTO and 3592, you must adhere to rules 1 and 2 with the following exceptions:

- All LTO logical libraries must be managed in the same manner, that is, SME, LME, and AME.
- All 3592 logical libraries must be managed in the same manner, that is SME, LME, and AME.

However, LTO and 3592 tape drives can be managed differently. For example, all LTO drives can be LME while all 3592 drives can be SME or all AME.

Rule 3A: TS3500 non-ALMS, mixed 3592 and LTO drives

In this environment, only LTO or only TS1120 intend to have encryption enabled.

You have to adhere to the rules only if you intend to enable encryption for that drive type. If you intend to enable 3592 encryption only and not LTO encryption, you only have to adhere to rule 1. If you intend to enable LTO encryption only and not 3592 encryption, you have to adhere only to rule 2.

Rule 4: TS3500 ALMS-enabled, 3592 drives only library

With ALMS enabled, all drives in the physical library do not need to be encryption-capable. That is, the physical library can consist of both encryption-capable and 3592 drives that are not encryption-capable.

All drives in the logical library must be encryption-capable if using LME or AME. All drives in a SME-managed logical library do *not* need to be encryption-capable.

Rule 5: TS3500 ALMS-enabled, LTO drives only library

With ALMS enabled, all LTO drives (in the physical library *or* in the logical library) do not need to be encryption-capable for encryption to be enabled. For example, a logical library can consist of LTO4, LTO2, and LTO3 drives, and yet the LTO4 drives can be encryption-enabled using SME, LME, and AME.

Rule 5A: TS3500 ALMS-enabled, mixed 3592 and LTO drives

You only need to adhere to rules 4 and 5 if you intend to enable encryption for that drive type. If you intend to enable 3592 encryption only and not LTO encryption, only rule 4 applies. If you intend to enable LTO encryption only and not 3592 encryption, only rule 5 applies. If you intend to implement encryption on both 3592 and LTO, rules 4 and 5 both apply.

Note: ALMS is required with new TS3500 installations, when TS1130 tape drives are installed in the library, or when a TS7700 is attached to the TS3500.

In summary:

- On an existing TS3500 library without ALMS

  Without ALMS, implementing encryption on an existing library is very inflexible. It also can be costly, because older technology cannot coexist with newer encryption-capable technology.

- On a newly ordered library without ALMS

  Without ALMS, implementing encryption is more difficult to manage and not very flexible. This environment is useful only if you intend to implement encryption on a new library that will not change over time. All logical libraries require the same encryption method, which makes management an issue when you have to create non-encrypted cartridges.

- On a new or existing TS3500 library with ALMS

  With ALMS, implementing encryption is easily managed, flexible, and much more cost-effective, regardless of your library configuration. This environment is cost-effective, because older technology can coexist in the same physical library with newer encryption-capable technology. Management is much easier, because multiple encryption methods can be used within the same library. This environment is more flexible, because a logical partition can consist of both old and new encryption-capable technology.

On 29 August 2006, IBM announced entry and intermediate-priced offerings of ALMS that mesh with existing Capacity on Demand (CoD) library features. This announcement provides full ALMS functionality for smaller libraries at a lower entry fee and lessens the impact of cost as a barrier.

4.7.5 TS1120 and TS1130 rekeying considerations

You also have to consider rekeying requirements in your planning. Rekeying can be required as part of sharing the same tape with your business partners.

It can also be a requirement if you have certificates or private keys that you expect have been compromised. This compromise is analogous to losing your house keys and then calling a locksmith to rekey all of the locks in your house. Then, the lost keys cannot be used to get into your house.

We consider the business partner situation first. If you plan to share the same information with multiple business partners, you have additional planning considerations. If you share information today by sending multiple tapes at the same time, you can continue to do that, but you have to write the tapes for each business partner using the individual business partner's unique public key (TS1120 and TS1130).

If you pass the same tape from one business partner to another business partner, you have to consider certain changes if you encrypt that tape and do not want to share your private key. The tape going to Business Partner A needs to be written using Business Partner A's public key. When Business Partner A finishes with the tape, they need to send the tape back to you. At this point, you have two options. You can use the rekeying function to rewrite just the key labels on the tape. Rekeying allows you to change the public key alias label used from business partner A's public key to business partner B's public key without having to rewrite the complete data portion of the tape. Details for performing rekeying in z/OS are provided in 12.9, "TS1120 and TS1130 tape cartridge rekeying in z/OS" on page 416.

The approach for compromised certificates is similar. You use rekeying to make the tape unreadable by anyone possessing the compromised certificate or private key.

4.8 Upgrade and migration considerations

In this section, we provide you with important information for upgrading or migrating to tape data encryption.

4.8.1 Look out for potential problems

We found several common errors of which you need to be aware:

- Although multiple systems can be attached to a tape drive, the systems cannot use the drive simultaneously.
- TS1130 and TS1120 tape drives cannot be attached to the same 3592 Model J70 controller with 3590 tape drives.
- TS1130 and TS1120 tape drives are not supported for attachment to a 3590 controller model A60, A50, or A00.
- TS1120 tape drives will operate in J1A Emulation mode when they are attached to the same 3592-J70 or C06 controller with 3592 Model J1A tape drives. This mode is set automatically by the microcode. In this mode, the TS1120 tape drives read and write only in the J1A format at the J1A capacity and approximate performance ratings. This configuration requires J70 code level 1.19.1.*xx* or later and Library Manager 532.*xx* or later. *Encryption is not allowed in J1A Emulation mode.*
- Under z/OS system-managed tape support, TS1120 tape drives that are encryption-enabled are not supported under the same 3592-J70 or C06 controller with TS1120 tape drives that are not encryption-enabled. With z/OS system-managed tape support, although a mix of TS1120 tape drives can exist in the library, the TS1120 tape drives under the same control unit need to be homogeneous and support the same capabilities. Mixed capabilities require encryption-enabled TS1120 drives under one controller and non-encryption-enabled drives under a different controller.

- TS1120 tape drives attached to a 3494 VTS Model B10 or B20 operate in J1A Emulation mode. This mode is set automatically by the microcode. In this mode, the TS1120 tape drives read and write only in the J1A format at the J1A capacity and approximate performance ratings. This configuration requires VTS code 2.32.745.*x* or later and Library Manager 532.*xx* or later.
- 3592-J1A drives attached to the 3494 VTS Models B10 or B20 cannot be replaced by TS1120 tape drives. You can however add TS1120 tape drives to already installed J1A drives (up to a total of 12). The TS1120 drives when attached to a VTS operate in J1A Emulation mode.
- The B10 and B20 VTS do not support the enabled IBM TS1120 Tape Drive encryption feature.

4.8.2 TS1120 and TS1130 compatibility considerations

Several compatibility considerations exist, from both a hardware and a software perspective.

IBM TS1130 Tape Drive modes

The TS1130 can operate in one of 5 modes:

EEFMT3	Enterprise Tape Format 3with encryption enabled
EFMT3	Enterprise Tape Format 3
EEFMT2	Enterprise Tape Format 2 with encryption enabled
EFMT2	Enterprise Tape Format 2
EFMT1	Read-only Enterprise Tape Format 1(note that the TS1130 does not emulate a 3592-J1A drive)

The TS1130 is read and write compatible with the IBM TS1120 Tape Drive and is read compatible with the IBM 3592 Model J1A tape drive:

- The IBM TS1130 Tape Drive cannot read cartridges written by the 3590 or 3490. Cartridges written by the IBM TS1130 Tape Drive cannot be read by the 3590 or 3490. Even though the cartridges are similar in size, they contain different media and thus are not interchangeable.
- The 3592-J1A cannot read or append to cartridges that were created on an IBM TS1130 Tape Drive in EFMT3, EEFMT3, EFMT2 or EEFMT2 mode.
- The TS1120 cannot read or append to cartridges that were created on an IBM TS1130 Tape Drive in EFMT3 or EEFMT3 mode.

IBM TS1120 Tape Drive modes

The TS1120 can operate in one of three modes:

EEFMT2	Enterprise Tape Format 2 with encryption enabled
EFMT2	Enterprise Tape Format 2
EFMT1	Enterprise Tape Format 1, which is also called J1A Emulation mode

In J1A Emulation mode, the TS1120 cannot be enabled for Encryption, but it is read and write compatible with the IBM 3592 Model J1A tape drive:

- The IBM TS1120 Tape Drive can read and append to cartridges written by the IBM 3592 Model J1A tape drive.
- The IBM TS1120 Tape Drive can write cartridges in a 3592 Model J1A format that can be read and appended to by the IBM 3592 Model J1A tape drive.
- The IBM TS1120 Tape Drive cannot read cartridges written by the 3590 or 3490. Cartridges written by the IBM TS1120 Tape Drive cannot be read by the 3590 or 3490.

Even though the cartridges are similar in size, they contain different media and thus are not interchangeable.

- The 3592-J1A cannot read or append to cartridges that were created on an IBM TS1120 Tape Drive in either EFMT2 or EEFMT2 mode.
- The IBM TS1120 Tape Drive can be attached to the same 3592 Model J70 or C06 controller, TS7700 Virtualization Engine, or 3494 VTS Models B10 or B20 with 3592 Model J1A tape drives, but this configuration is only supported when the TS1120 is running in J1A Emulation mode. When TS1120 drives are set to run in E05 native mode, intermixing is not supported.

Upgrade and migration considerations exist when you move from an existing environment to an encryption environment.

Coexistence support for the encryption-enabled IBM TS1120 Tape Drive is provided at z/OS V1R4 and later by installing the necessary full-support program temporary fixes (PTFs) without the device services enabling PTF. In addition to this, existing device services support prevents the encryption-enabled TS1120 tape drives from coming online on a system that does not have all of the full-support PTFs installed. Installation of the device services enabling PTF brings in all of the required full-support PTFs.

Device services provides coexistence support to allow the encryption-enabled IBM TS1120 Tape Drive to come online, but until the full support is installed, it appears to the host as a 3592-2 without encryption capability. You must install the needed coexistence support on systems that will not have all of the encryption-enabled IBM TS1120 Tape Drive support installed.

DFSMSdss

DFSMSdss, a z/OS functional component, allows you to copy, move, dump, and restore data sets and volumes. DFSMSdss is the primary data mover of DFSMS/MVS.

Using encryption-enabled TS1120 tape drives does not require changes to your installation's DFSMSdss jobs. It does, however, require changes to your installation's Data Classes and DFSMShsm dump classes. We briefly summarize these considerations here for DFSMSdss administrators.

Using an encryption-enabled IBM TS1120 Tape Drive requires changes to your SMS Data Class definitions. For tapes that require software (or host-based) encryption, ensure that your dump-requesting jobs use only tape drives that are not enabled for hardware encryption. To do so, check the Data Classes of the output ddnames to ensure that the jobs do not specify a Data Class that requests encryption from the encryption-enabled tape drives.

Stand-alone drives

z/OS DFSMS and related program products provide full support for the encryption-enabled IBM TS1120 Tape Drive and MEDIA5, MEDIA6, MEDIA7, and MEDIA8 with z/OS V1R4 and later, with support for media types MEDIA9 and MEDIA10 provided with z/OS V1R5 and later.

The encryption-enabled IBM TS1120 Tape Drive support enables the tape drives to operate in a stand-alone environment in 3590 Model B1*x* Emulation mode and to coexist with other emulated tape drives. However, prior to using the encryption-enabled IBM TS1120 Tape Drive, ensure that all existing 3592 Model J1A and all existing (base support) 3592 Model E05 tape drives have their microcode upgraded to recognize and enable the EEFMT2-formatted cartridges to be relabelled and reused on the 3592 Model J1A and the base 3592 Model E05. Also, ensure that `VOLNSNS=YES` is in the DEVSUP*xx* member of PARMLIB. Otherwise, job failures can occur with a drive with the incorrect microcode load allocated.

In the stand-alone (non-system-managed) environment, all of the TS1120 Model E05 devices under the same control unit should be homogeneous for easier separation and management, even though the control unit allows a mix of TS1120 Model E05 devices (encryption-capable and non-encryption-capable).

IBM tape library

z/OS DFSMS and related program products provide full support for the encryption-enabled IBM TS1120 Tape Drive and MEDIA5, MEDIA6, MEDIA7, and MEDIA8, with z/OS V1R4 and later, with support for media types MEDIA9 and MEDIA10 provided with z/OS V1R5 and later. The system-managed tape library support allows the tape drives to operate in an ATL or MTL environment as 3590 Model B1*x* devices, providing device allocation and tape media management support. As appropriate for the library type and model, this support allows the encryption-enabled IBM TS1120 Tape Drive to coexist with other emulated 3590-1 tape drives in the same tape library. However, prior to using the encryption-enabled IBM TS1120 Tape Drive, ensure that all existing 3592 Model J1A and all existing (base support) 3592 Model E05 tape drives have their microcode upgraded to recognize and enable the EEFMT2-formatted cartridges to be relabelled and reused on the 3592 Model J1A and the base 3592 Model E05. Also, ensure that `VOLNSNS=YES` is in the DEVSUP*xx* member of PARMLIB. Otherwise, job failures can occur with a drive with the incorrect microcode load allocated.

In addition, in the system-managed tape library environment, all TS1120 Model E05 drives under the same control unit must have the same recording format capabilities and report under the same ERDS physical identifier (EPI). So, if one of the TS1120 Model E05 devices is encryption-enabled, all of the TS1120 Model E05 devices under that same control unit must also have encryption enabled. This setup ensures that all of the devices under the same control unit are homogeneous and that each device under the same control unit is capable of handling the request.

A tape configuration database (TCDB) with EEFMT2 volume records can coexist with lower level systems. Coexistence support is provided at z/OS V1R4 and later to enable, during job processing, a scratch volume that was previously written with an up-level recording format to be used by a lower level system that does not recognize the recording format. Because there is only one scratch pool per media type and that scratch pool can be used across systems at different levels of support, this support ignores the recording format in which the volume was previously written and enables the scratch volume to be used on the lower level system.

DFSMSdfp Device Services/Asynchronous Operations Manager

Coexistence is provided in the Device Services Exit for z/OS V1R4 and later. This capability allows an encryption-enabled 3592 Model E05 drive to come online as a non-encryption-capable 3592 Model E05 with the EPI value stored as `X'12'` in the UCB class extension (the UCBCXEPI field in the IECUCBCX mapping macro). This setup allows an encryption-enabled drive to be used as a non-encryption-capable drive on systems that do not have the full function support installed.

When the Device Services full function support APAR is installed, the Device Services Exit checks whether the enabling APAR is also installed. If it is, the Device Services Exit records the EPI value in the UCB class extension as `X'13'`.

The coexistence support recognizes the new EPI value and displays the real device type as 3592-2 for the **`DS QTAPE,MED`** command, because the new encryption-enabled 3592 Model E05 drive looks and acts exactly the same as a non-encryption-capable drive in coexistence systems that do not have the full function support installed.

HSMplex

In an HSMplex, all systems in the HSMplex must have full support for the TS1120 tape subsystem before any instance of DFSMShsm uses tape hardware encryption. If any system does not fully support tape hardware encryption in an HSMplex with tape hardware-encrypted tapes, a request for tape input can fail because an encryption-enabled 3592 device is not available on that system.

OAMplex

For object access method's (OAM) object support, coexistence considerations exist that you must take into account for your installation before you install the software in an OAMplex:

- For the encryption-enabled IBM TS1120 Tape Drive support, OAM object tape coexistence support is provided at z/OS V1R4 and later through the installation of the full support PTF without the Device Services enabling PTF.
- OAM coexistence support prevents lower level systems from selecting volumes with ERDS Physical Identifier (EPI) values for object write requests that are not supported on that system.
- OAM object support has coexistence considerations when running in an OAMplex environment with at least one system with the full support installed and enabled and at least one system at a release level where the new devices are supported, however, all of the support is not installed and enabled. In this mixed environment, it is possible for a retrieve request to be received for an object that resides on a tape cartridge volume written in the EEFMT2 format by a system that does not have the encryption-enabled IBM TS1120 Tape Drive support installed. Coexistence support is provided that allows OAM to attempt to locate an instance of OAM in the OAMplex where the full support is installed and enabled. If an instance of OAM is located where the request can be processed, the OAM on the system where the request originated will ship the retrieve request, if the object is not greater than 50 MB, to the target system using XCF messaging services.
- After encryption-enabled IBM TS1120 Tape Drive devices are used in an OAMplex environment and objects are written to tape volumes with the new EPI value recorded, it is expected that any OAM on a system where the full support is installed and enabled is eligible for processing requests using that volume. Therefore, encryption-enabled IBM TS1120 Tape Drive devices must be available to all instances of OAM where the full support is installed.

Open/Close/End-of-Volume (OCE)

The FILEE parameter list is now longer to accommodate the possible key encryption key (KEK) labels and their encoding mechanism. The version of the FILEE parameter list (TEPEVER) has been updated (to a 2) to reflect the longer FILEE parameter list. Before referencing the key label-related fields in the FILEE parameter list, ensure that either the version is set to 2 or the TEPMCRYP bit is "ON". When the TEPMCRYP bit is "ON", the key label-related fields contain pertinent data; otherwise, these fields contain binary zeroes.

Coexistence support is added at z/OS V1R4 and later to prevent an encrypted tape from being used on a lower level system. If an encrypted volume is detected during OPEN processing on a system that does not have all of the encryption support installed, abend code 613-84 is issued:

```
no software support for the media type or the recording technology
```

RMMplex

For the encryption-enabled IBM TS1120 Tape Drive support, DFSMSrmm coexistence support is provided at z/OS V1R4 and later, either through installation of the full support RMM

PTF without the Device Services enabling PTF or using installation of the PTFs for the toleration APAR OA16524.

This allows the coexisting system to tolerate tapes written by the encryption-enabled IBM TS1120 Tape Drive in EEFMT2 format and their associated key labels. If you have any systems sharing a control data set (CDS) with a system that installs the new RMM function (you do not need to have encryption in use), you must install the toleration PTFs for APAR OA16524, whether client/server is used or not. If client/server is used, the PTFs for APAR OA16523 (preconditioning) are also required. If you plan to fall back from any z/OS release RMM with encryption function to any z/OS release RMM without encryption function, toleration is required on that fallback system also.

RMM provides a toleration APAR OA16524 for an RMM Sysplex. With this APAR, systems in the Sysplex where the encryption code is not yet installed are enabled to keep (not to use) the key encryption key label fields (1 and 2). If the encrypted tape belongs to a system-managed tape library, the processing of this tape on a toleration system is limited: RMM's internal call to OAM fails with the message:

```
unsupported recording format
```

RMM provides a preconditioning APAR OA16523 and a toleration APAR OA16524 for a client/server RMMplex. The way that data is transmitted between a client system and a server system has changed. Transmission of CDS records is now release independent. A system where only the preconditioning APAR OA16523 is installed accepts lower and higher level systems. A system where the toleration APAR OA16524 is installed accepts preconditioning and higher level systems only, but also tolerates CDS records containing encryption key labels.

Important: For the installation on a client/server RMMplex, installing the preconditioning PTF on all systems of the RMMplex is *mandatory*. Then, install the toleration PTF on all systems, and *only* then, can you install the RMM Tape Encryption PTF. Preconditioning and Toleration APARs contain a ++HOLD(MULTSYS) text, which describes this dependency.

This process ensures that for a client/server RMMplex, you only need to update a single system at a time depending on your enterprise change control process.

4.8.3 DFSMSdss host-based encryption

If your DFSMShsm dump classes are currently defined to use host-based encryption (and possibly, host-based compression before encryption), we recommend that you remove the host-based encryption requests from any dump classes for which you plan to use hardware encryption. Over time, as you migrate your DFSMShsm dump classes to use hardware encryption, you might still have dump classes that are defined to use host-based encryption, while their associated Data Classes are defined to use hardware encryption. Here, DFSMSdss ignores requests for host-based encryption for tape volumes and, instead, uses hardware encryption. This processing allows you to complete the migration to hardware encryption without having to modify your DFSMSdss jobs.

If you no longer require host-based encryption for any of your tape volumes, remove the host-based encryption requests from all of your DFSMShsm dump classes. Thereafter, your jobs can write to a mixture of encrypting tape devices and non-encrypting tape devices without incurring informational messages. This setup allows you to encrypt tapes to send off-site for disaster recovery purposes, while retaining unencrypted tapes on-site.

4.8.4 Positioning TS1120 Tape Encryption and Encryption Facility for z/OS

The z/OS implementation of this product leverages z/OS security and availability features for key management.

The *Encryption Facility for z/OS* program product provides a complementary encryption solution for software encrypting non-TS1120 tape cartridge data. In a z/OS environment, the IBM Encryption Key Manager component for the Java platform can utilize the same hardware and software cryptographic features available on z/OS.

Figure 4-2 compares the areas that you should consider when implementing the complementary solutions of TS1120 Tape Encryption and the Encryption Facility for z/OS.

Depends on Customer Requirements ✓ Satisfies requirement	Business Partner Exchange	Archival	Backup/Restore
IBM Encryption Facility for z/OS (Business Partner Exchange)	✓ Compatibility ✓ Key Distribution/ manageability/ security ✓ Audit and access control	Performance ✓ Key retention Enterprise wide key mgmt. ✓ Audit and access control	✓ Key failover Performance ✓ Audit and access control
IBM Encryption Capable TS1120 Tape Drive (Archive and backup/restore)	Compatibility ✓ Key Distribution/ manageability/ security ✓ Audit and access control	✓ Performance ✓ Key retention Enterprise wide key management ✓ Audit and access control	✓ Key failover ✓ Performance ✓ Audit and access control
Can use the same mainframe centralized key management			

Figure 4-2 Encryption Facility for z/OS and TS1120 Tape Encryption solution comparison

Part 2

Implementing and operating the EKM

In this part, we provide detailed information about planning for, implementing, and operating the Encryption Key Manager (EKM).

5

Planning for EKM and its keystores

In this chapter, we discuss the planning for the Encryption Key Manager (EKM). EKM must be implemented in at least one of the Java Runtime Environments before you can encrypt any tape using System-Managed Encryption (SME) or Library-Managed Encryption (LME). Application-Managed Encryption (AME) does not require an EKM implementation. This planning can occur even before your hardware arrives. Chapter 6, “Implementing EKM” on page 159 completely describes the implementation of EKM.

EKM is part of the IBM Java environment and uses the IBM Java Security components for its cryptographic capabilities. The keystore used by EKM is defined as part of the Java Cryptography Extension (JCE) and an element of the Java Security components, which are, in turn, a part of the Java Runtime Environment.

EKM Release 1 or later is required for TS1120 encryption support. EKM Release 2 or later is required for Linear Tape-Open (LTO) 4 or LTO4 encryption support. Install the latest release available whether you are implementing for TS1120 and TS1130 encryption or for LTO4 encryption.

You must decide on what platform (or platforms) the EKM will run. In this chapter, we first provide an EKM planning quick-reference, then we describe requirements for each of the platforms where EKM can run. Then, we discuss various EKM and keystore implementation considerations.

5.1 EKM planning quick-reference

Table 5-1 compares the encryption characteristics of the TS1120 and TS1130 drives to the LTO4 drive. Table 5-2 on page 141 compares the Encryption Key Manager (EKM) software prerequisites for each platform. On Open Systems platforms, the TS1120 and the LTO4 are almost identical as to which encryption methods they support and the operating system software requirements. However, the LTO4 support might require later software levels, and fewer keystores are supported for LTO4.

Table 5-1 compares encryption characteristics of the TS1120 and TS1130 drives to the IBM LTO4 drive.

Table 5-1 Encryption implementation characteristics comparison

Description	TS1120 and TS1130 tape drive	LTO4 tape drive
Encryption standard	AES (256-bit)	AES (256-bit)
Encryption process for data	Symmetric AES (256)	Symmetric AES (256)
Encryption key model	Wrapped key	Direct key
Encryption type for data keys	Public-private key (Asymmetric)	None
Data keys used	Unique data key for each cartridge	Keyset: A list or range of data keys used, pregenerated in keystore
Data keys stored?	Wrapped (that is, encrypted) data keys (2) stored on cartridge, called EEDKs	Stored in keystore
Keystore	Contains private keys and certificates representing public keys. Preloaded in keystore	Contains data keys that have been pregenerated
Keystores types supported	► JCEKS (All platforms) ► JCE4758KS/JCECCAKS (z/OS) ► JCE4758RACFKS/ JCECCARACFKS (z/OS) ► JCERACFKS (z/OS) ► IBMi5OSKeyStore (i5) ► PCKS11IMPLKS (Open) ► DCMKS (Open)	► JCEKS (All platforms) ► JCECCAKS (z/OS) ► PCKS11IMPLKS (Open)

Table 5-2 compares the software prerequisites for each EKM platform.

Table 5-2 Encryption Key Manager platform software prerequisites

Description	TS1120 tape drive	LTO4 tape drive
Encryption Key Manager platform:	Requires EKM Release 1 or later	Requires EKM Release 2 or later
IBM z/OS	IBM SDK[a] for z/OS, J2TE[b], V1.4, 5655-I56 (at the SDK 1.4.2 SR6[c] level or later) IBM 31-bit SDK for z/OS, J2TE, V5.0, (z/OS 1.6 and later only), order 5655-N98 (at the SDK 1.5 SR3 level or later)	IBM SDK for z/OS, J2TE, V1.4, 5655-I56 (at the SDK 1.4.2 SR8 level or later) IBM 31-bit SDK for z/OS, J2TE, V5.0, (z/OS 1.6 and z/OS.e 1.6 and later only), order 5655-N98 (at the SDK 1.5 SR5 level or later)
IBM z/VM, z/VSE, and z/TPF	Not available	Not available
IBM AIX	Java 5 SR2 (32-bit or 64-bit) Java 1.4.2 SR5 (32-bit or 64-bit)	Java 5 SR2 (32-bit or 64-bit) Java 1.4.2 SR5 (32-bit or 64-bit)
IBM System i5	i5/OS V5.3, IBM Developer Kit for Java- Java Developer Kit 5.0 i5/OS V5.4, IBM Developer Kit for Java - J2SE™[d] 5.0 32-bit	i5/OS V5.3, IBM Developer Kit for Java- Java Developer Kit 5.0 i5/OS V5.4, IBM Developer Kit for Java - J2SE 5.0 32-bit
Linux on System z	J2SE 5.0 SR2 or J2SE 1.4.2 SR5	J2SE 5.0 SR2 or J2SE 1.4.2 SR8
Linux on other servers	J2SE SDK 5 or J2SE SDK 1.4.2	J2SE 5.0 SR5 or J2SE 1.4.2 SR8
Sun Solaris 8, 9, and 10	TPC-LE (5608-VC6) and IBM Runtime Environment for Solaris, J2TE, Version 5.0 SR2 (Solaris SPARC 64-bit)	TPC-LE (5608-VC6) and IBM Runtime Environment for Solaris, J2TE, Version 5.0 SR5 (Solaris SPARC 64-bit)
HP-UX 11.0, 11i v1 or v2	TPC-LE (5608-VC6) and one of the following Java Runtime Environments (JREs) HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0, SR2 for 64-bit Itanium64 HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0, SR2 for 64-bit PA-RISC	TPC-LE (5608-VC6) and one of the following JREs HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0, SR5 for 64-bit Itanium64 HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0, SR5 for 64-bit PA-RISC

Description	TS1120 tape drive	LTO4 tape drive
Windows Server 2000 or Windows 2003 Server	TPC-LE (5608-VC6) and one of the following JREs IBM 64-bit Runtime Environment for Windows on AMD64/EM64T architecture, J2TE, Version 5.0 IBM 32-bit Runtime Environment for Windows, J2TE, Version 5.0 IBM 64-bit SDK for Windows on Intel Itanium® architecture, J2TE, Version 1.4.2	TPC-LE (5608-VC6) and one of the following JREs IBM 64-bit Runtime Environment for Windows on AMD64/EM64T architecture, J2TE, Version 5.0, SR5 IBM 32-bit Runtime Environment for Windows, J2TE, Version 5.0, SR5 IBM 64-bit SDK for Windows on Intel Itanium architecture, J2TE, Version 1.4.2, SR8

a. Software Developers Kit (SDK)
b. Java 2 Technology Edition (J2TE)
c. Service Refresh (SR)
d. Java 2 Standard Edition (J2SE)

5.2 Ordering information and requirements

In this section, we list the supported EKM platforms and indicate the requirements for EKM on each of the supported operating system platforms. In this section, the level of software that must be installed is given for a Service Refresh (SR) that supports both TS1120 or TS1130 and LTO4. If you have to know the minimum SR level that will support just the TS1120 and TS1130, refer to the tables in 5.1, “EKM planning quick-reference” on page 140. These tables contain information from the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide,* GA76-0418.

5.2.1 EKM on z/OS or z/OS.e requirements

If the EKM is run on the z/OS system, one of the IBM SDKs for Java 2 in Table 5-3 must be installed. They are available from:

`http://www.ibm.com/servers/eserver/zseries/software/java`

Table 5-3 EKM minimum software requirements for z/OS and z/OS.e

IBM Software Developer’s Kit (SDK)	Model and PID number
IBM SDK for z/OS, Java 2 Technology Edition, V1.4.2 *SR8*	5655-I56 (at the SDK 1.4.2 *SR8* level or later)
IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V1.5.0 *SR5* (z/OS and z/OS.e 1.6 and later only)	5655-N98 (at the SDK 1.5 *SR5* level or later)

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 EKM in order to serve AES keys.

The preferred method to replace these files 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 have to copy them to the `lib/security` directory, for example:

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

Alternatively, you can download the unrestricted policy files from the following Web site:

`https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

After you log in, select the **Unrestricted JCE Policy files for SDK 1.4.2**, which works for both Java 1.4.2 and Java 5.0 SDKs. *Do not select the 1.4.1 version*, because these files are incompatible.

Secure symmetric key support on z/OS is through 5655-N98 (at the SDK 1.5 SR9 level or later).

As you plan for production deployment of the EKM on z/OS, note that there is a difference in the cryptographic capabilities between the SDK 1.4.2 SR8 and SDK 5.0 SR5 Java products. SDK 5.0 SR5 and later allow for stronger keys that can reside in host storage in an unencrypted fashion (that is, keys that are not encrypted by the IBM Integrated Cryptographic Service Facility (ICSF) host master key.

Another consideration is what platforms and SDK levels will be used by your partners reading the tapes written from your z/OS system. Their environments have to be understood to ensure that the cryptographic capabilities of the reader are compatible with the SDK level that you have chosen for the z/OS deployment.

Note: RACF APAR OA13030 is required on z/OS 1.4, 1.5, 1.6, and 1.7 for greater than 1024 modulus support. This APAR also 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 to 1.7, you need to verify that the PTF for this APAR has been applied.

5.2.2 EKM on z/VM, z/VSE, and z/TPF

EKM does not run on z/VM, z/VSE, or z/TPF, but EKM can be used with these operating systems *through the out-of-band tape controller connection to an EKM server* running on z/OS or another supported EKM platform.

5.2.3 EKM on IBM System i5 requirements

The EKM is supported on i5/OS V5.3 or later. If the EKM component is run on i5/OS, IBM Developer Kit for Java is required: Java Developer Kit 5.0 (5722-JV1).

If the EKM is run on the i5/OS system, then one of the following IBM SDKs for Java 2 in Table 5-4 on page 144 must be installed.

Table 5-4 EKM minimum software requirements for i5/OS

i5/OS	IBM software developer kit	Model and PID number
V5R3	IBM Developer Kit for Java - Java Developer Kit 5.0	5722-JV1 option 7 plus PTF SI25093 for 5722-SS1 option 3. 5722-AC3 is also required.
V5R4	IBM Developer Kit for Java - J2SE 5.0 32-bit	5722-JV1 option 8 plus SR3, and PTF SI25094 for 5722-SS1 option 3.

Note: Regardless of which IBM JDK™ 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 EKM in order to serve AES keys. For details about installing the unrestricted policy files, refer to the information about installing the IBM Software Developer Kit (i5/OS) in the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418.

5.2.4 EKM on AIX requirements

If the EKM is run on a System p server with AIX, the requirements is to have AIX V5.2 or later. Table 5-5 lists the options for IBM Java SDKs to update in order to include the latest version of the EKM.

Table 5-5 AIX EKM minimum software requirements

IBM Software Developer Kit	Type of update
Java 5 SR5 (32-bit) Java 5 SR5 (64-bit) Java 1.4.2 SR8 (32-bit) Java 1.4.2 SR8 (64-bit)	IBM AIX Developer Kit and Runtime, Java Technology Edition AIX Software Update Web
Download available from: `http://www.ibm.com/developerworks/java/jdk/aix/service.html`	

Updates to the AIX Java SDK can be obtained at the following Web site:

`http://www.ibm.com/developerworks/java/jdk/aix/index.html`

Note: Regardless of the version of IBM SDK that 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 that you can download from:

`https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

These files install the unrestricted policy files that EKM requires in order to serve AES keys. After you log in, select **Unrestricted JCE Policy files for SDK 1.4.2**, which works for both Java 1.4.2 and Java 5.0 SDKs. *Do not select the 1.4.1 version*, because these files are incompatible.

5.2.5 EKM on Linux requirements

If the EKM is run on a Linux system, one of the IBM SDKs for Java 2 (listed in Table 5-6) must be installed.

Table 5-6 Linux EKM minimum software requirements

Platform	IBM Software Developer Kit
64-bit AMD/Opteron/EM64T 32-bit xSeries (Intel -compatible) 32-bit pSeries 64-bit pSeries 31-bit zSeries (S/390®) 64-bit zSeries (S/390)	J2SE 5.0 SR2 or J2SE 1.4.2 SR5 for TS1120 drives J2SE 5.0 SR5 or J2SE 1.4.2 SR8 for LTO4 drives
Available at: http://www.ibm.com/developerworks/java/jdk/linux/download.html	

Note: Regardless of the version of the IBM SDK that you use, you must replace the `US_export_policy.jar` and `local_policy.jar` files in your directory `java_home/usr/java5/jre/lib/security` with new files that you can download from:

https://www.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk

These files install the unrestricted policy files that EKM requires in order to serve AES keys. After you log in, select **Unrestricted JCE Policy files for SDK 1.4.2**, which works for both Java 1.4.2 and Java 5.0 SDKs. *Do not select the 1.4.1 version*, because these files are incompatible.

Updates to the Linux Java SDK can be obtained at the following Web site:

http://www.ibm.com/developerworks/java/jdk/linux/download.html

5.2.6 EKM on Hewlett-Packard, Sun, and Windows requirements

To run the EKM component on Hewlett-Packard UNIX (HP-UX), Sun Solaris, or Windows, the IBM TotalStorage Productivity Center - Limited Edition (TPC-LE), licensed program product 5608-VC6, is required.

If the EKM is run on HP, Sun, or Windows platforms, you must install one of the Java Runtime Environments listed in Table 5-7.

Table 5-7 EKM minimum software requirements for HP, Sun, or Windows

Platform	Operating system	Runtime environment bundled with TPC[a]
HP	HP-UX 11.0, 11i v1 or v2	One of the following items: ► HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0 for 64-bit Itanium SR2 or later for TS1120, SR5 or later for LTO4 ► HP Runtime Environment for J2SE HP-UX platform, adapted by IBM for IBM Software, Version 5.0 for 64-bit PA-RISC SR2 or later for TS1120, SR5 or later for LTO4
Sun	Sun Solaris 8, 9, or 10	IBM Runtime Environment for Solaris, Java 2 Technology Edition, Version 5.0 SR2 or later for TS1120, SR5 or later for LTO4

Platform	Operating system	Runtime environment bundled with TPC[a]
System x	Windows 2000 Server or Windows Server 2003	IBM 64-bit Runtime Environment for Windows on AMD64/EM64T architecture, Java 2 Technology Edition, Version 5.0 Base for TS1120, SR5 or later for LTO4 IBM 32-bit Runtime Environment for Windows, Java 2 Technology Edition, Version 5.0 Base for TS1120, SR5 or later for LTO4 IBM 64-bit SDK for Windows on Intel Itanium architecture, Java 2 Technology Edition, Version 1.4.2 Base for TS1120, SR8 or later for LTO4

a. TPC: IBM TotalStorage Productivity Center - Limited Edition (TPC-LE) - licensed program product 5608-VC6

5.3 EKM and keystore considerations

We begin with questions for you to consider:

- On what platforms will you run EKMs? The EKM is currently supported on:
  - z/OS 1.4 and later
  - AIX 5.2 and later
  - i/5OS 5.2 and later
  - Linux (Red Hat Enterprise Linux 4 and SUSE Linux Enterprise Server 9)
  - HP-UX 11.0, 11i v1 and v2, or later
  - Sun Solaris 8, 9, and 10
  - Windows 2000 Server and Windows Server 2003

  You might want to run the EKM on more than one of these platforms. In all cases, you want EKM to be running on a highly available platform that is available any time you require access to the drives.

  If you have z/OS systems, we expect you will probably implement at least one instance of the EKM on the z/OS platform. If you also have Open Systems tape encryption requirements, you can use the z/OS EKM or you can decide that you want a separate and additional EKM implementation on one or more of your Open Systems platforms. Section 5.3.4, “Typical EKM implementations” on page 149 has more detail about various EKM platform considerations.
- What keystore deployment model will you employ? Options include:
  - For z/OS, possible keystore options are:
    - JCEKS (file-based)
    - JCE4758KS/JCECCAKS (ICSF secure hardware)
    - JCE4758RACFKS/JCECCARACFKS (RACF with secure hardware)
    - JCERACFKS (RACF/SAF)
  - For distributed Open Systems, possible keystore options are:
    - JCEKS (file-based)
    - PCKS11IMPLKS (PKCS11 hardware crypto)

– For i5, possible keystore options are:
  - JCEKS (file-based)
  - IBMi5OSKeyStore (i5 platform capabilities)
– For LTO4 encryption, only the JCEKS or the PCKS11IMPLKS keystore types are currently supported.

Table 5-8 indicates the supported keystores for drives. If you want to support both TS1120 and TS1130 encryption and LTO4 encryption from the same keystore, your choices are limited.

Table 5-8 Comparison of supported keystores

Keystore type	Platforms supported	TS1120, TS1130 (stores keypairs and certificates)	LTO4 (stores symmetric keys)	TS1120, TS1130 and LTO4	Symmetric key tools available
JCEKS	ALL	Yes	Yes	Yes	keytool
PKCS11Impl	Open	Yes	Yes	Yes	keytool
IBMi5OS	System i	Yes	No	No	No
JCE4758KS JCECCAKS	System z	Yes	Yes	Yes	Yes keytool (clear keys) hwkeytool (securekeys)[a]
JCERACFKS	System z	Yes	No	No	No
JCE4758RACFKS JCECCARACFKS	System z	Yes	No	No	No

a. Java 1.5 SR9 and later

- Do you want to use secure hardware cryptographic services?

  This consideration is driven by the regulations and requirements your business needs to meet. You can start simple, implementing a software JCEKS. If you later decide to move to a hardware solution, all you need to do is point your EKM at this new keystore. All of your other application setup stays the same. If you use this approach, remember that you must import the keys from the JCEKS to the new keystore to be able to decrypt previously encrypted data. This topic is discussed in 7.1, “Keystore and SAF Digital Certificates (keyrings)” on page 228.
- Do you want to use ICSF/RACF keyrings/keystores?

  Again, you may start simple with JECKS and move to ICSF/RACF keyrings/keystores. Chapter 8, “EKM operational considerations” on page 275 has an in-depth discussion about why you might choose this over another keystore implementation.
- Is your EKM behind a firewall?

  As part of your centralized key management strategy, the EKM that your platform needs to access might be behind a firewall. Our EKM was behind a firewall in our internal cross-site testing (Example 5-1 on page 148), and we solved this issue by working with our network support team to create exceptions in the firewall.

Example 5-1 Firewall exception process sample form

```
Requester's Name                    :Your Name
Requester's Email Address           :yourid@us.ibm.com
Requester's Manager Name            :Your Manager
Requested Site                      :Poughkeepsie
Application                         :IBM Tape Encryption
IP Address of Source                :9.11.214.32, 9.11.214.187,
                                     9.11.214.184, 9.11.214.190,
IP Address of Destination           :9.12.17.104
Protocol of Ports                   :TCPIP / 3801
Length of Access                    :YE06
Bus. Justification                  :The Tape Team is developing a new Key
   Serving Encryption Software Application that can be installed on any
   available host that supports Java. A significant design of this solution is the
   ability to support remote hosts running this application along with various
   hardware configurations.
Impact if not approved              :Unable to validate important simulations
   of supported encryption solutions and the concepts of centralized key
   management.
```

In our case, resolving this issue meant providing the source and destination IP addresses and protocol of the ports as seen in Example 5-1. If your solution requires dealing with your company's firewall, plan to obtain this type of access as part of your implementation. For z/OS solutions, this is where the use of Virtual IP Addressing (VIPA) can reduce the number of exceptions required, because one VIPA address can provide access to any number of EKM addresses behind it.

5.3.1 EKM configuration planning checklist

Make sure you:

- Know the recipients for the tapes to be written and to be read. For each recipient:
  - An associated X.509 certificate must exist when using TS1120.
  - A key or range of keys must be pregenerated within the keystore when using LTO4.

  Note: If you are only going to encrypt tapes for use within your own organization, you can start with only a single certificate in the keystore for TS1120 encryption and a single symmetric key in the keystore for LTO4 encryption.

- Know the tape drives that will be used.

  For each tape drive, you have to determine the drive serial number for input into the EKM drive table. However, if you use the EKM acceptunknown function, this step is handled automatically for you. Refer to Chapter 6, "Implementing EKM" on page 159 for details.

- Have existing keys and certificates that you plan to use.

  With this information, you can import and create keys and certificates into the keyring and keystore to be used.

- Determine the keystore type that you will employ:
  - The *keystore* holds the public-private keys and certificates used by the TS1120 encryption method or the pregenerated symmetric data keys used by the LTO4 encryption method. These keystores are used by the EKM in assisting the tape drives in generating, providing, protecting, and retrieving encryption keys.

- Depending on the keystore chosen, a keystore *might* be shareable between EKM servers, such as RACF, Sysplex, and so forth.
- If EKM servers are running on separate systems, you are likely to use separate keystores.

Note: For EKM servers that typically handle requests from the same set of tape drives, the information in the associated keystores *must* be kept the same.

This consistency is required so that no matter which EKM server is contacted, the necessary information is available for the EKM server to use to support the requests that the EKM server receives from the tape drives.

We do not expect that this requirement to keep your keystores in synch will require much time, because we expect only occasional changes to the keystores after your initial implementation. Changing encryption keys at least once each year is a good practice.

5.3.2 Best security practices for working with keys and certificates

Best practices are:

- *Do not* lose your keys and certificates.
- *Do not* leave your keys and certificates available for other users to see.
- Make sure to *back up your keys and certificates*

Important: Although IBM has services that can help you to recover data from a damaged tape, if the damaged tape is encrypted, what you receive from the recovery will still be encrypted data. So, if you lose your keys, you have lost your data.

5.3.3 Acting on the advice

Maintenance, backup, and restoration of key and certificate information depend on the keyring and keystore implementation that you use. One method is to:

1. Create copies of the keystores that the EKM will use.
2. Retain a PKCS12 format file for *each* key/certificate combination and store this file in a secure location, for example, on read-only media in a locked cabinet.
3. Retain a copy of the PKCS12 format and the keystore at your DR sites.

This method allows you to re-create keystores if absolutely necessary. Also, refer to the documentation for each keystore (keyring) management tool in Chapter 7, “Planning and managing your keys” on page 227 for more information.

5.3.4 Typical EKM implementations

You can run the EKM on a different platform from where your data is being encrypted. This section summarizes several typical scenarios.

Data transfer and EKM both on z/OS

Figure 5-1 is a simple illustration of encryption enablement, using TS1120 as an example. In this scenario, the client requests the encryption of data through the DFSMS Data Class attribute. The IBM TS1120 Tape Drive requests a key from the EKM through the tape control unit. Over the FICON channel, the EKM provides the key. The data is then encrypted on the IBM TS1120 Tape Drive.

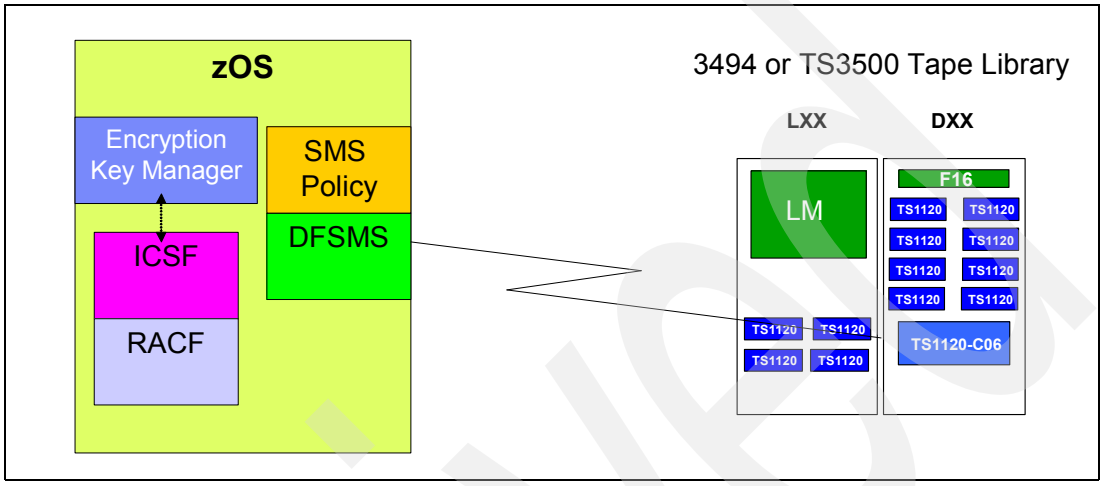


Figure 5-1 Single z/OS system key management

Data transfer on Open Systems with EKM on z/OS

Now, let us extend this concept to include the management on z/OS of the encryption keys associated with Open Systems tape data. In Figure 5-2 on page 151, the Open Systems server, System C, is using the library-managed approach for its tape data. When a key is required on the AIX system, the request travels from the tape drive to the tape library to the z/OS system over IP. The key manager and keystore are located on z/OS. This topology leverages the zSeries and System z9 infrastructure to manage the keys for an enterprise, including the ability to store the keys, using ICSF, in the unique zSeries or System z hardware-enabled keystore. This method provides a highly secure environment for the keystore and integration with RACF for tight control of the enterprise keystore, and allows the primary and secondary key managers to share the same keystore using z/OS Sysplex data sharing. The primary EKM and secondary EKM can also have different keystores that are kept synchronized using mirroring technology or administrative processes.

Note: No requirement exists to change the applications on the Open Systems.

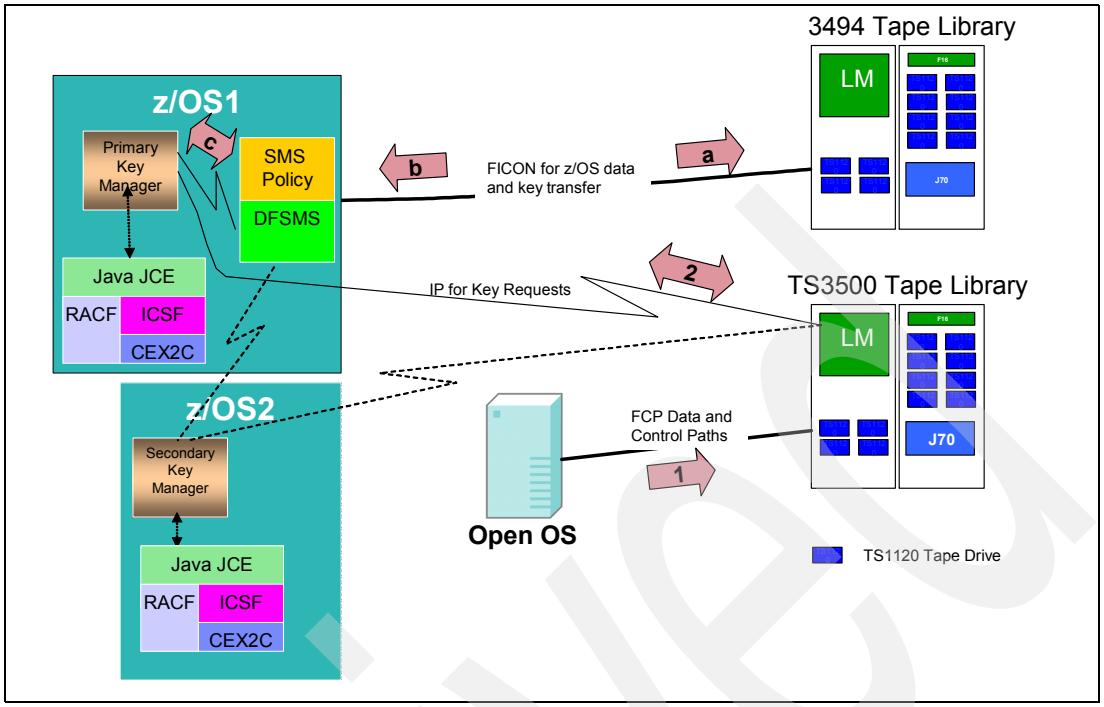


Figure 5-2 Example z/OS system central key management

Data transfer on Open Systems with EKM on AIX

The Open Systems environment supports the concept of a centralized key manager. In Figure 5-3, the Library-Managed Encryption methodology is used in combination with the keystore on AIX to support keys across multiple servers. Remember, no requirement exists to change the applications on the attached Open Systems servers. You might use this approach when key management for the enterprise resides on one of the supported Open Systems platforms. Again, this approach allows you to have a central keystore that is used by multiple platforms within your enterprise. In this case, the application on another Open Systems platform requests a cartridge mount on a library with encryption policy set to `ON` for the VOLSER range requested. The library then routes the drive key request to the EKM on AIX, which provides the key to the library. The key is then provided to the drive by the library, and the application on another Open Systems server begins to write.

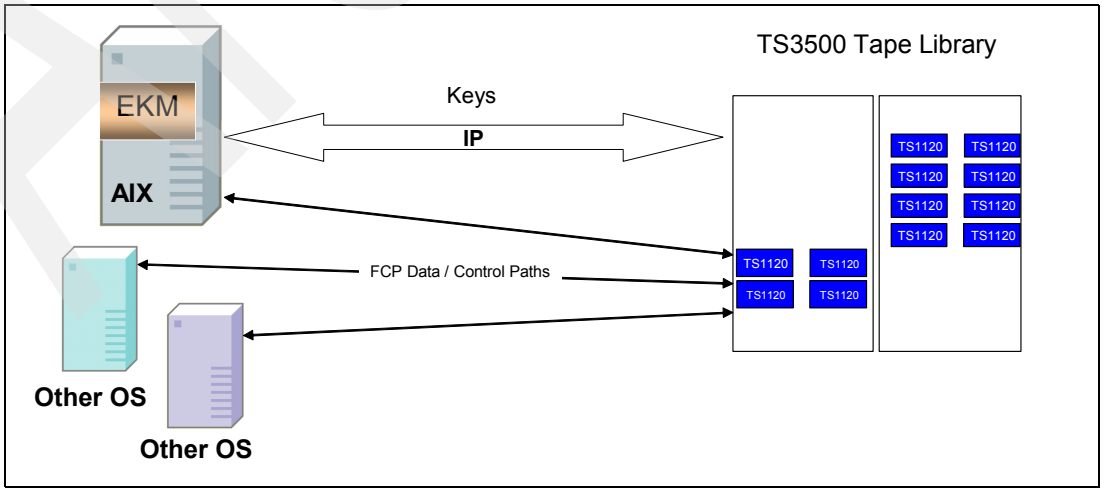


Figure 5-3 Example of Open Systems central key management with Library-Managed Encryption

Multiple-site EKM implementations

Figure 5-4 shows a multi-site tape data encryption environment where multiple platforms across the sites can all interoperate. Figure 5-4 is an example from the solution test environment used by the tape data encryption test team to validate the interoperability of all the supported combinations that are possible for the initial support that was released.

This setup included the need to access EKMs through a firewall as well as leveraging Virtual IP Addressing (VIPA) to add additional redundancy within a z/OS environment. By using shared common keys, it was possible to run jobs on any of these environments, writing a tape from an EKM on one platform and then reading that tape by pointing to any other EKM within this enterprise.

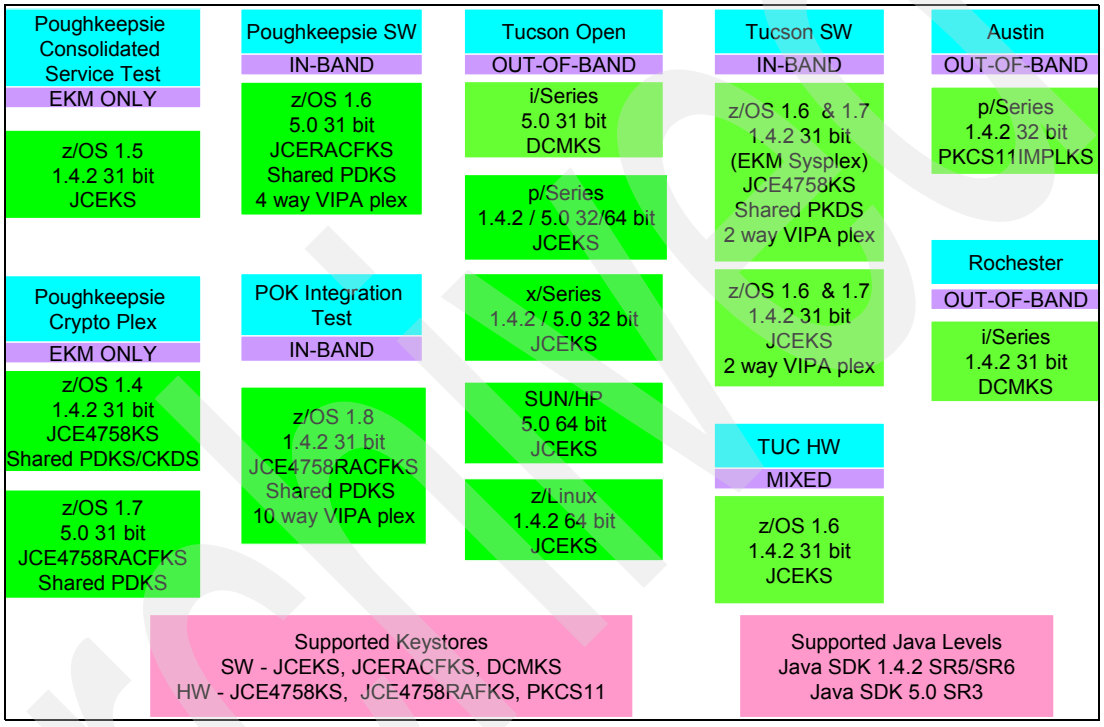


Figure 5-4 Sample multiple site tape data encryption environment

This setup is obviously an extreme example that facilitated testing various platforms and stress levels, but it gives you an idea of the unlimited options that are available and aspects of planning that you need to consider.

5.3.5 Multiple EKMs for redundancy

A number of solutions are available that you can use to provide redundancy. The certificates and keys stored in EKM's keystore are the point of control allowing a tape drive or library to decrypt the data on the tape. This approach makes the information in the keystore vital, because without it, the tape cannot be read. Therefore, although protecting this information so that others cannot obtain the private keys from the keystore is important, this information must also always be available to you so that you can read the tapes when necessary.

The following considerations are related to the type of keystore that you might select to meet your security needs, as well as your business needs, with regard to performance, backup, and archival.

EKM is designed to work with tape drives and libraries to allow redundancy, and thus high availability, so you can have more than one EKM servicing the same tape drives and libraries. Moreover, these EKMs do not need to be on the same systems as the tape drives and libraries. Refer to Figure 5-5. The only requirement is that the EKMs are available to the tape drives through TCP/IP connectivity. This approach allows you to have two EKMs that are mirror images of each other with built-in backup of the critical keystore information and a failover, if an EKM becomes unavailable.

When you configure your device (or system, library, or control unit, as appropriate), you can point it to multiple EKMs. If one EKM becomes unavailable for any reason, your device simply uses the alternate EKM. You also have the capability to keep the two EKMs synchronized. Taking advantage of this important function when needed is crucial, both for its inherent backup of critical data and also for its failover capability to avoid any outages in your tape operations.

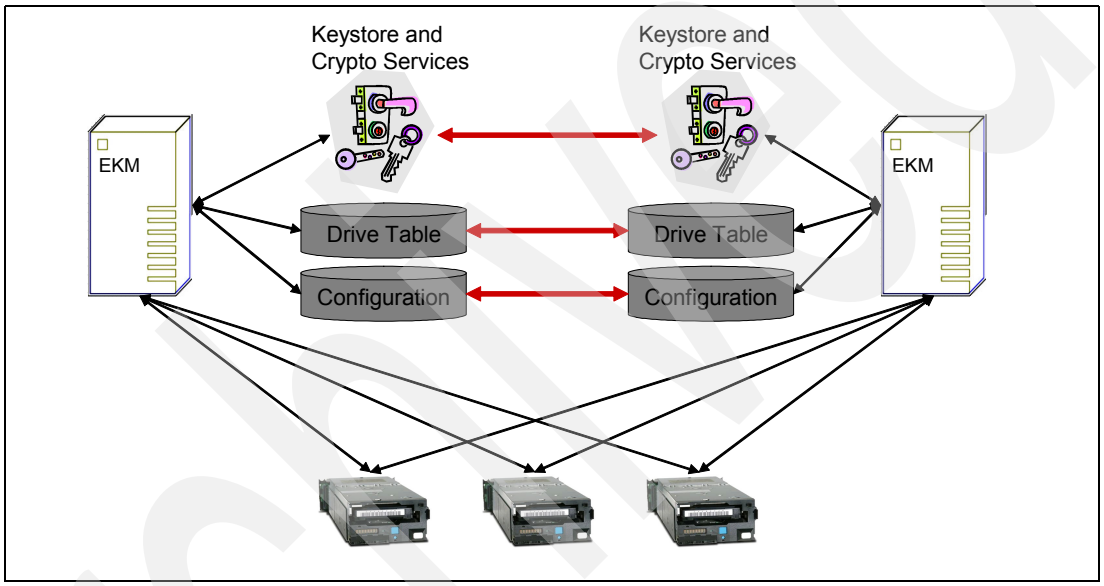


Figure 5-5 Keystore redundancy example

5.3.6 Using Virtual IP Addressing

The basic design of the Sysplex Distributor provides an advisory mechanism that checks the availability of applications, such as the EKM, running on different z/OS servers in the same Sysplex and then selects the best-suited EKM for a new connection request. The Sysplex Distributor bases its selections on real-time information from sources, such as Workload Manager (WLM) and quality of service (QoS) data from the Service Policy Agent. Sysplex Distributor also measures the responsiveness of target servers in accepting new TCP connection setup requests, favoring those servers that are more successfully accepting new requests. Internal workload balancing within the Sysplex ensures that a group or cluster of application server instances can maintain optimum performance by serving EKM requests simultaneously. High availability considerations suggest that at least two EKM instances have to exist, both providing the same service to request for keys. If one EKM instance fails, the other EKM instance carries on providing service. Multiple EKM instances minimize the number of requests affected by the failure of a single EKM instance. Thus, load balancing and availability are closely linked.

Virtual IP Addressing (VIPA) can be used to automatically maintain continuous availability in your Sysplex environment and add another layer of redundancy. See Figure 5-6 on page 154. We have configured an 8-way z/OS Sysplex where we have set up EKMs running on four of

the images in this plex. Hosts running tape data encryption can point to the VIPA address of this z/OS Sysplex and allow the Sysplex Distributor to route the request to an available EKM within the plex. This provides the advantage that if one of the EKMs is down because of an image IPL, the Sysplex distributor is aware of this situation and, acting as a primary EKM address, can still route traffic to the remaining images still available where an EKM is actively listening.

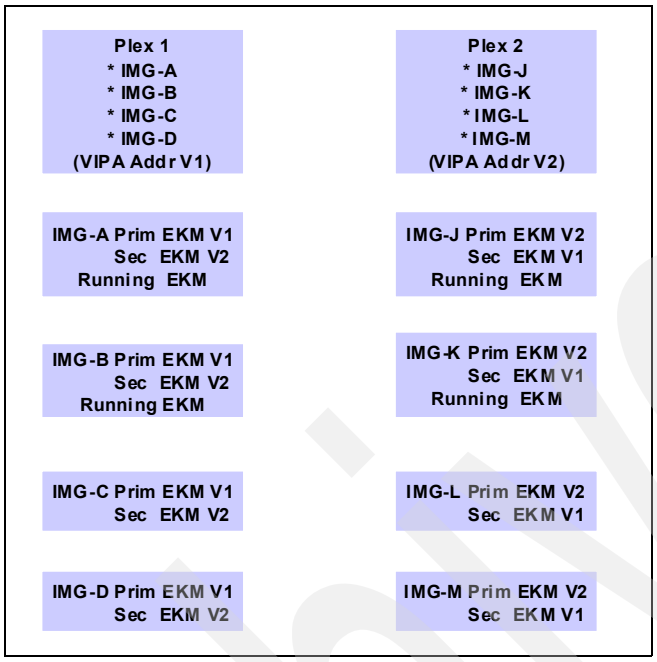


Figure 5-6 VIPA Sysplex Distributor

The Sysplex Distributor example in Figure 5-6 requires that all systems have access to a common keystore, or that if the keystores are separate, they have the same keys and EKM configuration setup.

You have to set up procedures to keep PLEX1 and PLEX2 EKMs in sync. On PLEX1, use SETIOS command to use the V1 address; on PLEX2 you SETIOS to use the V2 address. For large enterprises, it is best if PLEX1 and PLEX2 are in different data centers. If there are other Sysplexes or systems performing tape data encryption, they can all use V1 and V2 and have uninterrupted key management capability.

When running on PLEX1, the only time you ever need to go to PLEX2 for keys is if every EKM in PLEX1 is down. This is very rare. The benefit of this setup is after you get it up and running, you no longer need to worry about key manager access during planned and unplanned outages, especially outages completely unrelated to the key manager (for example, a planned window to POWER® ON RESET (POR) a CEC that houses the z/OS image on which the key manager runs).

Note: In an open systems environment, a common approach is to use network load balancers and similar network hardware to enable the ability to have more available EKM addresses than the tape libraries and virtual tape systems support.

5.3.7 Key Manager backup

Because of the critical nature of the keys in your keystore, *make sure to back up* this data so that you can recover it as necessary and be able to read the tapes that were encrypted using the certificate associated with that tape drive or library. There are many ways to back up this keystore information. Each keystore type has its own unique characteristics, but these general guidelines apply to all:

- 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, because it is impossible to decrypt it for recovery.
- If you are using crypto hardware, be sure to copy the PKDS.
- Maintain a primary and secondary EKM and keystore copy (for backup as well as failover redundancy).
- If you are using 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, because it is impossible to decrypt it for data recovery.
- Keep a copy of all certificates loaded into the keystore (usually a PKCS12 format file).

Important: Backups must not be encrypted. If the backup with the data you are recovering is encrypted, you do not have the keystore (that is damaged) with which to get the data back.

5.3.8 FIPS 140-2 certification

EKM does not provide cryptographic capabilities, and therefore, it does not require, nor is it allowed to obtain, FIPS 140-2 certification. However, EKM takes advantage of the cryptographic capabilities of the IBM JVM in the IBM Java Cryptographic Extension (IBMJCE) 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 EKM use the IBMJCEFIPS provider for all cryptographic functions.

In addition to setting the FIPS parameter in the Configuration Properties file, the following provider must be added to the java.security file in `$JAVA_HOME/lib/security/`:

```
security.provider.?=com.ibm.crypto.fips.provider.IBMJCEFIPS
```

You must add this provider before adding the IBMJCE provider. Renumber the providers accordingly.

Note: You must not use hardware-based keystore types with FIPS.

5.4 Other EKM considerations

This section summarizes additional EKM implementation and migration considerations.

5.4.1 EKM Release 1 to EKM Release 2 migration

If you already have EKM Release 1 installed, you must upgrade to EKM Release 2 in order to support LTO4 encryption. To upgrade:

1. After shutting down the EKM server, replace the Release 1 `IBMKeyManagementServer.jar` with the Release 2 version.
2. Add the required `Audit.metadata.file.name` property to the configuration file.
3. If you are planning to support LTO4 encryption:
   a. Generate and add the required symmetric keys to the keystore specified in the `config.keystore.file.name` property in the configuration file. This action assumes that the keystore that you are using will support symmetric keys.
   b. Add the *symmetricKeySet* property to the configuration file.
4. Restart the EKM.

5.4.2 Data exchange with business partners or different platforms

A common practice is to share tapes with other organizations for joint development, contracting services, or other purposes. To facilitate this, EKM can store two sets of wrapped encryption keys on the tape, allowing another organization to read that specific tape without you having to provide them any shared secret information or having to compromise the security of your certificates and keys. This process is done by adding the public part of the other organization's public-private certificate and keys to your EKM's keystore using a second alias (or key label).

When the tape is written, the encryption keys are stored on the tape in several places and protected by two sets of public-private keys, yours and the other organization's. The other organization is then able to use their EKM and their private key to unwrap the data key that allows them to read that specific tape. To reiterate, your EKM 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 EKM. This gives you the flexibility to make a specific tape readable by both your own organization and another organization. If you want to take advantage of this capability, you must add that other organization's certificate and public key to your keystore.

5.4.3 Disaster recovery considerations

If you plan to use a disaster recovery (DR) site, EKM provides a number of options to enable that site to read and write encrypted tapes:

- Create a duplicate EKM at the DR site with the same information as your local EKM (configuration file, tape drive table, and keystore). This EKM is then in place and capable of taking over for one of your existing production EKMs to read and write encrypted tapes.
- Create a backup copy of the three EKM data files to be able to recover as needed. If you create a current copy of the three data elements needed by EKM (configuration file, tape drive table, and keystore), you are able to start an EKM at any time to act as a duplicate at the DR site. 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.

Note: Remember that you must not use the EKM to encrypt the EKM data files, because you cannot decrypt them without a functioning EKM.

- When going to DR it is standard practice to set the EKM variable `drive.acceptUnknownDrives` in the configuration file to true. Refer to Chapter 6, “Implementing EKM” on page 159 for more information.

5.4.4 i5/OS disaster recovery considerations

The following considerations apply for i5/OS disaster recovery:

- The i5/OS support requires the EKM server to run on a different partition or system other than where the encrypted save is performed. Failure to do so can result in data loss.
- Prior to recovering encrypted data, the EKM must be running or recovered on another system.
- Maintaining primary and secondary EKM servers is desired for maximum availability of encrypted backup and recovery.
- The EKM 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), you must keep a copy of the password in a secure location. The keystore password must be available to recover the EKM.
- Encrypted save or archive operations must not be performed on the partition or system where the EKM server is running. If data is encrypted on the system where EKM is running, EKM cannot be recovered without the availability of a secondary EKM server.

5.4.5 EKM performance considerations

With System-Managed or Library-Managed Encryption enabled, when writing from loadpoint, the access time to the first write from the beginning of tape increases because of the time needed to retrieve, read, and write the encryption keys. In z/OS, this added time (which is the time between the mount message and the IEC705I “Tape On” message) is detected in OPEN processing.

If your EKM is on a z/OS platform, insure EKM has a Workload Manager (WLM) job priority similar to other system services, such as VTAM® or TCP/IP. You do not want situations where the EKM has to wait for CP cycles to return keys to access and return keystore information, because this wait can delay processing across your enterprise.

Using Virtual IP Addressing (VIPA) in your z/OS setup can contribute to both better performance and redundancy when running with a z/OS-based EKM. Refer to 5.3.6, “Using Virtual IP Addressing” on page 153 for more information about this topic.

With z/OS, you might also see a longer delay when using in-band encryption if your primary key manager is unavailable. In this case, the I/O Supervisor (IOS) Proxy Retry Logic first attempts to communicate with the primary key manager. The IOS proxy interface might retry several times before switching over to the secondary key manager. While the retries are occurring, the job can appear to have stopped.

Before cancelling a job, ensure that enough time is allowed for the retry attempts that can be occurring on the primary key manager and also the secondary key manager. Typically, each attempt can take approximately three minutes with two retry attempts on the primary key manager before attempting to connect to the secondary key manager.

Similar logic is then in place with the secondary key manager. After the proxy interface has switched to the secondary key manager, it always attempts to communicate with the primary key manager on subsequent communications; however, in this case only one (shortened) attempt is made to communicate with the primary key manager before going back to the secondary key manager. If the IOS proxy interface cannot communicate with the primary key manager, even though the job might have been successful, message IOS627E is issued in the joblog and in the system log alerting you to a potential problem with the primary key manager.

6

Implementing EKM

In this chapter, we describe the steps to take to install the Encryption Key Manager on different host platforms.

6.1 Implementing the EKM in z/OS

The Encryption Key Manager (EKM) is a common platform application written in Java that runs under the Java Virtual Machine (JVM). The EKM can also reside outside of the z/OS environment. If the EKM resides within z/OS, it is part of the UNIX System Services, which is part of the Open MVS (OMVS) address space.

The EKM interfaces with the associated keystores using application programming interfaces (APIs). Secure TCP/IP connections are used to communicate with the tape drives (in-band or out-of-band).

The following keystores are possible for z/OS:

- JCEKS (file-based)
- JCE4758KS/JECCAAKS (ICSF secure hardware)
- JCE478RACKFKS/JCECCARACFKS (RACF with secure hardware)
- JCERACFKS (RACF/SAF)

Software-based keystores are JCEKS and JCERACFKS. The hardware-based keystores work with the Integrated Cryptographic Service Facility (ICSF). If the EKM resides outside of the z/OS environment, the JCEKS software-based keystores or the PKCS11IMPLKS hardware-based keystore will be used.

We recommend that you use more than one EKM because of redundancy. EKMs can either share keystores or use separate keystores. For example, it is possible to have a primary EKM running on z/OS and a secondary EKM that resides on a System p server under AIX. A connection between them is only necessary for synchronization purposes. For more detailed information about EKMs, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418.

6.1.1 z/OS UNIX System Services

The System Control Program (SCP) of z/OS contains address spaces for general Multiple Virtual Storage (MVS) and address spaces for open MVS, which is also called UNIX System Services. When a Time Sharing Option (TSO) user logs on to the system and tries to use UNIX System Services, the appropriated UNIX System Services address space will be created.

In-band tape data encryption requires that the I/O Supervisor (IOS) address space has security permissions for a UNIX System Service segment. Security permissions can be obtained in RACF by issuing the following command:

```
ADDUSER IOSSAS OMVS(UID(0) HOME('/'))
```

If a UNIX System Services segment is unavailable at the time of tape data encryption, the following message appears:

```
IOS628E ENCRYPTION ON DEVICE dddd HAS FAILED DUE TO OMVS SEGMENT FAILURE
```

Note: The UID does not need to be 0, but this user ID does need an OMVS segment. Changes to the IOSAS user ID require an IPL to be recognized. IOSAS is only used by the I/O Supervisor (IOS) component and must be defined as protected, so giving it UID(0) works fine.

It is assumed that the UNIX System Services address space is already present. Refer to *UNIX System Services z/OS Version 1 Release 7 Implementation,* SG24-7035, for more

information about how to install and tailor UNIX System Services in z/OS.

To ensure that an UNIX System Services address space is up and running, use the **/D A,L** MVS command as shown in Figure 6-1.

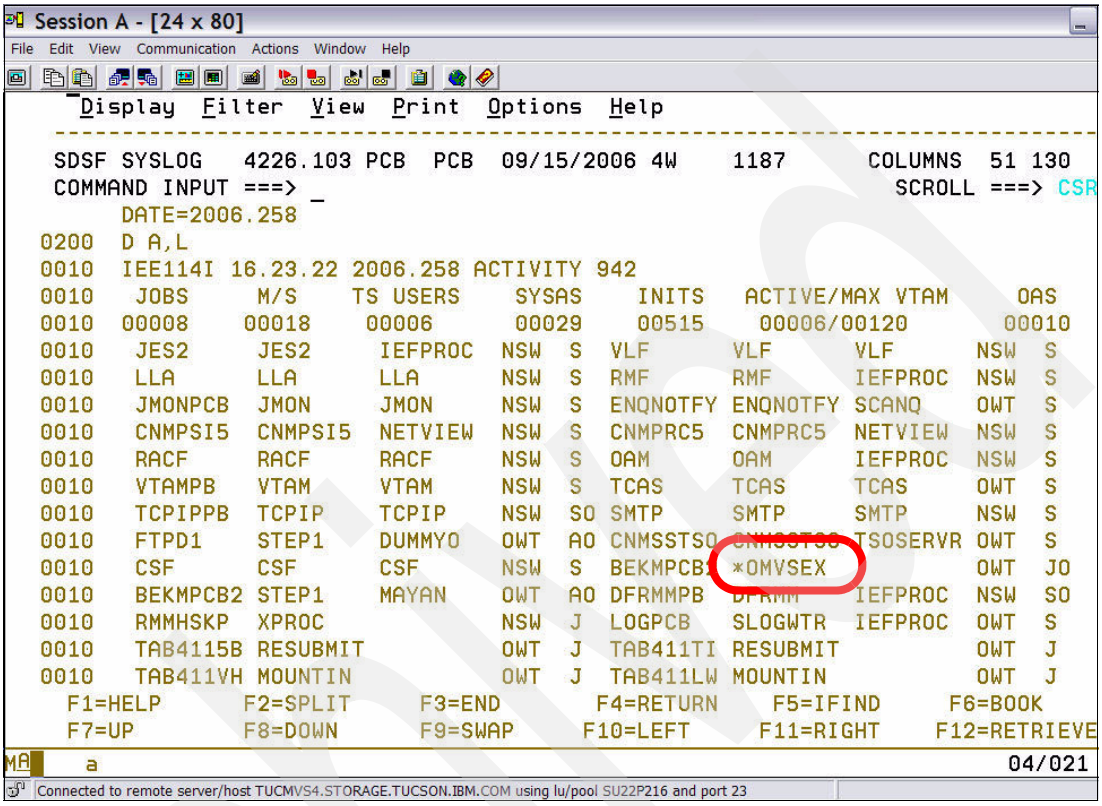


Figure 6-1 The OMVSEX address space is present

If the UNIX System Services address space is present, the OMVSEX for OMVS is shown.

6.1.2 Installing the EKM in z/OS

The EKM is automatically installed as part of the IBM Java Developer Kit (JDK).

To download the latest version of the IBM EKM, go to either:

- http://www.ibm.com/servers/storage/support/tape/ts1120/downloading.html
- http://www-947.ibm.com/systems/support/supportsite.wss/brandmain?brandind=5000034

You may download the IBM 31-bit Software Developer Kit (SDK) for z/OS Java 2 Technology Edition from the following Web site:

http://www.ibm.com/servers/eserver/zseries/software/java/j5pcont31.html

Note: Before you can download the latest version of the IBM SDK, you must register your company or yourself to give IBM statistics for a comparative look at which SDKs are being downloaded.

In ISPF or TSO, use either of the following steps:

- Select Option 6, select **Commands**, and type `OMVS`, and then press Enter.
- Run the **`telnet/rlogin, ssh`** command into an OMVS session.

Use a File Transfer Protocol (FTP) and copy the file into file system. Then, extract the file into /usr/lpp/java by using the following command:

```
pax -rf UK17593.PAX.Z
```

To verify that the correct version of the EKM was installed, use the following command at the OMVS command prompt:

```
/u/st6t25c>java -version
```

The information shown in Example 6-1 appears.

Example 6-1 Output from java -version

```
JVMHP030: Unable to switch to IFA processor - issue "extattr +a libhpi.so"
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
cm142-20060921 (JIT enabled: jitc))
```

The following error has to do with the way that we executed the **pax** command:

```
JVMHP030: Unable to switch to IFA processor - issue "extattr +a libhpi.so"
```

The command did not keep all the file attributes upon inflating the JDK pax file. This error means that the JDK does not have authority to run on the zAAP processor. To fix the problem, issue:

```
extattr +a $JAVA_HOME/bin/libhpi.so
```

Issuing the **java -version** command gives you output similar to Example 6-2.

Example 6-2 Output from java version with fixed libhpi.so attributes

```
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
cm142-20060921 (JIT enabled: jitc))
```

Download and install the latest EKM server code. Obtain the EKM code and documentation from:

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

After downloading the new EKM code, place it in the $JAVA_HOME/lib/ext folder. If your environment variables are set up correctly and the EKM program code is in the working directory, simply enter the following command to copy it:

```
cp IBMKeymanagerServer.jar $JAVA_HOME/lib/ext
```

At this point, the installation of the EKM for z/OS is already complete.

Note: You cannot start the EKM without an associated keystore.

The *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418, provides additional information.

To perform the installation steps necessary to use EKM, special authorization rights are required; for example, the RACDCERT command must be enabled for the OMVS segment. In

the following sections, we create hardware-related keystores as well as software-based keystores. In Chapter 8, “EKM operational considerations” on page 275, you can read about types of keystores and their usage. In the following sections, we create a hardware keystore.

EKM does not provide cryptographic capabilities and therefore it does not require and is not allowed to obtain FIPS 140-2 certification. However, EKM 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 EKM use the IBMJCEFIPS provider for all cryptographic functions.

In addition to setting the FIPS parameter in the Configuration properties file, the following provider must be added to the `java.security` file:

```
$JAVA_HOME/lib/security/security.provider.?=com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Add this line before the IBMJCE provider. Renumber the providers accordingly.

Note: We do not recommend that you use hardware-based keystore types with FIPS.

6.1.3 Security products involved: RACF, Top Secret, and ACF2

In-band tape data encryption requires that the I/O Supervisor (IOS) address space has security permissions for the UNIX System Services segment.

Note: The user ID (UID) in the following command examples does not have to be 0 (zero), but it does require an OMVS segment. Changes to the IOSAS user ID require an IPL to be recognized. IOSAS is only used by the IOS component and must be defined as protected, so giving it UID(0) works fine.

RACF

Security permissions can be obtained from RACF by issuing the following command:

```
ADDUSER IOSAS OMVS(UID(0) HOME('/'))
```

Top Secret

Top Secret is a security product provided by Computer Associates. Security permissions can be obtained from *Top Secret* by issuing the following command:

```
TSS ADD(IOSAS) UID(0) HOME(’/’)
```

Refer to the appropriate documentation for eTrust CA-Top Secret Security for z/OS.

ACF2

ACF2 is a security product provided by Computer Associates. Security permissions can be obtained from ACF2 by issuing the following command:

```
INSERT IOSAS NAME(ID IOSAS) UID(0) HOME
```

Refer to the appropriate documentation for eTrust CA-ACF2 Security for z/OS.

6.1.4 Create a JCE4758RACFKS for EKM

In this section, we create a JCE4758RACFKS. We use it later as the keystore for EKM initialization. This is the most secure of the keystores. Access to the keyring is protected by RACF administration, and private key material is stored in a PKDS that is protected by cryptographic hardware.

Chapter 7, “Planning and managing your keys” on page 227 discusses this keystore type.

The RACF commands in Example 6-3 define the FACILITY classes necessary to use the RACDCERT commands to create and work with keyrings and certificates.

Example 6-3 Defining irr.digtcert facility classes

```
RDEFINE  FACILITY  IRR.DIGTCERT.ADD     UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.ADDRING   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.DELRING   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.LISTRING   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.CONNECT   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.REMOVE    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.LIST     UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.ALTER    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.DELETE    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.GENCERT   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.GENREQ    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.EXPORT    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.EXPORTKEY  UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.MAP     UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.ALTMAP    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.DELMAP    UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.LISTMAP   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.ROLLOVER   UACC(NONE)
RDEFINE  FACILITY  IRR.DIGTCERT.REKEY    UACC(NONE)
```

The commands in Example 6-4 give *user* CONTROL access to all the FACILITY classes that deal with the RACDCERT command. The CONTROL access gives the user full access to administer everything that concerns the RACDCERT command.

To develop a more restrictive security policy, refer to the description of the RACDCERT command in 6.1.7, “Additional definitions of hardware keystores for z/OS” on page 174.

Example 6-4 PERMIT commands

```
PERMIT   IRR.DIGTCERT.ADD    CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.ALTER   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.DELETE   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.LIST    CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.ADDRING  CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.DELRING  CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.LISTRING  CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.CONNECT  CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.REMOVE   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.MAP    CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.LISTMAP  CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.ALTMAP   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.DELMAP   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.REKEY   CLASS(FACILITY)  ID(user)  ACCESS(CONTROL)
PERMIT   IRR.DIGTCERT.ROLLOVER  CLASS(FACILITY)  ID(user) ACCESS(CONTROL)
```

Example 6-5 shows how to export a certificate reply and how to import the certificate response back into a keyring. RACDCERT commands used in Example 6-5 perform the following actions:

- Creates a new self-signed certificate authority (CA) with a key size of 2048, private key information created with crypto-hardware, and that is stored in the active PKDS. The label of this CA is `CAexample`.
- Creates two personal certificates with labels of `ITSOCert1` and `ITSOCert2`, size of `2048`, signed with our CA, with private keymaterial created with crypto-hardware, and stored in the active PKDS.
- Imports a PKCS12 certificate from the data set `johann.private.key1` and store its private keymaterial in a PKDS. The PCICC keyword cannot be used, because we are only importing keys and not generating them.
- Creates a new keyring `ITSOring`.
- Connects the CA to the ring and the three personal certificates.

If a third-party certificate verification and response are required, refer to:

- 7.5, “JCE4758RACFKS and JCECCARACFKS” on page 248
- http://publibz.boulder.ibm.com/epubs/pdf/ichza460.pdf

Example 6-5 RACDCERT commands to create and import certificates and add them to a keyring

```
RACDCERT  CERTAUTH  GENCERT SUBJECTSDN(CN('ITSOEX')O('IBM')C('US')) PCICC
WITHLABEL('CAexample') SIZE(2048)

RACDCERT GENCERT SUBJECTSDN(CN('ITSO')  O('IBM') C('US')) PCICC
WITHLABEL('ITSOCert1') SIZE(2048) SIGNWITH(CERTAUTH LABEL('CAexample'))

RACDCERT GENCERT SUBJECTSDN(CN('ITSO')  O('IBM') C('US')) PCICC
WITHLABEL('ITSOCert2') SIZE(2048)  SIGNWITH(CERTAUTH LABEL('CAexample'))

RACDCERT ADD('johann.private.key1') TRUST
WITHLABEL('ITSOImport')password('password') ICSF

RACDCERT ADDRING(ITSOring)
RACDCERT CONNECT(CERTAUTH  LABEL('CAexample')  RING(ITSOring))

RACDCERTCONNECT(LABEL('ITSOCert1')RING(ITSOring)USAGE(PERSONAL)DEFAULT)
RACDCERTCONNECT(LABEL('ITSOCert2')RING(ITSOring)USAGE(PERSONAL))
RACDCERT CONNECT(LABEL('ITSOImport') RING(ITSOring) USAGE(PERSONAL))
```

These commands associate the keyring and personal certificates with the user executing them. To associate them with another user, append the ID(*user)* keyword after the RACDCERT command.

Tip: Be careful when cutting and pasting commands into a TSO session. When we tested cutting and pasting commands into a TSO session, the single quotation marks were stripped off.

6.1.5 Setting up the EKM environment

For this example, we assume that the user ID `EKM` is going to run on EKMSERV and that the UNIX System Services home directory for this user is /u/ekmserv. To create this user, issue the following command:

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

This command creates the user ID EKMSERV, sets up its home directory to /u/ekmserv, sets the default shell to /bin/sh, and associates it with the next available UID. If a more strict security policy is in effect, the RACF administrator must replace the AUTOUID with UID(*UserIDNumber*). Avoid using UID 0.

In addition, this user will need access to the IRR.DIGTCERT.* facility classes as described in Example 6-4 on page 164 in section 6.1.4, “Create a JCE4758RACFKS for EKM” on page 164.

Before continuing, download and save the following items to `/u/ekmserv/`:

- IBM EKM application
- IBM EKM sample configuration file
- JZOSEKM files for z/OS batch

Download them from:

`http://www-1.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc=en_US&cs=utf-8&cc=us&lang=en`

We first have to make sure that the `ekmserv` ID has an acceptable profile file. In `/u/ekmserv`, edit the `.profile` file so it looks like the sample profile shown in Example 6-6 on page 167.

In the example, note the following process:

1. The `stty quit ^V` line must be included in a profile when logging on to OMVS using Secure Shell (SSH). The `export PS1` line is setting up this user’s command line. In this case, we are setting up the command line to always list the current working directory. This setting is useful to have when navigating around an OMVS file system.
2. After that, we set up the `JAVA_HOME` environment variable. This variable has to point to the Java home on your system. Next, we add `JAVA_HOME/bin` to our path in addition to other useful directories. Notice that we are careful to keep our original path here by appending it to our new PATH.
3. We create an environmental shorthand. When using OMVS as the shell, and not logging on using Telnet, rlogin, or SSH, we have a limited length command line. The arguments to start EKM and perform Java hwkeytool commands might be longer than the command line; by setting some of these long parameters as environment variables, we can save space on our limited command line.

Example 6-6 Sample .profile

```
stty quit ^V
  export PS1='$PWD>';
  export JAVA_HOME=/u/java/J1.4
export PATH=.:${JAVA_HOME}/bin:/usr/sbin:$PATH
export DJ='-J-Djava.protocol.handler.pkgs=com.ibm.crypto.provider -v'
export DH='-J-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider -v'
export DT='-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider'
export JS='-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider'
alias kt="keytool -debug -list -storetype JCERACFKS -keystore"
alias hw="hwkeytool -debug -list -storetype JCE4758RACFKS -keystore"
export keyFile=KeyManagerConfig.properties
export EKM=com.ibm.keymanager.KMSAdminCmd
```

After setting up the profile, we can reread it with either of the following methods:

- Type the following command:

  ```
  . ./.profile
  ```

- Log out and log in again, which executes the .profile again.

Now, we must copy the updated EKM. This command copies the EKM program code into `JAVA_HOME/lib/ext`. The JARs in the `lib/ext` directory are all loaded into the JVM's classpath. A simple listing of that directory reveals security provider JAR files among other things:

```
cp IBMKeyManagementServer.jar $JAVA_HOME/lib/ext
```

The JZOS EKM files have to be extracted. Enter the command:

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

This command extracts the files:

- PROCLIB.EKM2ENV
- README
- jzosekm.jar
- PROCLIB.EKM2

The file `jzosekm.jar` contains Java wrapper code for the EKM. To interact with the EKM using write to operator (WTO), we must register a callback using APIs from the JZOS toolkit. The program code contained in this JAR does that for us. To ensure that this code is in our classpath, we copy it to `lib/ext`:

```
cp jzosekm.jar $JAVA_HOME/lib/ext
```

The EKM config file KeyManagerConfig.properties is now edited with our information. In Example 6-7 on page 168, we have turned on extra tracing and debugging information.

Example 6-7 Sample EKM config

```
TransportListener.ssl.port = 4430
TransportListener.tcp.port = 38010
fips = Off
Admin.ssl.keystore.name = safkeyring://ST6T25B/ITSOring
config.keystore.provider = IBMJCE4758
Admin.ssl.truststore.password = passphrase
TransportListener.ssl.clientauthentication = 0
config.keystore.password = password
TransportListener.ssl.ciphersuites = JSSE_ALL
Audit.handler.file.size = 500
zOSCompatibility = true
drive.acceptUnknownDrives = true
TransportListener.ssl.truststore.name = safkeyring://ST6T25B/ITSOring
Audit.handler.file.directory = keymanager/audit
TransportListener.ssl.protocols = SSL_TLS
debug.output = simple_file
TransportListener.ssl.truststore.type = jce4758racfks
config.keystore.file = safkeyring://ST6T25B/ITSOring
TransportListener.ssl.keystore.name = safkeyring://ST6T25B/ITSOring
TransportListener.ssl.keystore.password = passphrase
TransportListener.ssl.truststore.password = passphrase
Audit.event.outcome.do = success,failure
Audit.eventQueue.max = 0
debug.output.file = keymanager/debug/ekmdebug.jce
Audit.handler.file.name = ekm.audit.log
TransportListener.ssl.keystore.type = jce4758racfks
config.keystore.type = jce4758racfks
requireHardwareProtectionForSymmetricKeys = true
Audit.event.types.backup = authentication,authorization,data
synchronization,runtime,audit management,authorization terminate,configuration
management,resource management,none
drive.default.alias2 = itsocert1
drive.default.alias1 = itsocert2
Audit.event.outcome = success,failure
debug = all
Audit.event.types = all
Admin.ssl.truststore.name = safkeyring://ST6T25B/ITSOring
config.drivetable.file.url = FILE:////keymanager/drivetab
Admin.ssl.keystore.password = passphrase
```

At this point, we must create several directories. Example 6-8 on page 169 shows the commands and directories that we have to create. These commands and directories are relative to the path where we are starting the EKM server. In this case, they are relative to /u/ekmserv/.

The debug.output.file option points to a file. You will get java.io.permission errors if the assumption is made that it points to a directory. If the file does not exist, EKM creates it, but the file cannot point to a directory.

Example 6-8 Create directories

```
mkdir keymanager
mkdir keymanager/debug/
mkdir keymanager/audit
```

After setting this up, we can verify that our EKM can be started and load keys from our keyring by issuing the following command (use this command if you use the .profile from Example 6-6 on page 167):

```
java $DT $EKM $keyFile
```

The expanded command is:

```
java -Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider
com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties
```

If this command is too long, refer to “MVS Open Edition tips” on page 570.

Tip: To verify that the EKM server has sufficient authority, the following hwkeytool command can be issued from OMVS:

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

The provider list in the java.security file located in the path $JAVA_HOME/lib/security/java.security must contain the following providers:

```
security.provider.1=com.ibm.jsse.IBMJSSEProvider
security.provider.2=com.ibm.crypto.hdwrCCA.provider.IBMJCE48758
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.security.cert.IBMCertPath
```

6.1.6 Starting EKM

You can start the EKM by using a procedure or through commands. We discuss both options in the following sections.

Starting EKM with JZOS as a started task

A z/OS *started task* is a procedure that 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 it can be associated with a functional subsystem.

We suggest that the EKM run as a started procedure on z/OS using the JZOS batch launcher, which is shipped as part of the z/OS Java product. Refer to “EKM and JZOS” on page 567 for more information.

To define the EKM as a started procedure, update the started class table with the z/OS user ID of the EKM. Previously in this section, we created EKMSERV as the user ID to be used with the EKM and the group associated with that started procedure will be SYS1. Details of RACF processing and the definition of started procedures are discussed in the *z/OS Security Server RACF Security Administrator’s Guide*, SA22-7683.

To set up the STARTED class, enter the commands in Example 6-9 on page 170.

Example 6-9 Define EKM as a started task

```
SETROPTS  GENERIC(STARTED)
RDEFINE  STARTED  EKM*.*  STDATA(USER(EKMSERV)  GROUP(STCGROUP)  TRACE(YES))
SETROPTS  CLASSACT(STARTED)  SETROPTS  RACLIST(STARTED)
```

The JZOSEKMFiles.pax.Z file downloaded in the beginning of this section consists of jzosekm.jar, sample JCL, the STDENV file for the sample JCL, and a EKMConsoleWrapper. Use the readme file that explains where each file must be located and the installation-specific customization that might be required.

To extract the contents of the download file, issue the following command:

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

Place the jzosekm.jar file in the $JAVA_HOME/J1.4/lib/ext path.

The EKM can create audit records, which wrap the log to three files. When the last file is full, the first file is rewritten. On z/OS, you need to allocate file system space for the EKM audit logs, and possibly, the EKM debug log.

You may choose to allocate a file system specifically for use by the EKM for audit and debug file storage. Assume 500 cylinders of space to allocate to the EKM's audit and debug log file system until you have observed, based on tape and EKM activity, how quickly the audit logs wrap. The file system must not be shared by the EKM instances running in a Sysplex environment, but the files must be private to each EKM instance. This setup prevents any possible interleaving of audit or debug logs between EKM instances.

Mount the ekmlogs filesystem and create a directory for each system on which EKM will run. For example, the two file systems created can be ekmlogs with JA0 and JB0 for two system names of two images within a Sysplex:

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

Create a new partitioned data set (PDS) to contain the STDENV environment variables. In this example, a partitioned data set was allocated with the name EKMSERV.ENCRYPT.CONFIG that has the following attributes:

Organization	PO
Record format	FB
Record length	80
Block size	6160
First extent cylinders	3
Secondary cylinders	1

Create a member in EKMSERV.ENCRYPT.CONFIG named EKM2ENV. Create or Edit the shell script contents as shown in Example 6-10.

Example 6-10 EKM environment script

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

#Set  these  variables  to  the  installation  unique  values
#  EKM_HOME  =  directory  from where  EKM  runs
#  JAVA_HOME  =  directory  where  Java  is  mounted
```

```
#Update to point to your EKM Home directory
export  EKM_HOME="/u/ekmserv"
#Update to point to your JDK
export  JAVA_HOME="/u/java/J1.4"
export  PATH=/bin:"${JAVA_HOME}"/bin:

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

export  LIBPATH="$LIBPATH":

#  Customize  your  CLASSPATH  here
CLASSPATH=${JAVA_HOME}/lib
CLASSPATH=$CLASSPATH:"${EKM_HOME}"

export  CLASSPATH="$CLASSPATH":

#  Set  JZOS  specific  options
#Update  with  location  of  config  data
export  keyFile="KeyManagerConfig.properties"
#The EKM main class
export  ekmClass="com.ibm.keymanager.KMSAdminCmd"
export  JZOS_MAIN_ARGS="$XXXX  $ZZZZ"

#  Configure  JVM  options  (if  any)
#Uncomment this only for JCERACFKS
#IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwr.provider"
#Uncomment this only for JCE4758RACFKS
IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider"
#  for  jceks  and  jce4758ks/jceccaks,  no  IJO  definition  is  required.
#  comment them out

export  IBM_JAVA_OPTIONS="$IJO  "

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

Customize the sample procedure in Example 6-11 for your system.

Example 6-11 Start procedure

```
//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=EKMSERV.ENCRYPT.CONFIG(EKM2ENV),DISP=SHR
//*
```

Starting EKM with a command

The EKM process can now be started with the operator **start** command as a started task. The following command starts the EKM server:

```
S EKMSERV
```

In Figure 6-2, we have started the EKM as a job using JZOS. We can see the "`Loaded drive keystore`" message and that EKM is in fact up, running, and communicating with the console.

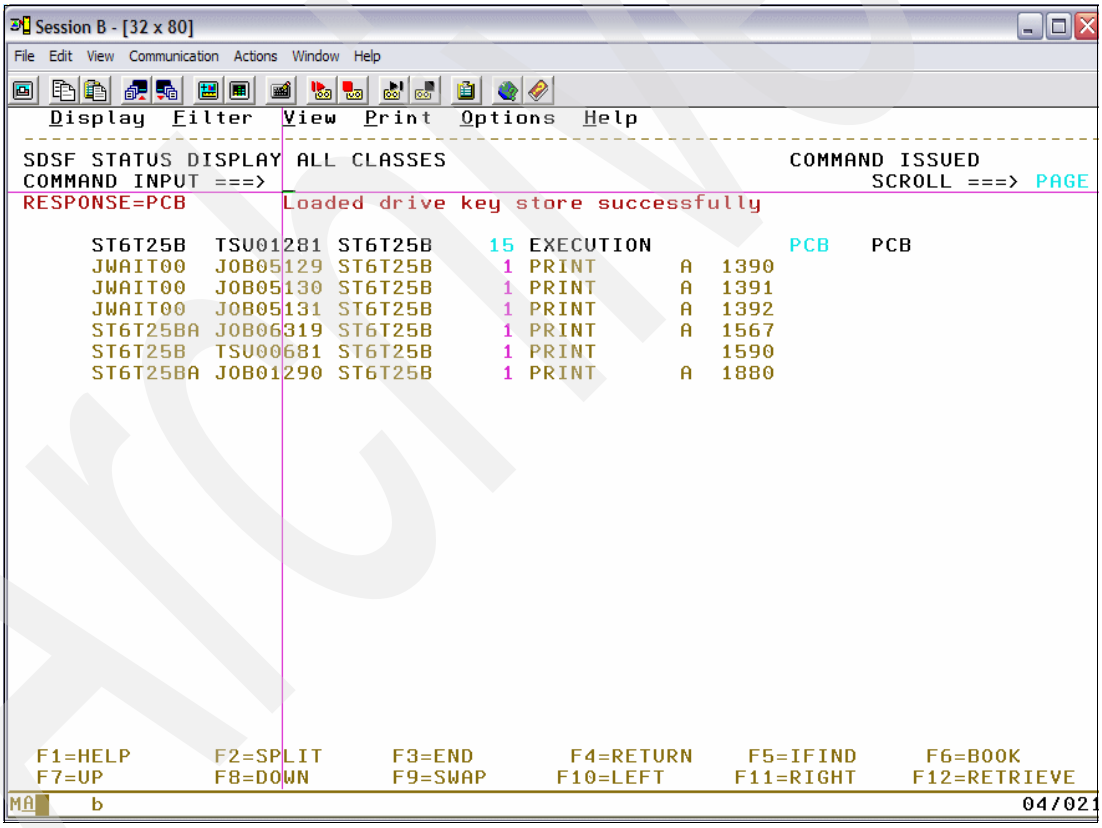


Figure 6-2 EKM Start message

If we execute the following command, the message shown in Figure 6-3 on page 173 displays:

```
f EKMSERV,appl='status'
```

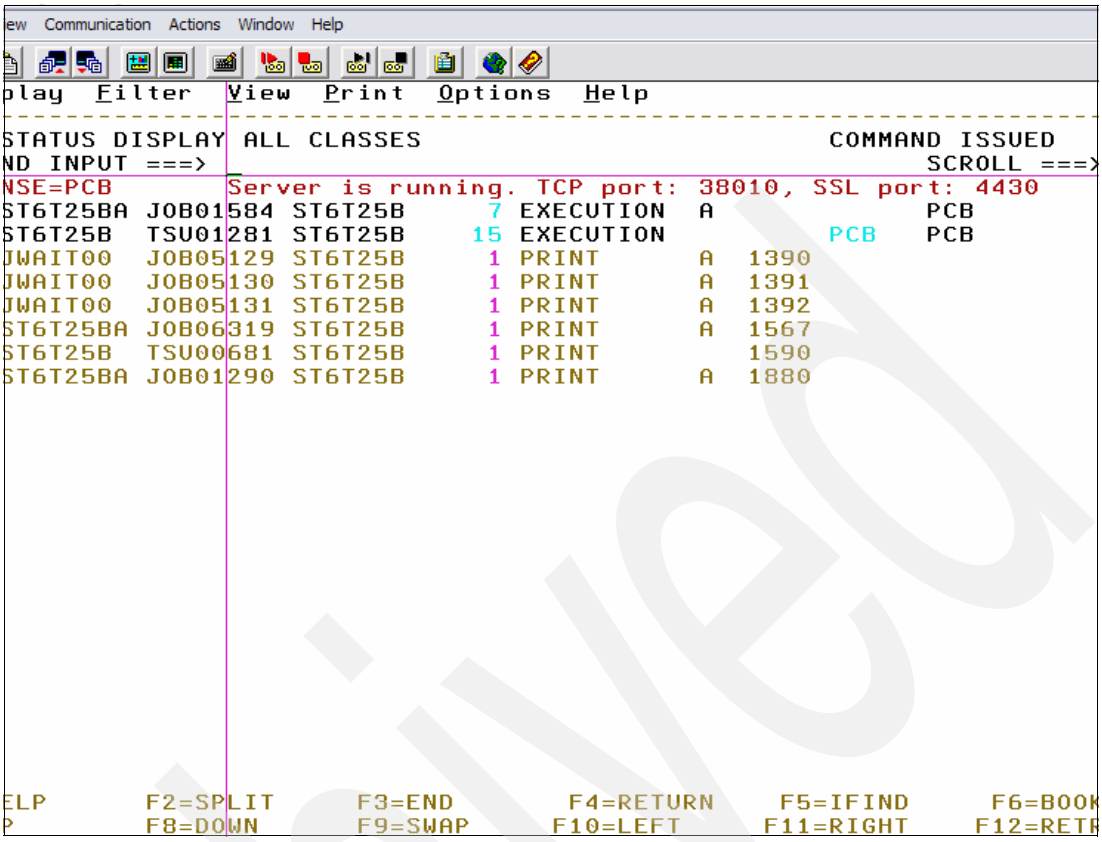


Figure 6-3 EKM Status

We can also list how many certificates were loaded into the EKM by executing:

```
f EKMSERV,appl='listcerts'
```

The output from this command is listed in Figure 6-4 on page 174.

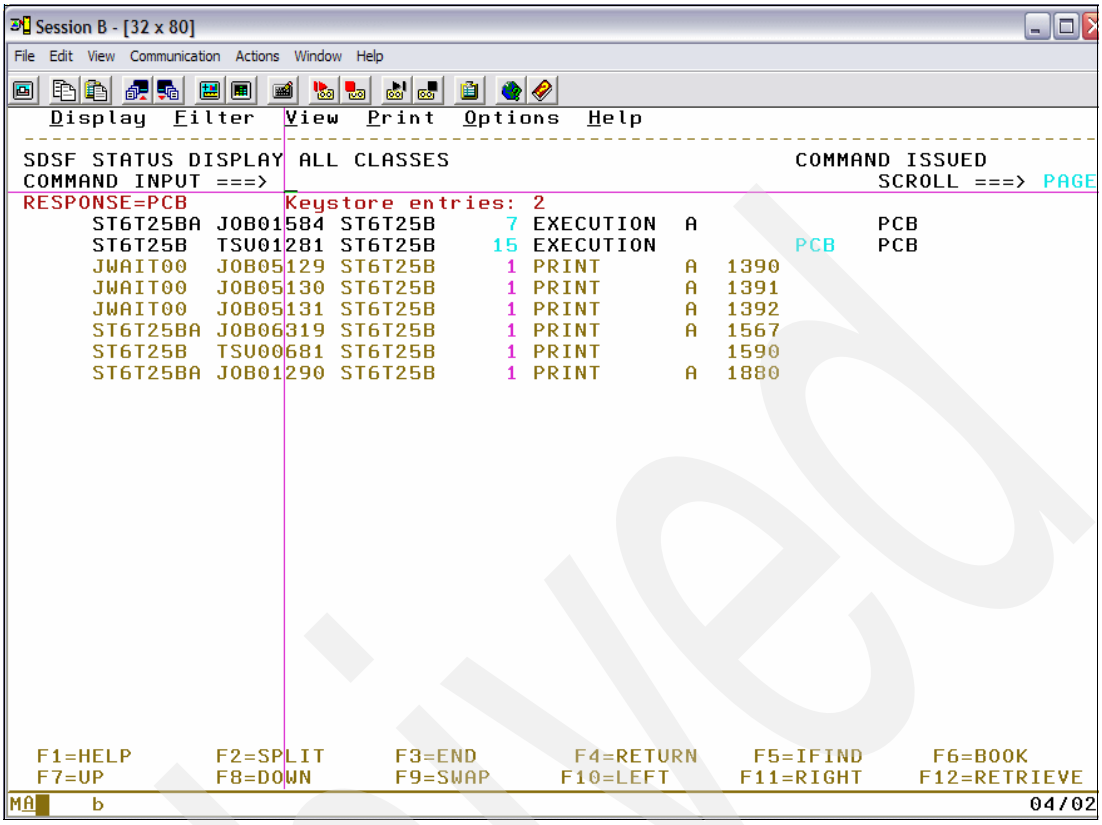


Figure 6-4 EKM certificate list

For a complete listing of EKM commands, refer to 8.1, “EKM commands” on page 276.

Note: A RACF keyring must have a CA certificate and a personal certificate connected to it in order for EKM to load the keystore. If it does not a CertPath error occurs and EKM fails to start. The personal certificate does not need the whole certificate chain; however, it is good practice to always connect the whole chain to the ring.

In this example, note that our EKM is running its SSL listener on port 4430. WebSphere® might use port 4430 for SSL processing, so we changed the port so EKM does not collide with WebSphere SSL processing.

The following section describes the use of keystores.

6.1.7 Additional definitions of hardware keystores for z/OS

This section describes the most secure keystore with a keyring using JCE4758RACFKS or JCECCARACFKS.

The following additional examples show sequences of RACF commands where certificates and keystores are created. The keyring is named ULIRING and the certificate is named ULICERT1.

In Example 6-12 on page 175, we create a keyring with the name ULIRING.

Example 6-12 Creating a keyring

```
RACDCERT ADDRING(ULIRING)
```

In Example 6-13, we create a self-signed certificate authority (CA) called ULICA.

Example 6-13 Creating a self-signed certificate authority

```
RACDCERT GENCERT CERTAUTH SUBJECTSDN(CN('ULICA') OU('ITSO')O('IBM') L('TUCSON')
SP('AZ') C('USA'))  ICSF WITHLABEL('ULICA')
```

In Example 6-14, we allow the generation of a personal certificate signed to the CA that was previously generated.

Example 6-14 Generating a personal certificate

```
RACDCERT GENCERT SUBJECTSDN(CN('ULICERT1') OU('ITSO') O('IBM')L('TUCSON') SP('AZ')
C('USA'))  ICSF WITHLABEL('ULICERT1') SIGNWITH(CERTAUTH LABEL('ULICA') )
```

In Example 6-15, we connect the certificate authority to the keyring.

Example 6-15 Connecting the certificate authority to the keyring

```
RACDCERT CONNECT(CERTAUTH LABEL('ULICA') RING(ULIRING) USAGE(CERTAUTH))
```

In Example 6-16, you find the definition for a key label called ULICERT1.

Example 6-16 Defining a key label

```
RACDCERT CONNECT(LABEL('ULICERT1') RING(ULIRING) USAGE(PERSONAL) DEFAULT)
```

In Example 6-17, we create an additional key label with the name ULICERT2, which is needed later for a second key label statement in the JCL.

Example 6-17 Creating an additional key label

```
RACDCERT CONNECT(LABEL('ULICERT2') RING(ULIRING) USAGE(PERSONAL) DEFAULT)
```

In Example 6-18, we show an RACF command where we delete a key label with the name ULICERT3.

Example 6-18 Deleting a key label

```
RACDCERT DELETE (LABEL('ULICERT3'))
```

In Example 6-19, the command deletes a keyring.

Example 6-19 Deleting a keyring

```
RACDCERT DELRING(ULIRING)
```

6.1.8 Virtual IP Addressing

In TCP/IP networking, Internet Protocol (IP) addresses are typically assigned to physical network interfaces. If a server has two physical interfaces, a separate IP address is assigned to each of them. IBM introduced the concept of Virtual IP Addressing (VIPA) for its z/OS environment in order to support the use of IP addresses representing TCP/IP stacks,

applications, or clusters of applications that are not tied to any specific physical interface. The association between a VIPA and an actual physical interface is subsequently accomplished using either the Address Resolution Protocol (ARP) or dynamic routing protocols (such as OSPF). For details, refer to *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 High Availability, Scalability, and Performance,* SG24-7171.

The TCP/IP infrastructure, including any VIPA definitions, is transparent to the EKM. That is, the EKM does not have to know that it was reached using a VIPA. So, the only place that has to be configured to take advantage of a VIPA installation is IECIOS*xx* by using the VIPA IP address instead of the system host name. In an out-of-band solution, the device is configured to the VIPA address instead of the direct host name of the EKM. Therefore, it is probably a good idea to ensure that the EKMs that are behind the VIPA addresses in a Sysplex all point to the same keystore, or at the very least, that the same keys under the same labels exist in the keystores that you are using.

Note: We show an example of the IECIOS PARMLIB member that includes the EKM definitions in 12.6.2, “Update PARMLIB member IECIOSxx” on page 395.

6.1.9 EKM TCP/IP configuration

Most often, the definitions described in 6.1.8, “Virtual IP Addressing” on page 175 are sufficient. However, if you have a large environment and want to use automatic backup solutions for EKM in a complex Sysplex environment, the hints appended to the commands can help give you an idea of how to define a solution. TCP/IP experienced and trained personnel will be able to define this type of structure.

All of the TCP/IP configuration, including VIPAs, is done in the TCP/IP profile. For all images that are running the EKM, the following information is required in the profile:

- Ports 3801 and port 1443 must be reserved in the PORT section:

```
PORT
1443 TCP EKM                ;Key Manager SSL
3801 TCP EKM                ;Key Manager
```

- If a port is already being used, the option SHAREOPT must be added to each of the applications using the port. For example:

```
PORT
1443 TCP EKM SHAREOPT       ;Key Manager
1443 TCP IMWEB SHAREOPT     ;Web Server
```

- EKM can also be started when TCP/IP starts up by adding it to the AUTOLOG section. For example:

```
AUTOLOG
EKM                         ;Key Manager
ENDAUTOLOG
```

The following description is for a 10-way Sysplex with EKM running on most of the images. A Sysplex Distributor was configured using distributed VIPAs. The *Sysplex Distributor* distributes each key request based on the Workload Manager (WLM), if configured that way, to the appropriate stack/image. In case of a stack failure, WLM moves the Sysplex Distributor to another image where the requests will be handled. The backup stack has to be configured in the TCP/IP profile.

To set up a Sysplex Distributor:

1. Ensure you have a dynamic cross-system coupling facility (XCF) component; it is required. This is the path that is used for the requests through the Sysplex Distributor:

   ```
   IPCONFIG DYNAMICXCF 192.168.49.30 255.255.255.03
   ```

   Do this on all images where the Sysplex Distributor might be the backup.

2. Define the required DVIPA parameters in the VIPADYNAMIC section of the profile:

   ```
   VIPARANGE DEFINE 255.255.255.255 9.xxx.xxx.xxx;keystore
   ```

3. Define where the requests will be distributed that are sent to the DVIPA 9.*xxx.xxx.xxx*. This definition might look like the following example:

   ```
   VIPADEFINE MOVEABLE IMMED 255.255.255.0 9.xxx.xxx.xxx
   VIPADISTRIBUTE DEFINE SYSPLEX DISTM ROUNDROBIN 9.xxxxxxxxx PORT 3801 1443
   DESTIP ALL
   ENDVIPADYNAMIC
   ```

   The described definitions tell you that all requests that come to DVIPA address 9.*xxx.xxx.xxx* Port 3801 or Port 1443 are to be sent to all images in the Sysplex that are configured to accept requests (dynamic XCF setup).

4. For the images that were configured as backup, you should have dynamic XCF EKM running. The following example, tells the Sysplex which DVIPAs you are backing up:

   ```
   VIPADYNAMIC
   VIPABACKUP 9.xxx.xxx.xxx
   ENDVIPADYNAMIC
   ```

Example 6-20 shows a complete definition example.

Example 6-20 Definition example

```
IPCONFIG SYSPLEXROUTING DYNAMICXCF 193.9.200.4 255.255.255.240.1
VIPADYNAMIC
VIPADEFINE 255.255.255.192 9.67.240.02
VIPADISTRIBUTE DEFINE 9.67.240.02 PORT 3801 1443 DESTIP
193.9.200.2
193.9.200.4
193.9.200.6
ENDVIPADYNAMIC
```

6.2 Installing EKM on AIX

In this section, we describe the steps necessary to install the EKM in the AIX environment. We refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418, to install the EKM on our AIX server.

6.2.1 Install the IBM Software Developer Kit (SDK)

In the following section, we describe how to install the IBM Software Developer Kit (SDK) for AIX and other UNIX operating systems. We downloaded the Java Runtime Environment from the IBM developerWorks® Web site. And, we created the required directories and installed the Java Runtime Environment.

To install the SDK:

1. Download the SDK from the following Web site (which is reflected in Figure 6-5):

   http://www.ibm.com/developerworks/java/jdk/aix/service.html

Where to get SDK base image and JRE	Java 5 32-bit	Java 5 64-bit	Java 1.4.2 32-bit	Java 1.4.2 64-bit	Java 1.3.1 32-bit	Java 1.3.1 64-bit	Java 1.3.0 32-bit (latest updates using 1.3.1 32-bit code)
Where to get SDK fixes (PTFs)	Fix Info	Fix Info	Fix Info	Fix Info	Fix Info	Fix Info	Fix Info
Latest fixes list	fixes.html	fixes.html	fixes.html	fixes.html	fixes.html	fixes.html	fixes.html
Latest README/ sdkguide	sdkguide security guide	sdkguide security guide	sdkguide security guide	sdkguide security guide	README	README	README
Minimum AIX level supported - 5.1	n/a	n/a	5100-08 (APAR IY70781)	5100-08 (APAR IY70781)	5100-08 (APAR IY70781)	5100-08 (APAR IY70781)	5100-08 (APAR IY70781)
Minimum AIX level supported - 5.2	5200-07 (APAR IY67914)	5200-07 (APAR IY67914)	5200-06 (APAR IY67913)	5200-06 (APAR IY67913)	5200-06 (APAR IY67913)	5200-06 (APAR IY67913)	5200-06 (APAR IY67913)
Minimum AIX level supported - 5.3	5300-03 (APAR IY71011)	5300-03 (APAR IY71011)	5300-02 (APAR IY69190)	5300-02 (APAR IY69190)	5300-02 (APAR IY69190)	5300-02 (APAR IY69190)	5300-02 (APAR IY69190)
Filesets	Java5.*	Java5_64.*	Java14.*	Java14_64.*	Java131.*	Java13_64.*	Java130.*
Install directory	/usr/java5	/usr/java5_64	/usr/java14/	/usr/java14_64/	/usr/java131/	/usr/java13_64/	/usr/java130/
End of Service	30 Sept 2012	30 Sept 2012	30 Sept 2011	30 Sept 2011	30 Sept 2007	30 Sept 2007	31 Dec 2002 (30 Sept 2007 if using latest 1.3.1 32-bit code)

Figure 6-5 Java code Web site

   We selected and downloaded the Java 5, 32-bit Runtime Environment to our personal computer.

2. Create a new directory.

   We created a directory named `/usr/lpp/java5` and used FTP to get the `j532redist.tar` file to that directory, as shown in Example 6-21.

Example 6-21 New directory with Java code

```
/usr/lpp/java5>ls
j532redist.tar
/usr/lpp/java5>
```

3. Run the **`tar -xvf j532redist.tar`** command. The remainder of the Java runtime directory tree is built as shown in Example 6-22.

Example 6-22 After tar of Java runtime library

```
/usr/lpp/java5>ls
j532redist.tar  license         sdk

/usr/lpp/java5>ls sd*
COPYRIGHT   bin         docs        fixes.html  include     jre         lib
```

4. Edit the `/home/root/.profile` file, so that the JAVA_HOME, P8, P9, and CLASSPATH statements reflect the correct directories. Refer to Example 6-23.

Example 6-23 .profile file

```
/home/root>vi .profile
#NAME  .profile
JAVA_HOME=/usr/lpp/java5/sdk/jre
#JAVA_HOME=/usr/java14/sdk/jre
P8=/usr/lpp/java5/sdk/jre/bin
P9=/usr/lpp/java5/sdk/bin
CLASSPATH=/usr/lpp/java5/sdk/jre/lib
PATH=$JAVA_HOME:$P1:$P2:/etc:$P3:$HOME:$P4:$P5:$P6:$P7:$P8:$P9:.
```

5. Ensure that JAVA_HOME was set correctly. Because we use BASH as a shell, we checked the .bashrc file to ensure that JAVA_HOME was set correctly, as shown in Example 6-24.

Example 6-24 .bashrc profile

```
export JAVA_HOME=/usr/lpp/java5/sdk
```

6. Log off and log in again.
7. Determine the Java version.

   We issued the **`java-version`** command, the results of which are in Example 6-25. In addition to the logout and login process, you can also execute the profile using the following command:

   ```
   . ./.profile
   ```

Example 6-25 Java version command results

```
/home/root>java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pap32dev-20060511 (SR2))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 AIX ppc-32 j9vmap3223-20060504 (JIT enabled)
J9VM - 20060501_06428_bHdSMR
JIT  - 20060428_1800_r8
GC   - 20060501_AA)
JCL  - 20060511a
/home/root>
```

8. Ensure the correct directories are used.

   We used the **`echo $JAVA_HOME`** command, the results of which are in Example 6-26, to ensure that the correct directories are used.

Example 6-26 The echo command results

```
/home/root>echo $JAVA_HOME
/usr/lpp/java5/sdk
/home/root>
```

9. Install policy files.

   At this point in the installation, we installed only the *restricted policy files*. However, the EKM requires that the *unrestricted policy files* are installed in the JVM, before the EKM is able to generate keys. You must do this installation step *after the installation of the Java code,* otherwise, EKM is not able to serve keys.

10. Use the JRE 1.4.2 files. These files are different from the JRE 1.4.1 files. The JRE 1.4.2 files can also be used for JRE 5. The .zip file must be unpacked and the two JAR files placed in the JRE's jre/lib/security/ directory, replacing the existing files of the same name. These policy files are for use with Software Developer Kits (SDKs) that were developed by IBM.

    Here are the steps we followed to replace the policy files. Begin by pointing your browser to the following Web site:

    https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk

    a. Sign in. You are directed to a location where you can download the Unrestricted JCE Policy files for SDK 1.4.

    b. Download the unrestricted policy files for 1.4.2 SDK (these are also the files for SDK 5.0). The result is a file called `unrestrict.zip`. You can unzip this using the **`pkzip`**, **`tar`**, or **`jar`** command.

    c. Remove the files `local_policy.jar` and `US_export_policy.jar` from the `jre\lib\security` directory.

    d. Put the new files from the .zip file into the `jre\lib\security` directory. The files have the same names as the files removed in step c.

    See Example 6-27.

Example 6-27 EKM policy location

```
/usr/lpp/java5/sdk/jre/lib/security>ls -l
total 112
-rw-r-----   1 root     system         2199 Sep 22 07:39 US_export_policy.jar
-rw-r--r--   1 root     sys           29731 May 11 2006  cacerts
-rw-r--r--   1 root     sys            2646 May 11 2006  java.policy
-rw-r--r--   1 root     sys            9609 May 11 2006  java.security
-rw-r-----   1 root     system         2212 Sep 22 07:39 local_policy.jar
```

Creating a keystore using keytool

Now that Java is installed, create the files that are required by the Encryption Key Manager. The keystore is the first file to be created and populated. For this step, we use a command line utility called **`keytool`**. Additional information about the keytool is available at:

ftp://ftp.software.ibm.com/s390/java/jce/keytool.html

Use the *Keytool User Guide* as a reference. The guide is located at:

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

Example 6-28 shows the script that we used to create our keystore.

Example 6-28 Keystore create keytool commands

```
keytool  -genkey  \
-alias   cert1 \
-dname  CN=myCo  \
-keystore  tonyskeys.jcks  \
-provider  IBMJCE  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  "passphrase"  \
-storepass  "passphrase" \
```

```
-storetype  JCEKS \
-validity  999

keytool  -genkey  \
-alias  cert2  \
-dname  CN=myCo  \
-keystore  tonyskeys.jcks  \
-provider  IBMJCE  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  "passphrase"  \
-storepass  "passphrase" \
-storetype  JCEKS \
-validity  999

keytool  -export \
-file  rsa2048Cert1.cer \
-keystore  tonyskeys.jcks \
-alias  cert1 \
-storepass  passphrase \
-storetype  JCEKS  \
-provider  IBMJCE  \
-keypass  passphrase

keytool  -import  \
-file  rsa2048Cert1.cer \
-keystore  tonyskeys.jcks \
-alias  cert1ca \
-storepass  passphrase \
-storetype  JCEKS \
-provider  IBMJCE \
-keypass  passphrase
```

Importing and exporting certificates and why

The last two commands in the script **keytool export** and **keytool import** are there to establish a trusted certificate. You import a certificate for one of the following reasons:

- To add it to the list of trusted certificates
- To import a certificate reply received from a CA as the result of submitting a Certificate Signing Request to that CA

A trusted certificate is required for an SSL connection. The SSL protocol is described in detail in the “Secure Sockets Layer example” on page 20.

When more than one EKM is used in the IBM tape data encryption solution, the primary and secondary EKMs synchronize information using an SSL connection.

The value of the -alias option indicates which type of import is intended:

- If the alias points to a key entry, keytool assumes that you are importing a certificate reply. Keytool checks whether the public key in the certificate reply matches the public key stored with the alias and exits if they are different.
- If the alias does not point to a key entry, keytool assumes you are adding a trusted certificate entry. In this case, the alias does not already exist in the keystore.

If the alias does already exist, keytool outputs an error, because there is already a trusted certificate for that alias, and keytool does not import the certificate. If the alias does not exist in the keystore, keytool creates a trusted certificate entry with the specified alias and associates it with the imported certificate.

Important: Be sure to very carefully check a certificate before importing it as a trusted certificate.

Listing a keystore using keytool

You can use the script listed in Example 6-29 to list a keystore.

Example 6-29 List keystore script

```
keytool  -list \
-keystore  tonyskeys.jcks \
-storetype jceks \
-storepass passphrase
```

The results of the list keystore script reflect that the import did indeed create a trusted certificate. See Example 6-30.

Example 6-30 List keystore script results

```
Keystore type: jceks
Keystore provider: IBMJCE

Your keystore contains 3 entries

cert1ca, Sep 22, 2000, trustedCertEntry,
Certificate fingerprint (MD5): A9:A9:74:F7:FD:A4:27:88:39:28:DF:E4:47:25:33:E7
cert2, Sep 22, 2000, keyEntry,
Certificate fingerprint (MD5): FF:4E:3F:73:B6:26:79:A7:69:11:B1:6E:63:67:OD:91
cert1, Sep 22, 2000, keyEntry,
Certificate fingerprint (MD5): A9:A9:74:F7:FD:A4:27:88:39:28:DF:E4:47:25:33:E7
/home/root/ekmserv>
```

6.3 Installing EKM on a Windows platform

In this section, we describe the installation of EKM in the Windows environment.

The Java Runtime Environment is bundled with IBM TotalStorage Productivity Center - Limited Edition (TPC-LE) - 5608-VC6. This version of TPC is available without charge.

Important: Regardless of which version of IBM SDK you use, you must replace the `US_export_policy.jar` and `local_policy.jar` files in your directory `java_home/usr/java5/jre/lib/security` with new files that you can download for Java 5.0 on Sun, System x, and Hewlett-Packard UNIX (HP-UX) from this Web site:

`https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

These files install the unrestricted policy files that EKM requires in order to serve Advanced Encryption Standard (AES) keys.

Be sure to select the “Unrestricted JCE Policy files for SDK 1.4.2”, which works for both Java 1.4.2 and Java 5.0 SDKs. Do not select the 1.4.1 version, because these files are incompatible.

6.3.1 EKM setup tasks

Before you can encrypt tapes, EKM must first be configured and running so that it can communicate with the TS1120 tape drives. EKM does not need to be running while tape drives are being installed, but it must be running in order to perform encryption. You do not have to set up EKM if you are implementing Application-Managed Encryption.

Perform the following tasks before using EKM:

1. Decide what system or systems to use as EKM servers.
2. Upgrade the server operating system if necessary.
3. Install or upgrade IBM Java Virtual Machine if necessary.
4. Install IBM Java Unrestricted Policy Files.
5. Upgrade EKM JAR if necessary, which you can obtain at the IBM Web site at:

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

   Or visit the following Web site:

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

   Click **Downloads** and look for **IBM Encryption Key Manager for the Java platform**.
6. Decide on keystore type.

   The type of keystore that you select depending on your environment can affect your future flexibility for encryption implementation. Refer to 7.1, “Keystore and SAF Digital Certificates (keyrings)” on page 228 for more details.
7. Define and create a keystore and certificates (AIX and other operating systems) as shown in Example 6-28 on page 180.
8. Import or create keys and certificates into the keystore.

   Refer to “Importing and exporting certificates and why” on page 181.
9. Define the EKM configuration file. See Example 6-35 on page 191.
10. Define the tape drives to the EKM or set the drive.acceptUnknownDrives EKM configuration property value to `ON`.
11. Start EKM.

Important: Ensure that you use the forward slash character (/) in the Java properties EKM configuration file when defining file locations. Because `KeyManagerConfig.properties` is a Java properties file, only forward slashes are recognized in pathnames, even in Windows. If you use the back slash character (\) in the KeyManagerConfig.properties file, errors will occur.

6.3.2 Installing the IBM Software Developer Kit on Windows

In this section, we guide you through the steps to install the IBM Software Developer Kit (SDK).

Installation steps

To install the IBM SDK on Windows:

1. Insert the correct CD from the IBM TotalStorage Productivity Center - Limited Edition (TPC-LE) - LPP 5608-VC6 and select your IBM Java Runtime Environment:
   - Java 5.0 Service Release 2 (Windows/AMD64/EM64T)
   - Java 5.0 Service Release 2 (Windows/IA32)
   - Java 1.4.2 Service Release 5 (Windows/IA64)
2. Select a setup language. See Figure 6-6.

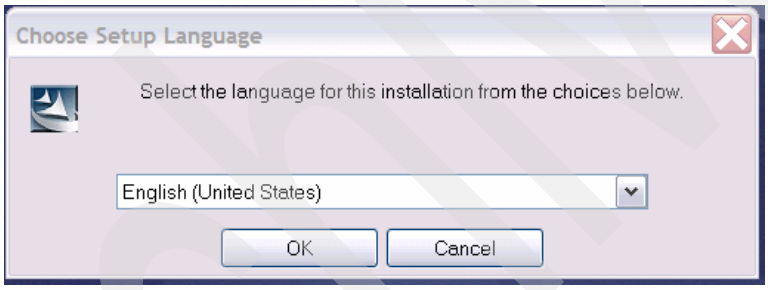


Figure 6-6 Language choice

3. Click **OK**. The installation wizard opens, as shown in Figure 6-7 on page 185.

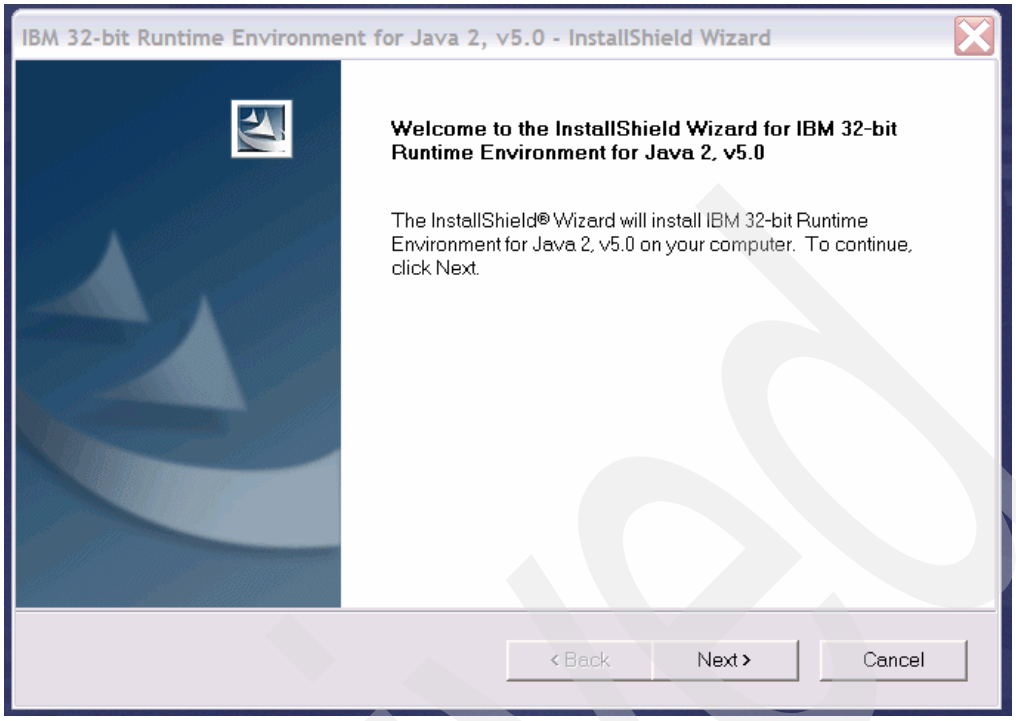


Figure 6-7 InstallShield Wizard

4. Click **Next**. The License Agreement window opens. See Figure 6-8.

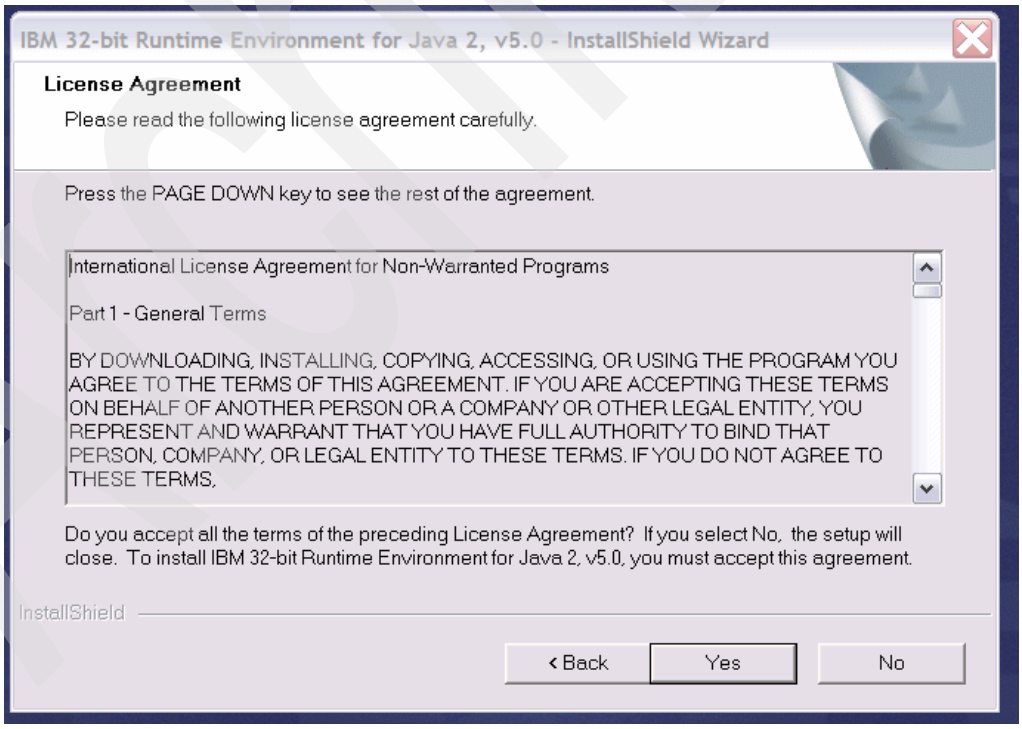


Figure 6-8 License agreement

5. Read the License Agreement and if acceptable, click **Yes**.

   The Choose Destination Location window opens. See Figure 6-9 on page 186.

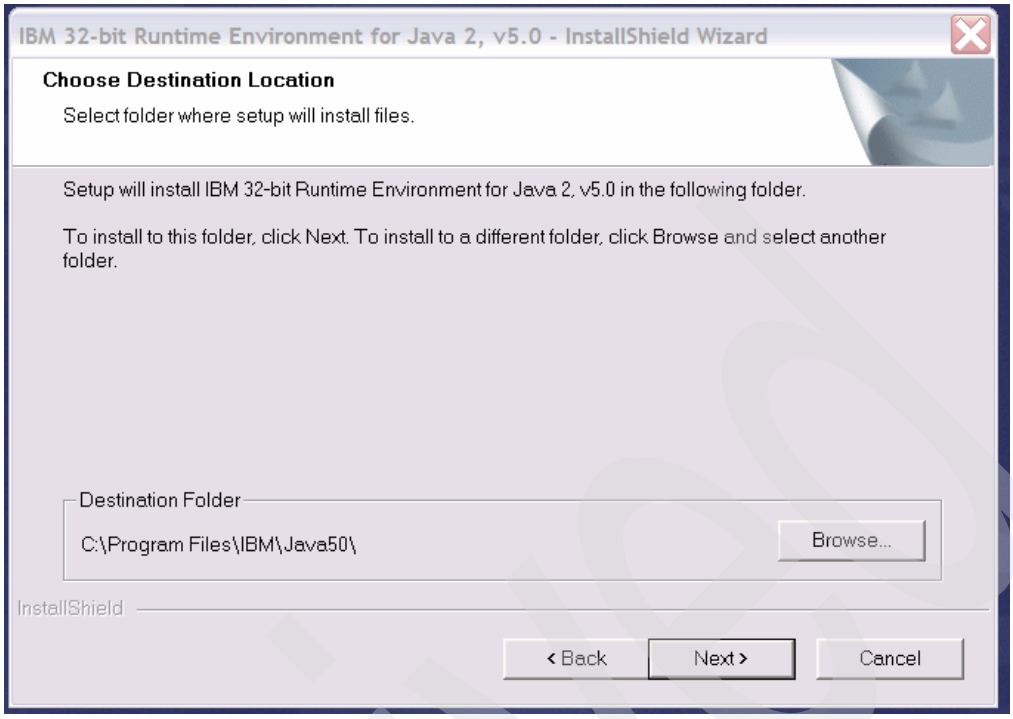


Figure 6-9 Java installation destination location

6. Either confirm or choose a folder and make note of it. You will need this Java path to launch EKM. Click **Next**.

   A confirmation window opens, asking if you want this Java Runtime Environment as the default System JVM. See Figure 6-10.

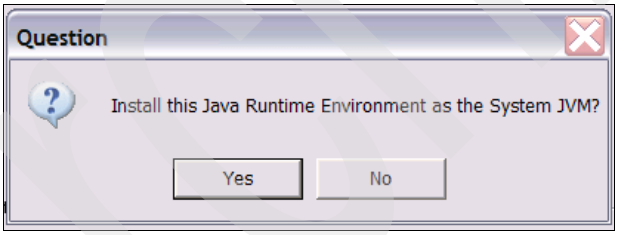


Figure 6-10 Do you want this version for your default System JVM

7. Click **No**.

   A window opens prompting you to review the target directories that Java will use. See Figure 6-11 on page 187.

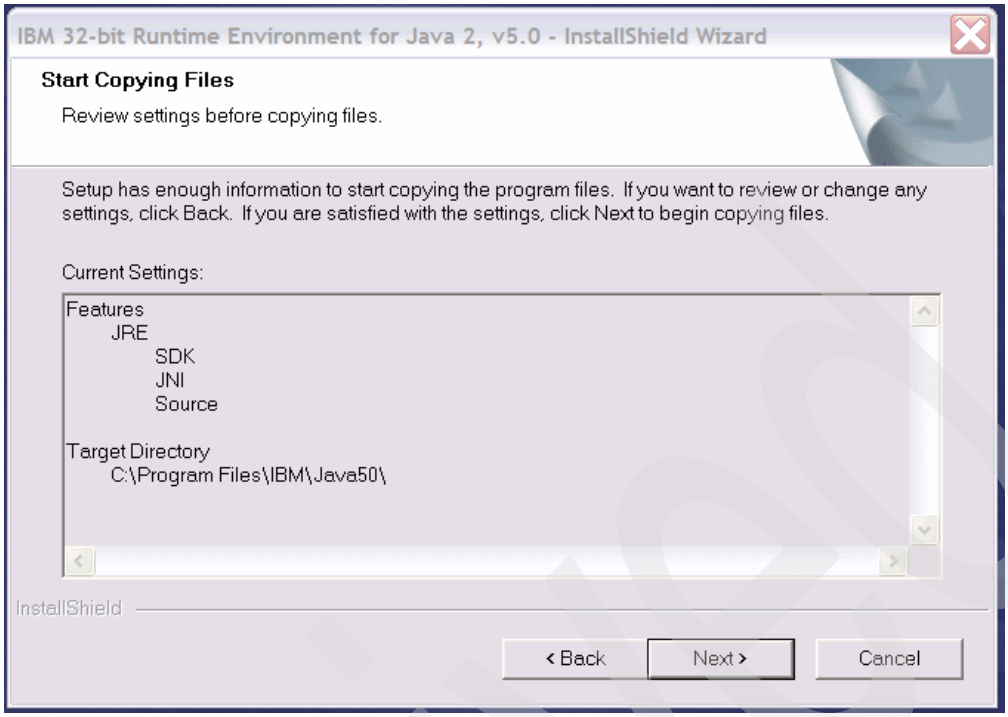


Figure 6-11 Review target directory

8. If acceptable, click **Next** to start the installation.
9. The last window that opens confirms that Java was installed successfully. See Figure 6-12. Click **Finish** to complete the installation.

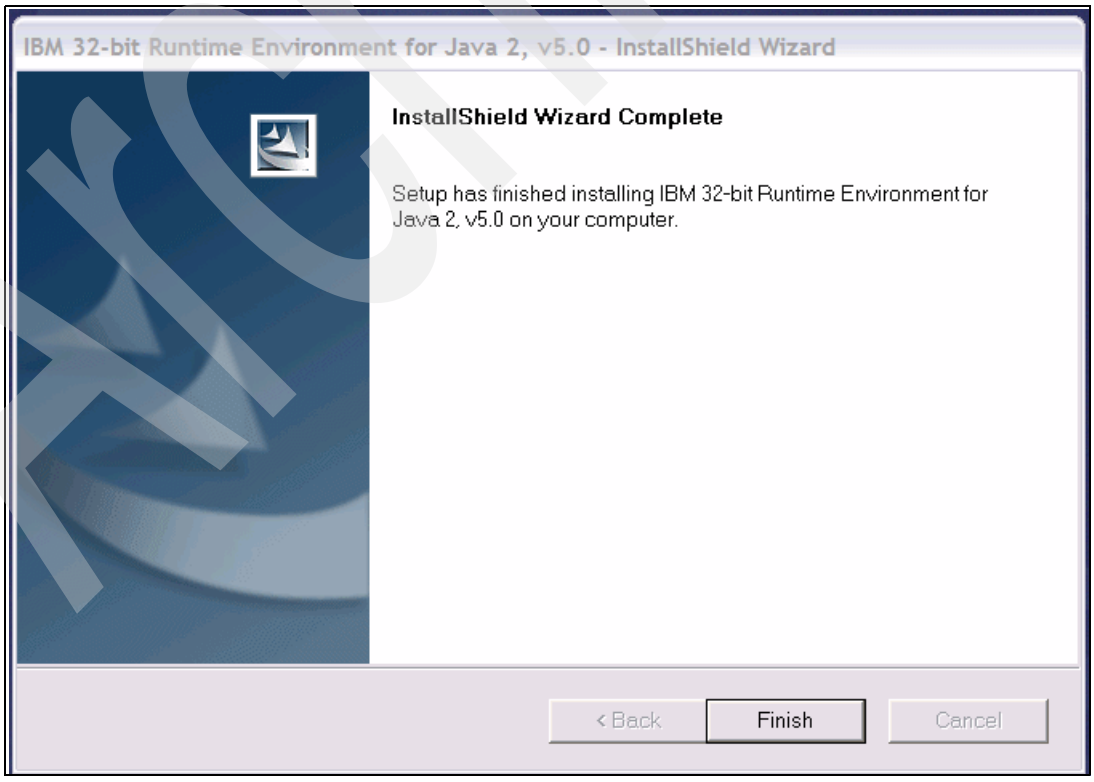


Figure 6-12 Installation complete

At a Windows C: prompt, typing **`Java -version`** results in Example 6-31.

Example 6-31 After Java installation

```
C:\Documents and Settings\Administrator>java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pwi32devifx-20060124)
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Windows XP x86-32 j9vmwi3223ifx-2006
0124 (JIT enabled)
J9VM - 20051027_03723_lHdSMR
JIT  - 20051027_1437_r8
GC   - 20051020_AA)
JCL  - 20060120
```

Replacing the Java JAR files

Regardless of the version of IBM SDK that you use, you must replace the `US_export_policy.jar` and `local_policy.jar` files in your directory `java_home/usr/java5/jre/lib/security` with new files that you can download from:

`https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

These files install the unrestricted policy files that EKM requires in order to serve AES keys. See Figure 6-13.

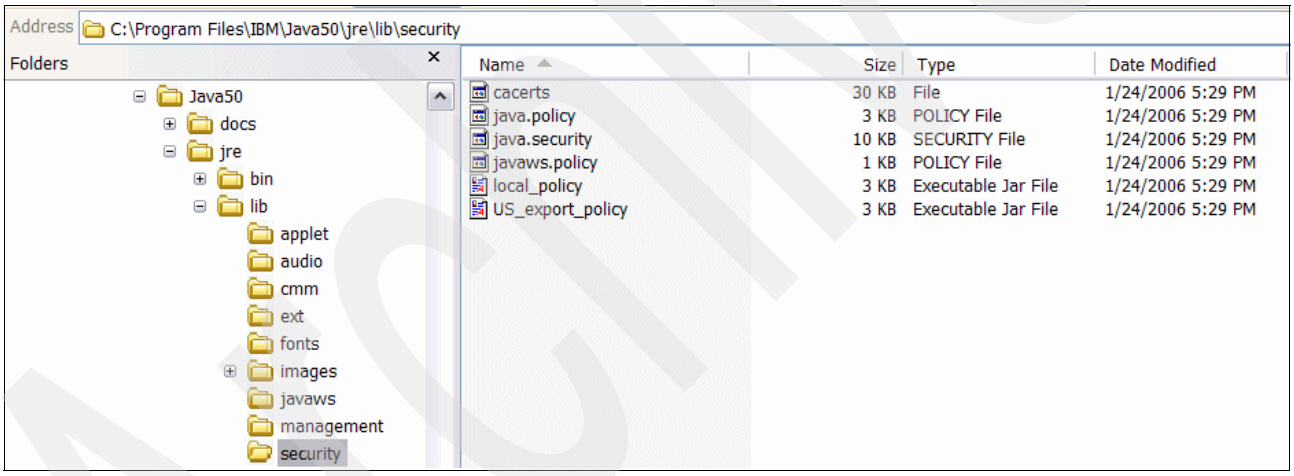


Figure 6-13 Replacing Jar files

The location of the JAR files that have to be replaced are shown in Figure 6-13.

Upgrade EKM Jar

Download the latest `IBMKeyManagementServer.jar` and `KeyManagerConfig.properties` files from the IBM Web site at:

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

Or, visit the following Web site and click **downloads** and look for **IBM Encryption Key Manager for the Java platform**. Then, download it into a directory of your choice:

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

In our example, we created a directory named `C:\$$$$ekm` and copied the `KeyManagerConfig.properties` file into it.

The IBMKeyManagementServer.jar file must be copied into the `C:\Program Files\IBM\Java50\jre\lib\ext\` directory, which was created during the Java SDK installation. The directory is shown in Figure 6-14.

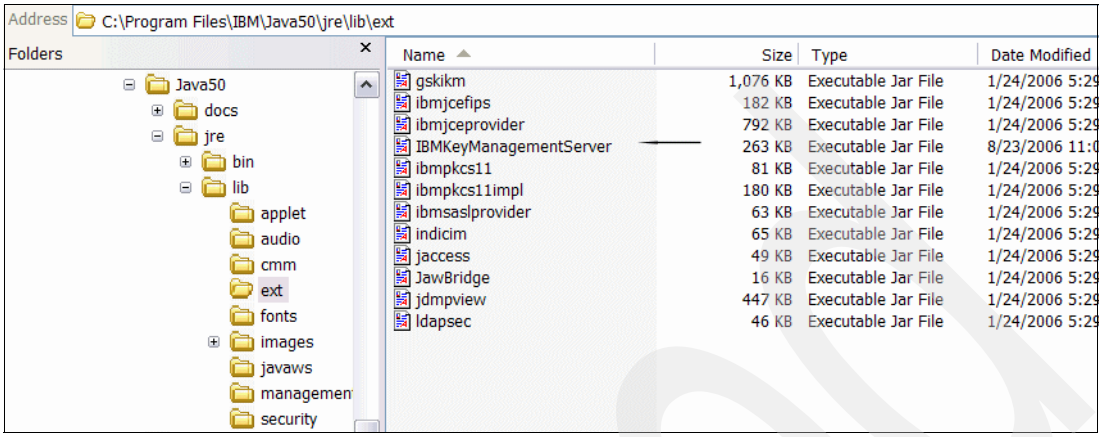


Figure 6-14 Jar file copied

Editing the EKM config file

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. An example of using forward slashes in relevant KeyManagerConfig parameters is shown in Example 6-32.

Example 6-32 Java Config forward slash example

```
Audit.handler.file.directory = keymanager/audit
config.drivetable.file.url = FILE:///keymanager/drivetable
debug.output.file = keymanager/debug
```

We discuss `KeyManagerConfig.properties` parameters in 6.3.4, “Configuring and starting EKM” on page 190.

6.3.3 Starting EKM on Windows

We updated the Windows PATH parameter with the Java directory information, so that we do not have to enter `C:\Program Files\IBM\Java50\bin` when starting EKM.

Starting EKM is a two-step process. To get to the Java command prompt, you enter:

```
java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig_full_file_path_name
```

Here are the steps we used:

1. So that we do not have to fully qualify all the configuration parameters, we ensure that we start in the correct directory that contains our KeyManagerConfig.properties file.

   At the C: prompt, we changed the Windows directory to the directory to which we copied the KeyManagerConfig.properties file, which in our example is C:\$$$$EKM, and enter the following command to start EKM:

   ```
   java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties.jce4758
   ```

2. At the # prompt, we entered **`startekm`**, as shown in Example 6-33 on page 190.

Example 6-33 starting EKM

```
C:\$$$$ekm>java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties.jce4758
# startekm
Loaded drive key store successfully
Starting the Encryption Key Manager 1.0
Processing Arguments
Processing
Server is started
#
```

6.3.4 Configuring and starting EKM

In this section, we discuss coding the EKM configuration file, we start EKM, and then we look at several of the more interesting EKM commands.

Configure EKM

To configure EKM:

1. Download a sample configuration file that you can use as a model from:

   http://www-1.ibm.com/support/docview.wss?&uid=ssg1S4000504

2. Scroll to the IBM EKM Sample Configuration File section identified in Figure 6-15.

DESCRIPTION	DOCUMENTATION	Download Options
Platform Multi-Platform Version Independent US English Byte Size 268588 Date 9/1/2006	Intro Planning & User's Guide	IBM EKM Application - Ver. 08232006 FTP
Platform Multi-Platform Version Independent US English Byte Size 957 Date 9/1/2006	Intro Planning & User's Guide	IBM EKM Sample Configuration File FTP
Platform z/OS Version Independent US English Byte Size 32256 Date 9/8/2006	Download for z/OS batch only	JZOSEKM files for z/OS Batch FTP

Figure 6-15 Location of sample EKM configuration file

3. Click **FTP** to get the sample `KeyManagerConfig.properties` configuration file. Edit it to suit your environment. Example 6-34 on page 191 shows the KeyManagerConfig.properties file that was downloaded from the Web site that you can use as a example to get started.

Example 6-34 Sample EKM KeyManagerConfig.properties file

```
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
Admin.ssl.keystore.name = testkeys
Admin.ssl.truststore.name = testkeys
config.drivetable.file.url = FILE://keymanager/drivetable
debug = none
debug.output = simple_file
debug.output.file = keymanager/debug
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 = testkeys
TransportListener.ssl.truststore.type = jceks
TransportListener.tcp.port = 3801
config.keystore.file = keymanager/testkeys
config.keystore.provider = IBMJCE
config.keystore.type = jceks
```

We changed several entries to match the naming conventions that we used. We changed very little. Example 6-35 is the file that we used to start EKM.

Example 6-35 Our edited KeyManagerConf.properties file

```
/home/root/ekmserv>cat startj.sh
java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties
/home/root/ekmserv>cat KeyManagerConfig.properties
TransportListener.ssl.port = 443
TransportListener.tcp.port = 3801
fips = Off
Admin.ssl.keystore.name = tonyskeys.jcks
config.keystore.provider = IBMJCE
config.keystore.password = passphrase
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.ciphersuites = JSSE_ALL
Audit.handler.file.size = 10000
TransportListener.ssl.truststore.name = tonyskeys.jcks
Audit.handler.file.directory = keymanager/audit
TransportListener.ssl.protocols = SSL_TLS
config.keystore.file = tonyskeys.jcks
TransportListener.ssl.truststore.type = jceks
debug.output = simple_file
TransportListener.ssl.keystore.name = tonyskeys.jcks
Audit.eventQueue.max = 0
debug.output.file = keymanager/debug
TransportListener.ssl.keystore.type = jceks
Audit.handler.file.name = kms_audit.log
config.keystore.type = jceks
```

```
Audit.event.outcome = success,failure
debug = none
Audit.event.types = all
config.drivetable.file.url = FILE://keymanager/drivetable
Admin.ssl.truststore.name = tonyskeys.jcks
```

Important: Regarding the FIPS parameter, EKM does not provide cryptographic capabilities and therefore, EKM does not require and not allowed to obtain FIPS 140-2 certification. However, EKM 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 EKM use the IBMJCEFIPS provider for all cryptographic functions.

In addition to setting the FIPS parameter in the Configuration properties file, the following provider must be added to the java.security file in $JAVA_HOME/lib/security/:

```
security.provider.?=com.ibm.crypto.fips.provider.IBMJCEFIPS
```

This provider must be added before the IBMJCE provider. Renumber the providers accordingly.

4. If you want to, create a script that tests EKM.

   To save effort, we created the script in Example 6-36 so that we did not have to retype the entire command every time that we wanted to test the EKM. This script also proved helpful to keep track of the correct version of the keymanagerconfig.properties file that we wanted to use, because we had several versions in different states for testing purposes.

Example 6-36 startj.sh script

```
/home/root/ekmserv>cat startj.sh
java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties
```

Start EKM

You are now ready to start the EKM, which is a two-step process.

To start EKM:

1. Get to the command prompt of EKM, which you do with the following command:

   ```
   java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties
   ```

   We used the **startj.sh** script as shown in Example 6-37 to get to the command prompt.

Example 6-37 EKM command prompt

```
/home/root/ekmserv>startj.sh
Sep 29, 2000 9:14:03 AM com.ibm.keymanger.config.ConfigImpl get
FINER: ENTRY
Sep 29, 2000 9:14:03 AM com.ibm.keymanger.config.ConfigImpl get
ALL: debug.output = simple_file
Sep 29, 2000 9:14:03 AM com.ibm.keymanger.config.ConfigImpl get
FINER: RETURN
#                                                  <<<< -----EKM command prompt
```

2. At the EKM command prompt, enter **startekm**. The results are shown in Example 6-38.

Example 6-38 Starting EKM

```
# startekm
Loaded drive key store successfully
Starting the Encryption Key Manager 1.0-20060823
Processing Arguments
Processing
Server is started
#
```

EKM commands

After EKM is running, entering help at the # command prompt provides a list of valid commands as shown in Example 6-39.

Example 6-39 EKM command examples

```
# help
EKMAdmin usage:

adddrive -drivename <name> [-rec1 <alias>] [-rec2 <alias>]

deldrive -drivename <name>
equivalent command is rmdrive or deletedrive or removedrive

exit or quit

export -drivetab|-config -url <url name>

import -merge|-rewrite -drivetab|-config -url <url name>

listcerts [-alias <alias>] [-verbose|-v]

listconfig

listdrives [-drivename <drive_name> [-verbose|-v]]

logout - equivalent command is logoff
Only useful when LoginModule is enabled

modconfig -set -property <name> -value <value> | -unset -property <name>
equivalent command is modifyconfig

moddrive -drivename <name> [-rec1 alias] [-rec2 alias]
equivalent command is modifydrive

refresh

startekm

status

stopekm

version

sync -all|-config|-drivetab -ipaddr <ip address:ssl port> [-merge|-rewrite]
```

```
Note that dashes and spaces must be contained in quotes for either to show up
as part of an argument value.
```

And, because we neglected to code the drive.accept.unknown.drives parameter in the EKM configuration, we modified the running configuration using the EKM **modconfig** command as shown in Example 6-40.

Example 6-40 The modconfig command

```
# modconfig -set -property drive.acceptUnknownDrives -value true
```

A **listconfig** of the configuration file "in use" reflects that the **drive.acceptUnknownDrives** parameter is now specified correctly. See Example 6-41.

Example 6-41 Config after modconfig command

```
# listconfig
debug.output=simple_file
config.drivetable.file.url=FILE://keymanager/drivetable
config.keystore.password=passphrase
Admin.ssl.keystore.name=tonyskeys.jcks
Audit.handler.file.directory=keymanager/audit
Audit.event.types=all
Admin.ssl.truststore.name=tonyskeys.jcks
debug.output.file=keymanager/debug
TransportListener.ssl.protocols=SSL_TLS
Audit.handler.file.name=kms_audit.log
TransportListener.ssl.keystore.name=tonyskeys.jcks
Audit.eventQueue.max=0
TransportListener.tcp.port=3801
TransportListener.ssl.truststore.name=tonyskeys.jcks
Audit.handler.file.size=10000
config.keystore.file=tonyskeys.jcks
config.keystore.type=jceks
TransportListener.ssl.ciphersuites=JSSE_ALL
TransportListener.ssl.clientauthentication=0
TransportListener.ssl.port=443
TransportListener.ssl.keystore.type=jceks
debug=none
config.keystore.provider=IBMJCE
TransportListener.ssl.truststore.type=jceks
Audit.event.outcome=success,failure
drive.acceptUnknownDrives=true
fips=Off
```

Example 6-42 on page 195 shows a few examples of sample output provided by EKM commands. Note that the tape drive we used to test writing an encrypted tape is now reflected in the drive table output of the **listdrives** command.

Example 6-42 Command responses

```
# listdrives
Drive entries: 1
SerialNumber = 000001365109
# version
build-level = -20060823
# status
Server is running. TCP port: 3801, SSL port: 443
```

To verify that EKM was using our keystore and our certificates, we issued the **listcerts** command as reflected in Example 6-43.

Note: The **listcerts** command on a production keystore results in a very large display. We shortened the certificate's listing in this display example.

Example 6-43 The listcerts command results

```
# listcerts
Keystore entries: 3
Keystore type:jceks
Keystore provider:IBMJCE

cert1ca, Fri Sep 22 09:07:15 MST 2000, trustedCertEntry
Certificate fingerprint (MD5withRSA):
57:b8:78:50:65:c5:17:e8:7c:66:b8:c1:42:5e:98:78:cf:ec:89:36:2:f3:ff:55:36:82:7:e4:
4:16:54:42:b4:c3:6d:2a:e6:6b:9c:f0:8:15:a4:66:ec:a2:c6:c9:90:c0:24:2b:61:69:3f:ad:
:e:f4:a1:80

cert2, Fri Sep 22 09:06:54 MST 2000, keyEntry
Certificate fingerprint (MD5withRSA):
:9f:37:1a:43:3c:3c:e4:fb:8e:9:d2:11:4b:1c:11:bb:15:70:c4:c6:79:30:40:a4:4e:f2:ce:7
3:3:e3:6d:a1

cert1, Fri Sep 22 09:06:01 MST 2000, keyEntry
Certificate fingerprint (MD5withRSA):
57:b8:78:50:65:c5:17:e8:7c:66:b8:c1:42:5e:98:78:cf:ec:89:36:2:f3:ff:55:36:82:7:e4:
:67:4a:47:70:7:d1:e7:44:c7:e6:f1:62:9:c7:5a:12:14:3f:4f:b3:46:e2:71:f1:79:5f:45:65
:e:f4:a1:80
```

Drive tables, configuration properties, and so forth

EKM carries over changes made to its drive table and configuration.

Prior to testing encryption, we checked the VOLSERs in our logical library to verify that these VOLSERs had not been encrypted:

- # adddrive -drivename 000001365109 -rec1 cert3 -rec2 cert4
- # listdrives
- Drive entries: 1
- SerialNumber = 000001365109

As shown in Figure 6-16 on page 196, VOLSER J1S357 is mounted but has not yet been encrypted.

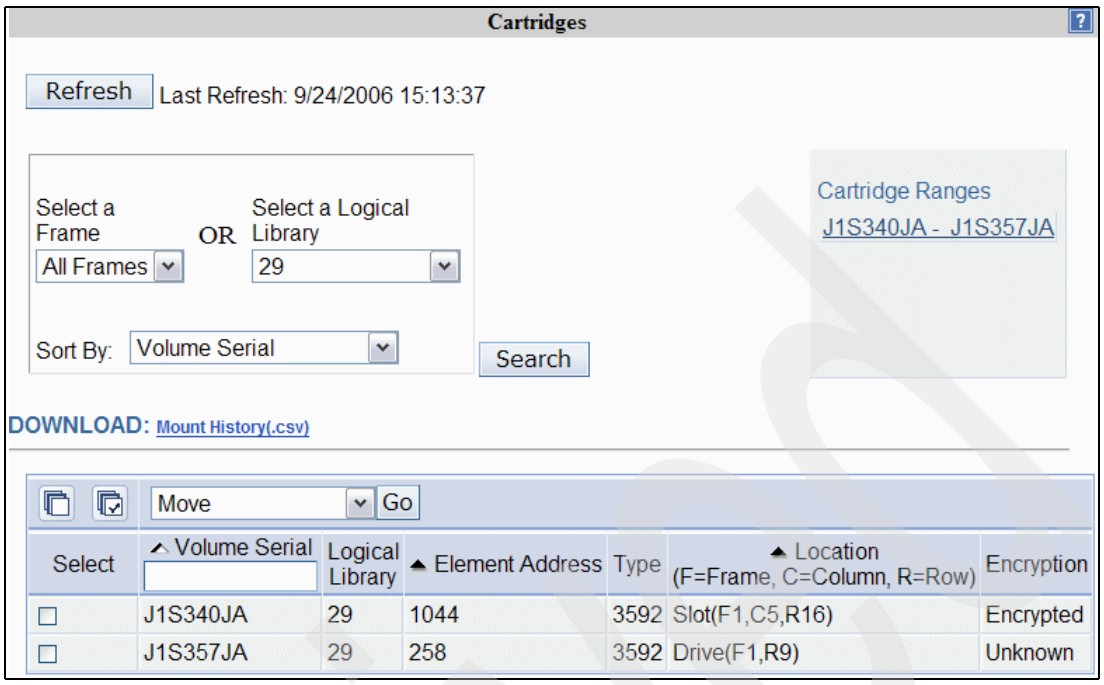


Figure 6-16 J1S357 prior to being encrypted

If, during your testing, you want to reset the 3584 Web GUI encryption indicator for a cartridge, you can do that in one of two ways:

- Change the indicator to NOT ENCRYPTED by ensuring that the cartridge is outside of a Barcode Encryption Policy (BEP) and issuing a write from beginning of tape.
- Change the indicator to UNKNOWN by physically removing the cartridge from the library and then reinserting it.

6.4 Installing the EKM in i5/OS

Refer to the following corresponding sections to either newly install or to upgrade an installed EKM:

- To newly install the *IBM Encryption Key Manager component for the Java platform* (EKM) on i5/OS as a primary or secondary EKM server, refer to 6.4.1, “New installation of the Encryption Key Manager” on page 196
- To upgrade an installed EKM release to a newer service release, refer to 6.4.2, “Upgrading the Encryption Key Manager” on page 199.

6.4.1 New installation of the Encryption Key Manager

For a new installation of IBM EKM component for the Java platform, do the following:

1. Install EKM as a *primary* or single EKM server on an i5/OS server by following the steps in “New installation of a primary EKM Server” on page 197.
2. If you want to set up a *secondary* EKM server on an i5/OS system, follow the steps in “New installation of a secondary EKM Server (optional)” on page 198.

New installation of a primary EKM Server

To install a *primary* EKM server:

1. Refer to 5.2.3, “EKM on IBM System i5 requirements” on page 143 to verify that you have all software prerequisites for either i5/OS V5R3 or V5R4 installed.
2. Install the *unrestricted JCE policy files* `local_policy.jar` and `US_export_policy.jar` Version 1.4.2, which can be downloaded from the IBM Web site:

   `https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk`

   Install the files to the following IFS directories:

   – For V5R4, install to:

   `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/security/`

   – For V5R3, install to:

   `/QIBM/ProdData/Java400/jdk15/lib/security/`

   Note: Make to sure to replace the existing policy files with the unrestricted ones downloaded in the previous step. The unrestricted JCE policy files Version 1.4.2 are the same for Java Version 1.4.2 and 1.5 or 5.0.

3. Edit the `java.security` file to include the following providers if they are not already included:

   – `security.provider.6=com.ibm.jsse2.IBMJSSEProvider2`
   – `security.provider.7=com.ibm.i5os.jsse.JSSEProvider`

   Note: The unique number to be used for adding these security providers depends on which providers are already specified in your `java.security` file.

   The `java.security` file is located in the following IFS directory:

   – For V5R4, the file is in:

   `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/security/`

   – For V5R3, the file is in:

   `/QIBM/ProdData/Java400/jdk15/lib/security/`

4. Create an IFS directory for EKM to hold the EKM keystore, configuration file, drive table, and so forth with a subdirectory for the EKM auditlogs using the i5/OS commands:

   ```
   CRTDIR DIR('/EKM')
   CRTDIR DIR('/EKM/auditlogs')
   ```

5. Copy the default EKM configuration file to the previously created EKM directory to make sure it will not be overwritten by an i5/OS Java software update; use the command:

   ```
   CPY OBJ('/QIBM/ProdData/OS400/Java400/ext/KeyManagerConfig.properties')
   TODIR('/EKM')
   ```

6. Proceed to section 15.2.1, “Creating an EKM keystore and certificate” on page 515 to create a keystore and corresponding certificates for EKM on i5/OS.
7. Complete the EKM configuration for 3592 Tape Encryption and Linear Tape-Open (LTO) 4 or LTO4 Tape Encryption described in the section 6.4.3, “Configuring EKM for tape data encryption” on page 201.
8. Set up the Encryption Key Manager address in the IBM TS3*xxx* library and enable Library-Managed Encryption by referring to section 15.2.2, “Configuring the TS3500 library for Library-Managed Encryption” on page 528.

9. After completing the EKM configuration, submit the EKM server as a batch job by using the command:

   ```
   SBMJOB CMD(QSH CMD('strEKM -server -propfile /EKM/KeyManagerConfig.properties
   1> /EKM/stdout.log 2> /EKM/stderr.log')) JOB(EKMBCH) JOBQ(QSYS/QUSRNOMAX)
   ```

   The *strEKM* script on i5/OS in /usr/bin explicitly refers to the correct Java version for starting the EKM server so there is no further specification required if multiple Java versions are installed on i5/OS.

   Note: Add an autostart job entry for the subsystem in which the EKM server will be running to guarantee that the EKM server is automatically started when the corresponding subsystem gets started, for example, after IPL. To add an autostart job entry, create a job description for the EKM server job and add an autostart job entry to your subsystem using the following job description:

   ```
   CRTJOBD JOBD(library/EKMJOBD) JOBQ(QSYS/QUSRNOMAX) USER(userid)
   RQSDTA('STRQSH CMD(''strEKM -server -propfile
   /EKM/KeyManagerConfig.properties 1> /EKM/stdout.log 2> /EKM/stderr.log'')')

   ADDAJE SBSD(library/subsystem) JOB(EKMBCH) JOBD(library/EKMJOBD)
   ```

10. Back up the EKM keystore, EKM configuration file, drive table, and audit logs without using encrypted saves, that is, transfer to a system that is not using tape encryption for backup.

New installation of a secondary EKM Server (optional)

These steps describe a new installation of an optional *secondary* EKM server on another i5/OS system.

To install and setup an optional *secondary* EKM server:

1. Follow the steps 1 to 4 for the installation of the *primary* EKM server.
2. When using solely LTO4 Tape Encryption, ensure that a public-private key certificate exists in the EKM server's JCEKS keystore for SSL communication to be used for synchronization with the secondary EKM server. Refer to "Creating a JCEKS keystore and certificate" on page 526 to list the certificates in a JCEKS keystore and create a public-private key if needed.
3. Copy the keystore file, such as `EKM.KDB` or `EKM.JCK`, the configuration file `KeyManagerConfig.properties`, and the `drivetable` file from your primary EKM server IFS directory (that is, `/EKM`) to the same directory on your i5/OS system with the secondary EKM server, for example, by using the iSeries Navigator or FTP transfer.
4. On your i5/OS system that is used for the *secondary* EKM server, start the EKM server as a batch job by referring to step 9.
5. Refer to the section "Setting up Encryption Key Manager addresses" on page 528 to set up the secondary EKM server IP address in the tape library.
6. On your i5/OS system that is used for the *primary* EKM server, end the EKM batch job (see step 1 on page 199) and start the EKM admin console from QShell by running the command:

   ```
   strEKM -propfile /EKM/KeyManagerConfig.properties
   ```

Note: Currently, the EKM admin console knows nothing about the EKM server started as batch job, so for any configuration changes to an EKM server started in a batch job, this batch job must be ended prior to the changes. Otherwise, the configuration changes are lost when the batch job is ended and the EKM server writes its current configuration from memory to its file.

7. Use the following commands from the *primary* EKM server's admin console to set up automatic synchronization between the primary and secondary EKM server:

   ```
   modconfig -set -property sync.ipaddr -value 9.11.202.6:443
   modconfig -set -property sync.type -value all
   modconfig -set -property sync.timeinhours -value 24
   ```

   This example sets up automatic synchronization between the primary EKM server and the secondary EKM server, which has the IP address 9.11.202.6, using the EKM default SSL port 443, as specified in the EKM configuration file TransportListener.ssl.port parameter. The specified sync.type parameter value of *all* means that the EKM configuration file, which is rewritten from the primary to the secondary server, *and* the EKM drive table, which is merged by sending new updates from the primary to the secondary server, are synchronized. Synchronization is started every 24 hours as specified by the sync.timeinhours parameter value of *24*.

   To verify your changes in the EKM configuration, run the **listconfig** command.

   For additional information about synchronization of the EKM server, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning and User's Guide*, GA76-0418.

   Note: The EKM admin console **sync** command, for example, `sync -all -ipaddr 9.11.202.6:443`, can be used to manually synchronize or test the synchronization between the primary and secondary EKM server; however, it requires that the primary EKM server be started interactively.

8. Exit from the *primary* EKM admin console by using the command **exit**.
9. Start the *primary* EKM server as a batch job by again referring to step 9 on page 198.

6.4.2 Upgrading the Encryption Key Manager

Use the procedure in this section to upgrade an existing *IBM Encryption Key Manager component for the Java platform* (EKM) installation on i5/OS to a newer EKM service release.

Note: EKM Release 2 is the official IBM service path for EKM Release 1, that is, no new EKM Release 1 maintenance releases will be made available.

To upgrade EKM:

1. Shut down the EKM server:
   - If the EKM server was started as a batch job, use the i5/OS command WRKACTJOB to locate the EKM batch job, which usually runs in the QUSRWRK subsystem, and end it either using the option **4** as shown in Figure 6-17 on page 200 or by using the following command:

     ```
     ENDJOB JOB(EKMBCH)
     ```

Note: Do not use the `*IMMED` option to end the EKM batch job immediately, because this option prevents a proper shutdown of EKM and does not update its configuration file, drive table, and XML metadata file.

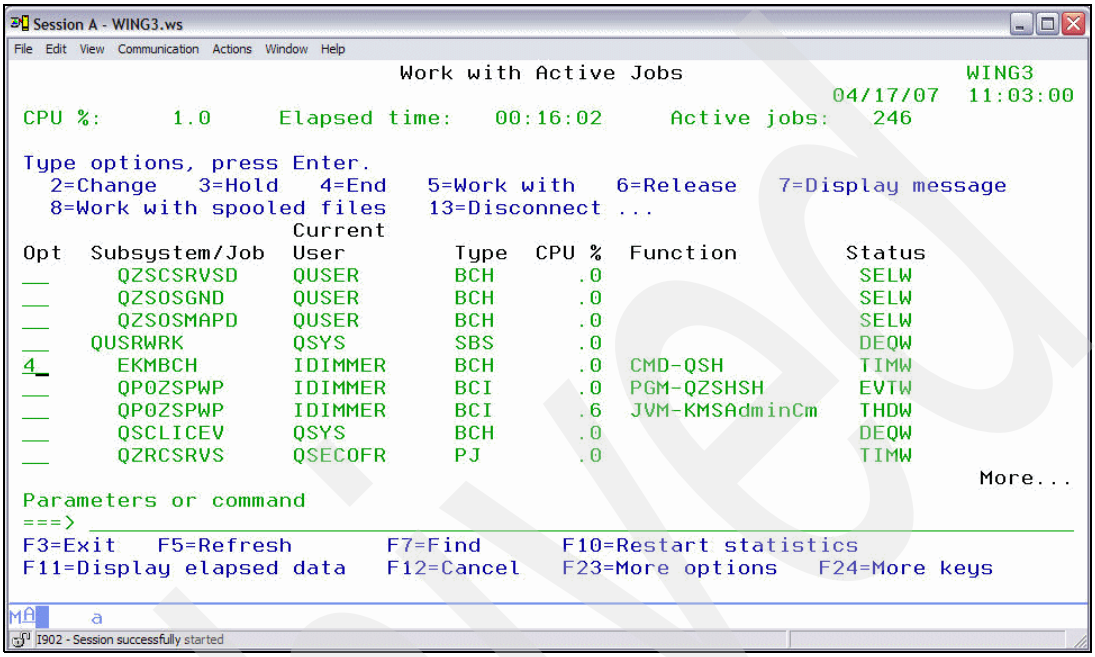


Figure 6-17 i5/OS WRKACTJOB panel showing the EKM batch job

- If the EKM server was started interactively, run the command **`exit`** from the EKM admin console in i5/OS Qshell to stop the EKM server *and* exit from the EKM admin console.

2. Update the EKM code to the new release by replacing the following `IBMKeyManagementServer.jar` Java extension file with its newer version:
   - For i5/OS V5R4:

     `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre/lib/ext/IBMKeyManagementServer.jar`
   - For i5/OS V5R3:

     `/QIBM/ProdData/OS400/Java400/ext/IBMKeyManagementServer.jar`

   Note: Ensure that you *delete* or *overwrite* the old `IBMKeyManagementServer.jar` version. Do *not* rename the file from the old version because Java will still find its class information from the old renamed file and the new EKM code is not used.

3. If upgrading from EKM Release 1 to a newer release, add the required `Audit.metadata.file.name` parameter to the EKM configuration file by referring to step 2 on page 202 (in “Customizing the EKM Configuration File” on page 202).
4. If you will be *newly* using LTO4 Tape Drive Encryption after this EKM upgrade:
   a. Ensure that the required IBM Java 5.0 Service Release 5 is installed with the enhanced *keytool* for support of symmetric key management (refer to 15.1.2, “Software prerequisites” on page 509).
   b. Generate the required symmetric keys in a JCEKS-type EKM keystore by referring to “Symmetric key generation for LTO4 encryption” on page 526.

c. Add the symmetricKeySet parameter to the EKM configuration file and make sure that the keystore file, its type, password, and provider are adjusted if migrating from an IBMi5OSKeyStore to a JCEKS keystore for LTO4 Tape Encryption. Refer to step 3 on page 203, step 4 on page 203, and step 8 on page 204 (all in "Customizing the EKM Configuration File" on page 202).

5. Verify that the upgraded EKM server starts without errors by first starting and stopping it interactively from the i5/OS Qshell by using the following commands and check that the reported build level matches the new version:

```
strEKM -propfile /EKM/KeyManagerConfig.properties
startekm
exit
```

6. Finally, start the newly upgraded EKM server as a batch job by using the command:

```
SBMJOB CMD(QSH CMD('strEKM -server -propfile /EKM/KeyManagerConfig.properties
1> /EKM/stdout.log 2> /EKM/stderr.log')) JOB(EKMBCH) JOBQ(QSYS/QUSRNOMAX)
```

6.4.3 Configuring EKM for tape data encryption

In this section, we describe the EKM configuration for TS1120 and LTO4 Tape Encryption assuming that you have completed steps 1 - 6 in section 6.4.1, "New installation of the Encryption Key Manager" on page 196.

Default EKM configuration file

The default EKM configuration file KeyManagerConfig.properties, which still has to be customized for a specific environment, is shown in Example 6-44.

Example 6-44 Default EKM configuration file

```
# Note that the file is sorted by property name.  EKM shutdown automatically
# reorders the values in the properties file.
Audit.event.outcome = success,failure
Audit.event.types = all
Audit.eventQueue.max = 0
# Need to change the following directory value or create the directories
Audit.handler.file.directory = /EKM/auditlogs
Audit.handler.file.name = ekm_audit.log
Audit.handler.file.size = 10000
# Need to change the following 2 pathnames to the correct pathnames for
# the keystores being used on your system
Admin.ssl.keystore.name = /EKM/EKM.kdb
Admin.ssl.truststore.name = /EKM/EKM.kdb
# Need to change the following pathname value or create the directories
config.drivetable.file.url = FILE:///EKM/drives/drivetable
# Need to change the following pathname to the correct pathname for
# the keystore being used on your system
config.keystore.file = /EKM/EKM.kdb
config.keystore.provider = IBMi5OSJSSEProvider
config.keystore.type = IBMi5OSKeyStore
debug = all
debug.output = simple_file
# Need to change the following pathname value or create the directory
debug.output.file = /EKM/debug.log
# Change this to 'false' if you do not want new tape drives automatically
# added to the EKM drive table
```

```
drive.acceptUnknownDrives = true
fips = Off
TransportListener.ssl.ciphersuites = JSSE_ALL
TransportListener.ssl.clientauthentication = 0
# Need to change the following pathname to the correct pathname for
# the keystore being used on your system
TransportListener.ssl.keystore.name = /EKM/EKM.kdb
TransportListener.ssl.keystore.type = IBMi5OSKeyStore
# Need to specify the ssl port being used on your system
TransportListener.ssl.port = 443
TransportListener.ssl.protocols = TLSv1
# Need to change the following pathname to the correct pathname for
# the keystore being used on your system
TransportListener.ssl.truststore.name = /EKM/EKM.kdb
TransportListener.ssl.truststore.type = IBMi5OSKeyStore
# Need to specify the tcp/ip port being used on your system
TransportListener.tcp.port = 3801
# Need to specify the passwords for the keystores being used
Admin.ssl.keystore.password = kspwd
Admin.ssl.truststore.password = kspwd
config.keystore.password = kspwd
TransportListener.ssl.keystore.password = kspwd
TransportListener.ssl.truststore.password = kspwd
```

Customizing the EKM Configuration File

Use the following i5/OS command to edit the EKM configuration file and customize it for your environment:

```
EDTF STMF('/EKM/KeyManagerConfig.properties')
```

To change the following configuration parameters according to your environment:

1. Adjust the Audit.handler.file.directory parameter value to match your audit log directory that was created in section step 4 on page 197 (in 6.4.1, “New installation of the Encryption Key Manager” on page 196), which must exist. For example:

   ```
   Audit.handler.file.directory = /EKM/auditlogs
   ```

2. Add the Audit.metadata.file.name parameter that is newly supported with EKM Release 2 for the tracking of key usage requests for volume serial numbers in a XML metadata file, which is an abbreviated version of the EKM audit log and especially useful for LTO4 Encryption (see “Example of the EKM audit metadata XML file” on page 553). For example:

   ```
   Audit.metadata.file.name = /EKM/auditlogs/metadata.xml
   ```

   The audit metadata XML file can be queried for specific volume serial numbers or key aliases with the *EKMDataParser* Java tool using the following syntax:

   ```
   EKMDataParser [-filename metadatafile] [-volser VOLSER] [-keyalias keyalias]
   ```

   Note: New XML metadata entries are generated with each key usage request. By default, 100 entries are cached in memory before they are written to the XML metadata file. You can use the optional Audit.metadata.file.cachecount parameter to set the value of the maximum cached entries; however for performance reasons, we do *not* recommend that you turn off caching by setting this parameter to 0 (zero).

3. Modify the values for the following parameters to match your *name, type,* and *the provider of the EKM keystore* if it differs from the default name `/EKM/EKM.kdb`, type `IBMi5OSKeyStore`, and provider `IBMi5OSJSSEProvider`. For example, the name for a JCEKS keystore might be `/EKM/EKM.jck`, the type might be `JCEKS`, and the provider might be `IBMJCE`:

   ```
   Admin.ssl.keystore.name
   Admin.ssl.truststore.name
   config.keystore.file
   config.keystore.provider
   config.keystore.type
   TransportListener.ssl.keystore.name
   TransportListener.ssl.keystore.type
   TransportListener.ssl.truststore.name
   TransportListener.ssl.truststore.type
   ```

   Note: The truststore and keystore are used for optional EKM to EKM server synchronization using SSL communication, not for communication between the EKM and the tape library.

4. Modify the values for the following parameters to match your *password* for the defined keystore:

   ```
   Admin.ssl.keystore.password
   Admin.ssl.truststore.password
   config.keystore.password
   TransportListener.ssl.keystore.password
   TransportListener.ssl.truststore.password
   ```

5. Modify the config.drivetable.file.url parameter value to reflect your preferred location of the EKM drive table, which is used to validate drives by their serial numbers for usage with EKM. For example:

   ```
   config.drivetable.file.url = FILE:///EKM/drivetable
   ```

6. Modify the debug.output.file parameter value to indicate the file used for the output of EKM debug information. For example (default value):

   ```
   debug.output.file = /EKM/debug.log
   ```

7. The parameter drive.acceptUnknownDrives is set to `true` in the sample EKM configuration file so that any new tape drive that contacts EKM through the IBM tape library is automatically added with its serial number to the EKM drive table. The EKM drive table contains the list of valid drives for usage with EKM. If instead, you prefer to control which drives are valid for usage with EKM, you can set this parameter to its default value of `false` so that new drives must be manually added to the drive table using the **`adddrive`** command from the EKM admin console.

   When the default parameter value of `true` is used with 3592 Tape Encryption, you *must* also set two default key aliases for the drives by adding the `drive.default.alias1` and `drive.default.alias2` parameters to make sure that an EEDK1 and EEDK2 can be generated for the new drive. For example:

   ```
   drive.default.alias1 = Tape_Certificate
   drive.default.alias2 = Tape_Certificate2
   ```

Note: If you only use one public key/certificate for 3592 Tape Encryption because sharing tapes with a business partner is currently not an issue, still make sure that if you use default aliases that *both* alias1 and alias2 are defined even if they are both set to the same key label. Otherwise, the IBM library EKM configuration test might fail. The *default key aliases* can still be overruled by explicit key aliases specified with the rec1 and rec2 parameters for a drive in the drive table or by the IBM TS3500 Tape Library Barcode Encryption Policy.

8. For *LTO4 encryption,* add the parameter symmetricKeySet that specifies that all symmetric keys are to be used for LTO4 Tape Encryption; single key aliases and aliasranges can both be specified at one time and delimited by commas. For example:

   ```
   symmetricKeySet = AES00-0F
   ```

9. Save the changed EKM configuration file `KeyManagerConfig.properties`.

Additional information

For additional information about the EKM configuration file parameters and the EKM admin console command line interface, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418, available at:

`http://www-1.ibm.com/support/docview.wss?rs=1139&context=STCXRGL&dc=D400&uid=ssg1S4000504`

6.5 LTO4 Encryption implementation

When setting up encryption for LTO4 drives and media, unless you are using Application-Managed Encryption (AME), you will have to select a key manager. For a complete discussion of the two key management solutions see 6.6, "LTO4 Library-Managed Encryption implementation" on page 217 or 6.7, "LTO4 System-Managed Encryption implementation" on page 225. At this time, if you want to run your key manager on System z your only option is to use the EKM version 2 or later. For other key server platforms, you may use TKLM. Your key server can run on any supported platform independent of the host systems you are using to access the tape drive.

Advanced Library Management System effect on encryption

If your TS3500 does not have the Advanced Library Management System (ALMS) feature, be aware that your ability to implement and manage encryption will be much more inflexible than if you have ALMS installed. If you intend to implement encryption on an existing library without ALMS, older technology cannot coexist with newer encryption-capable technology. If the non-ALMS library consists entirely of LTO4 technology, all logical partitions will have to be managed using the same encryption method, that is, all LME or all SME, and so forth.

Also be aware that many new features of the TS3500 such as High Density Frames, Tape System Reporter, the TS1130 Tape Drive and embedded SMI-S support are only available when the ALMS feature is installed and enabled.

Refer to 14.2.1, "Advanced Library Management System considerations" on page 451 for more details.

6.5.1 LTO4 EKM implementation checklist

This checklist is what we used to implement encryption using LTO4 drives located within a TS3500 library with EKM V2 running on a Windows system. We show the steps necessary for installing EKM V2 as well as how to implement encryption using LME and SME. The basic steps are:

1. Download the latest version of EKM software.

   Download unrestricted policy files to enable AES key generation by EKM.

2. Create a JCEKS Keystore.

   Generate encryption keys.

3. Start the EKM Admin Session.

4. Start EKM.

In this section, we describe how to enable LME on a TS3500 for LTO4 drives. For our testing purposes, we installed EKM on our personal computer under Windows. We already had Java 1.5.0 installed as shown in Example 6-45.

Example 6-45 Java version already installed

```
C:\Documents and Settings\Administrator>java -version
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pwi32dev-20070201 (SR4))

IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 Windows XP x86-32 j9vmwi3223-2007020
1 (JIT enabled)
J9VM - 20070131_11312_lHdSMR
JIT  - 20070109_1805ifx1_r8
GC   - 200701_09)
JCL  - 20070131
```

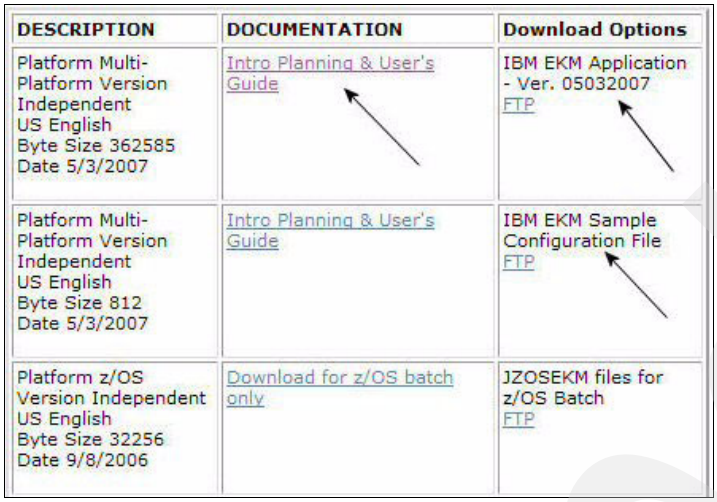
Ensure that a minimum of Software Developer Kit (SDK) 1.4.2 SR8 or SDK 5.0 SR5 is installed. If you have an earlier SDK, refer to *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418, to learn how to get the latest service refresh for your software platform.

6.5.2 Download the latest EKM software

Download the EKM JAR and verify the directory:

1. Download the latest version of the EKM JAR file (`IBMKeyManagementServer.jar`) and configuration files (`KeyManagerConfig.properties`) from the following Web site (also shown in Figure 6-18 on page 206):

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



DESCRIPTION	DOCUMENTATION	Download Options
Platform Multi-Platform Version Independent US English Byte Size 362585 Date 5/3/2007	Intro Planning & User's Guide	IBM EKM Application - Ver. 05032007 FTP
Platform Multi-Platform Version Independent US English Byte Size 812 Date 5/3/2007	Intro Planning & User's Guide	IBM EKM Sample Configuration File FTP
Platform z/OS Version Independent US English Byte Size 32256 Date 9/8/2006	Download for z/OS batch only	JZOSEKM files for z/OS Batch FTP

Figure 6-18 EKM Web site

2. In addition to the EKM code and sample EKM configuration, download the latest version of the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418.
3. Determine whether there is a copy of the IBMKeyManagementServer.jar file in the <JAVA_INSTALL>/lib/ext directory. If so, *delete* it and copy in the version just downloaded. Example 6-46 displays the directory after we copied in the newest version of the IBMKeyManagementServer.jar to the correct directory. (We downloaded the JAR file in step 1.)

Example 6-46 Our Java directory

```
Directory of C:\Program Files\IBM\Java50\jre\lib\ext
07/01/2007  12:15 PM    <DIR>          .
07/01/2007  12:15 PM    <DIR>          ..
02/01/2007  05:28 AM           183,719 CmpCrmf.jar
02/01/2007  05:28 AM            15,621 dtfj-interface.jar
02/01/2007  05:28 AM           201,824 dtfj.jar
02/01/2007  05:28 AM         1,110,163 gskikm.jar
02/01/2007  05:28 AM           179,050 ibmcmsprovider.jar
02/01/2007  05:28 AM           186,317 ibmjcefips.jar
04/11/2007  03:55 PM           860,283 ibmjceprovider.jar
02/01/2007  05:28 AM           213,991 ibmkeycert.jar
07/01/2007  12:13 PM           362,585 IBMKeyManagementServer.jar
02/01/2007  05:28 AM            82,640 ibmpkcs11.jar
02/01/2007  05:28 AM           253,282 ibmpkcs11impl.jar
02/01/2007  05:28 AM            64,506 ibmsaslprovider.jar
02/01/2007  05:28 AM            65,709 indicim.jar
02/01/2007  05:28 AM            50,129 jaccess.jar
02/01/2007  05:28 AM            15,661 JawBridge.jar
02/01/2007  05:28 AM           241,618 jdmpview.jar
              16 File(s)      4,087,098 bytes
               2 Dir(s)  20,393,205,760 bytes free
```

Download unrestricted policy files to enable AES keys

To enable AES keys:

1. Open a command window and create an EKM directory where all of the EKM-related files will be stored. On Windows, use the **mkdir c:\ekm\ekm1** command; the results of which are shown in Example 6-47.

Example 6-47 Windows EKM directory

```
C:\ekm>dir
 Volume in drive C has no label.
 Volume Serial Number is 6806-ABBD

 Directory of C:\ekm

07/11/2007  01:14 PM    <DIR>          .
07/11/2007  01:14 PM    <DIR>          ..
07/11/2007  01:14 PM    <DIR>          ekm1
               0 File(s)              0 bytes
```

2. Copy the sample KeyManagerConfig.properties file to the EKM1 directory. In our example, on Windows, use the following command:

```
copy KeyManagerConfig.properties c:\ekm\ekm1\KeyManagerConfig.properties
```

3. Because of governmental export regulations, JDKs are set up with the ability to do limited function cryptography. For full function cryptography, you must install the unrestricted policy files. Download the `US_export_policy.jar` and `local_policy.jar` files from:

   https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk

After identifying yourself by signing in at the Web site, the panel shown in Figure 6-19 on page 208 appears.

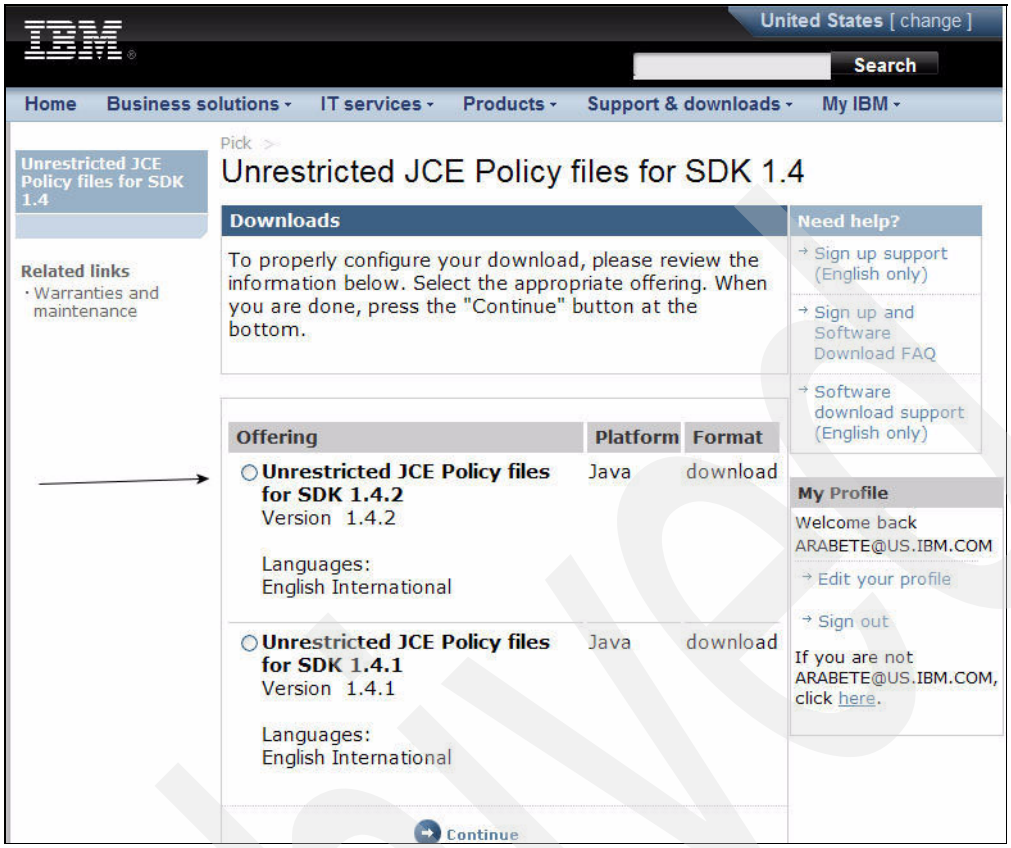


Figure 6-19 Web site for downloading the unrestricted policy files

Be sure to select **Unrestricted JCE Policy files for SDK 1.4.2**, which works for both Java 1.4.2 and Java 5.0 SDKs.

Important: Do not rename the JAR files, replace them. Do not select the 1.4.1 version, because it is incompatible with Java 1.4.2 or Java 5.0 SDKs.

Replace the files in your <*JAVA_INSTALL*>/lib/security directory. These are the unrestricted policy files that EKM requires in order to serve AES keys. Example 6-48 shows how our directory looked after replacing the two files.

Example 6-48 Directory after replacing the two JARs

```
Directory of C:\Program Files\IBM\Java50\jre\lib\security

07/01/2007  12:25 PM    <DIR>          .
07/01/2007  12:25 PM    <DIR>          ..
02/01/2007  05:28 AM            40,624 cacerts
02/01/2007  05:28 AM             2,706 java.policy
02/01/2007  05:28 AM             9,864 java.security
02/01/2007  05:28 AM               568 javaws.policy
05/12/2004  04:41 PM             2,212 local_policy.jar
05/12/2004  04:41 PM             2,199 US_export_policy.jar
               6 File(s)         58,173 bytes
```

6.5.3 Create a JCEKS keystore

EKM uses standard and operating system-specific Java keystore methods to store the public-private key and certificate information. Although EKM supports six keystore types, currently only the JCEKS keystore type supports both symmetric (LTO4) and asymmetric (3592) keys. We use a JCEKS keystore in our examples.

In order for EKM to operate correctly, it requires a keystore with a certificate and a private key. The **keytool** command in Example 6-49 creates a new JCEKS keystore called `mykeystore.jck` and populates it with a certificate and a private key with the alias of myprivcert.

This command prompts you for a password to access the keystore.

Note: Remember the keystore password you enter here, because you will need it later when you start EKM. When prompted for a *key* password, press Enter. Do not enter a new or different password.

Example 6-49 Create private certificate

```
C:\ekm\ekm1>keytool -keystore mykeystore.jck -storetype jceks -genkey -alias myp
rivcert -keyalg RSA -keysize 2048
Enter keystore password:  mykeystorepword
What is your first and last name?
  [Unknown]:  key administrator
What is the name of your organizational unit?
  [Unknown]:  tape encryption
What is the name of your organization?
  [Unknown]:  IBM
What is the name of your City or Locality?
  [Unknown]:  Tucson
What is the name of your State or Province?
  [Unknown]:  Arizona
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=administrator, OU=tape encryption, O=IBM, L=Tucson, ST=Arizona, C=US corre
ct? (type "yes" or "no")
  [no]:  yes

Enter key password for <myprivcert>
        (RETURN if same as keystore password):

C:\ekm\ekm1>
```

At this time, if you issue the following command, you receive the display shown in Example 6-50 on page 210:

```
keytool -ekmhelp
```

Example 6-50 keytool -ekmhelp display

```
C:\ekm\ekm1>keytool -ekmhelp
-exportseckey      [-v]
             [-alias <alias> | aliasrange <aliasRange>] [-keyalias <keyalias>]
             [-keystore <keystore>] [-storepass <storepass>]
             [-storetype <storetype>] [-providerName <name>]
             [-exportfile <exportfile>]

-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>]

-importseckey       [-v]
             [-keyalias <keyalias>] [-keypass <keypass>]
             [-keystore <keystore>] [-storepass <storepass>]
             [-storetype <storetype>] [-providerName <name>]
             [-importfile <importfile>]
```

Important: If implementing both LTO4 and 3592 encryption, ensure that the keystore that you select supports both symmetric (LTO4) and asymmetric (3592) keys. At this time, only the JCEKS keystore supports both symmetric and asymmetric key types.

As you can see from Example 6-50, the Keytool utility has been enhanced to generate aliases and symmetric keys for encryption on LTO Ultrium 4 tape drives using LTO4 tape. Use the **keytool -genseckey** command to generate one or more secret keys (genseckey = generate secret keys) and to store them in a specified keystore. Most of the **keytool -genseckey** parameters are obvious, so we describe only the parameters that merit more interest. The two parameters of interest for the **-genseckey** command for LTO4 are:

- `-alias`

  The `alias` parameter generates a single data key. Specify an *alias* value for a single data key with *up to* 12 printable characters (for example, `lto`, `abcfrg`, or `ibm123tape`).

- `-aliasrange`

  The `-aliasrange` parameter generates multiple data keys; this parameter specifies both the number of data keys to generate and the labels to assign to each key.

The `-aliasrange` parameter is specified as a three-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 that are 21 characters in length. For example:

- Specifying `ekm1-a` yields a series of aliases from `EKM000000000000000001` through `EKM00000000000000000A`.
- Specifying an `aliasrange` value of `xyz01-FF` yields a series of aliases from `XYZ000000000000000001` through `XYZ0000000000000000FF`.

Examples of acceptable aliases that can be associated with symmetric keys are:

`abcfrg`	Acceptable for a single key
`ibm123tape`	Acceptable for a single key

abc000000000000000001	Acceptable for range specification
LTO000000000000000001	Acceptable and our favorite
abc00a0120fa000000001	Acceptable for a single key

Examples of aliases that are not accepted by the EKM are:

abcefghij1234567	Greater than 12 characters for a single key
abcg0000000000000001	The range specification is acceptable because it is less than the maximum number of characters allowed; however, the prefix (abcg) is longer than three characters.

If an alias already exists in the keystore, the keytool program issues an exception message and stops.

The **-genseckey** command parameter that we used is shown in Example 6-51. In this example, note the number of characters needed for the keystore password. We cannot use pword, because the password must be at least six characters, so instead we used mykeystorepword as our keystore password. Note that we are using the same password for the keys. We specified a range of aliases by specifying lto00-1f. This generates 32 individual data keys: the first data key starts with the label lto000000000000000001 (21 characters) and the last data key ends with the label lto000000000000001f (also 21 characters).

Example 6-51 keytool -genseckey parameters

```
C:\ekm\ekm1>keytool -keystore mykeystore.jck -storetype jceks -genseckey -keyalg
 aes -keysize 256 -aliasrange lto00-1f
Enter keystore password:  pword
Keystore password is too short - must be at least 6 characters
Enter keystore password:  mykeystorepword
Enter key password for <keys>
        (RETURN if same as keystore password):
KeyTool is generating batch keys. This process will take a while, be patient ...

32 secret keys have been generated
```

Let us follow up the **-genseckey** command with a list of the keystore entries, which is shown in Example 6-52. Note that we had to specify the keystore password in the command, so if you decide to create a bat file to do this regularly, you must protect the bat file, because it contains your keystore password.

Example 6-52 keytool -list command

```
C:\ekm\ekm1>keytool -list -keystore mykeystore.jck -storepass mykeystorepword
-storetype jceks

Keystore type: jceks
Keystore provider: IBMJCE

Your keystore contains 33 entries

lto000000000000000009, Jul 16, 2007, SecretKeyEntry,
lto000000000000000008, Jul 16, 2007, SecretKeyEntry,
lto000000000000000007, Jul 16, 2007, SecretKeyEntry,
lto00000000000000000f, Jul 16, 2007, SecretKeyEntry,
lto000000000000000006, Jul 16, 2007, SecretKeyEntry,
```

```
lto00000000000000000e, Jul 16, 2007, SecretKeyEntry,
lto000000000000000005, Jul 16, 2007, SecretKeyEntry,
lto00000000000000000d, Jul 16, 2007, SecretKeyEntry,
lto000000000000000004, Jul 16, 2007, SecretKeyEntry,
lto00000000000000000c, Jul 16, 2007, SecretKeyEntry,
lto000000000000000003, Jul 16, 2007, SecretKeyEntry,
lto00000000000000000b, Jul 16, 2007, SecretKeyEntry,
lto000000000000000002, Jul 16, 2007, SecretKeyEntry,
lto00000000000000000a, Jul 16, 2007, SecretKeyEntry,
lto000000000000000001, Jul 16, 2007, SecretKeyEntry,
lto000000000000000000, Jul 16, 2007, SecretKeyEntry,
lto000000000000000019, Jul 16, 2007, SecretKeyEntry,
lto000000000000000018, Jul 16, 2007, SecretKeyEntry,
lto00000000000000001f, Jul 16, 2007, SecretKeyEntry,
lto000000000000000017, Jul 16, 2007, SecretKeyEntry,
lto00000000000000001e, Jul 16, 2007, SecretKeyEntry,
lto000000000000000016, Jul 16, 2007, SecretKeyEntry,
lto00000000000000001d, Jul 16, 2007, SecretKeyEntry,
lto000000000000000015, Jul 16, 2007, SecretKeyEntry,
myprivcert, Jul 17, 2007, keyEntry,
Certificate fingerprint (MD5): 5D:F7:A5:1F:27:47:23:17:C9:13:56:8F:34:53:57:BB
lto00000000000000001c, Jul 16, 2007, SecretKeyEntry,
lto000000000000000014, Jul 16, 2007, SecretKeyEntry,
lto00000000000000001b, Jul 16, 2007, SecretKeyEntry,
lto000000000000000013, Jul 16, 2007, SecretKeyEntry,
lto00000000000000001a, Jul 16, 2007, SecretKeyEntry,
lto000000000000000012, Jul 16, 2007, SecretKeyEntry,
lto000000000000000011, Jul 16, 2007, SecretKeyEntry,
lto000000000000000010, Jul 16, 2007, SecretKeyEntry,

C:\ekm\ekm1>
```

6.5.4 Off-site or business partner exchange with LTO4 compared to 3592

In 2.1.2, “Encryption keys and 3592 and LTO4 differences” on page 27, we explained a few basic differences between LTO4 and 3592 encryption. In this section, we investigate how off-site or business partner (BP) exchanges of encrypted tapes are accomplished.

The 3592 encryption makes cartridge exchange easy, because the Data Key (DK) that was used to encrypt the cartridge is included on each cartridge. The encrypted DK on the cartridge is protected by using the public key of the intended recipient of the tape cartridge.

The cartridge recipient simply needs to use their private key to unlock or decrypt the DK that was used to encrypt the user data on the cartridge.

So how do you handle off-site or BP exchange with LTO4, because the DK is not included on the data cartridge? Obviously, you have to provide the recipient with the DK that was used to encrypt the data. You can do this in one of two ways:

- Use a specific key or keys that are already known to the recipient
- Send the DK that was used to encrypt the cartridge data to the recipient

Using a specific key or keys that are already known

Clearly, the easiest way to exchange encrypted cartridges is to ensure that you encrypt the cartridge using a key already known by the recipient. This involves importing the recipient’s

key prior to cartridge creation, and using a BEP, a ILEP, or even SME to use the imported data key to encrypt the data that is going off-site.

Sending the DK that was used to encrypt the cartridge data

Another method is to send the DK to the recipient after cartridge is created. You need to identify the DK that was used to encrypt the cartridge so that you share the correct DK with the encrypted cartridge recipient.

Regardless of which method you select, you still must decide how to securely transport the DK either from or to the cartridge recipient.

Keytool has been enhanced with two commands to handle this dilemma: **-exportseckey** and **-importseckey**. We show examples of the command and its parameters in Example 6-50 on page 210. The commands have been designed to use public-private cryptography to ensure secure key transportation. Each command has a parameter called **-keyalias**. When importing keys, **-keyalias** specifies the alias of a private key in the keystore that will be used to unlock or encrypt the data being imported. When exporting keys, **-keyalias** specifies the alias of a public key that will be used to encrypt the key or keys that are being exported with the intent that the export recipient will have the correct private key that will be needed to decrypt the data keys.

Using our keystore as the source, we use **keytool -exportseckey** as shown in Example 6-53 to export the `lto000000000000000001` data key. The command will use the private key located within `myprivcert` to encrypt or wrap the `lto(shortened)01` DK so that transport security is not an issue. The recipient of the file keytobp will use the **keytool -importseckey** and our public key to unwrap or decrypt the package, yielding the datakey `lto000000000000000001`.

Example 6-53 export example

```
C:\ekm\ekm1>keytool -exportseckey -v -alias lto000000000000000001 -keyalias mypr
ivcert -keystore mykeystore.jck -storepass mykeystorepword -storetype jceks -exp
ortfile keytobp
1 secret keys have been successfully exported

C:\ekm\ekm1>
```

6.5.5 EKM Version 2 installation and customization on Windows

In Example 6-54, you see the EKM/EKM1 directory structure that we created. At this point, we are ready to customize the `keymanagerconfig.properties` file that we downloaded earlier.

Example 6-54 Our EKM directory

```
C:\ekm\ekm1>dir
 Volume in drive C has no label.
 Volume Serial Number is 6806-ABBD

 Directory of C:\ekm\ekm1

07/17/2007  07:39 PM    <DIR>          .
07/17/2007  07:39 PM    <DIR>          ..
07/06/2007  04:56 PM               946 KeyManagerConfig.properties
07/16/2007  09:45 AM            12,224 mykeystore.jck
07/01/2007  04:22 PM                63 sekm.bat
               3 File(s)         13,233 bytes
```

Example 6-55 is what our KeyManagerConfig.properties file looked like after making the minimum number of changes. The changes that we made to the original file are highlighted.

Example 6-55 Our EKM config after minimal changes

```
C:\ekm\ekm1>type KeyManagerConfig.properties
TransportListener.ssl.port = 1443
TransportListener.tcp.port = 3801
TransportListener.tcp.timeout = 0
Admin.ssl.keystore.name = mykeystore.jck
config.keystore.password = mykeystorepword
TransportListener.ssl.clientauthentication = 0
TransportListener.ssl.ciphersuites = JSSE_ALL
Audit.handler.file.size = 10000
drive.acceptUnknownDrives = true
Audit.metadata.file.name = metadata/EKMData.xml
TransportListener.ssl.truststore.name = mykeystore.jck
Audit.handler.file.directory = logs
TransportListener.ssl.protocols = SSL_TLS
config.keystore.file = mykeystore.jck
TransportListener.ssl.keystore.name = mykeystore.jck
TransportListener.ssl.keystore.password = mykeystorepword
Audit.eventQueue.max = 0
Audit.handler.file.name = kms_audit.log
Audit.event.outcome = success,failure
Audit.event.types = all
symmetricKeySet = 1to0000-1f
config.drivetable.file.url = FILE:filedrive.table
Admin.ssl.truststore.name = mykeystore.jck
Admin.ssl.keystore.password = mykeystorepword

C:\ekm\ekm1>
```

After generating keys and aliases, be sure to update the symmetricKeySet property in the KeyManagerConfig.properties file to specify the new alias or range of aliases and the filename under which the symmetric keys are stored. 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, EKM does not start and an audit record is created.

The management of EKM keys is expected to be performed using existing keystore management utilities and manual synchronization (that is, extract/export, send, receive, or import/insert) of the keys into the keystore that is used by the EKM. Note that with this feature or capability, the names (key IDs and key aliases or labels) of the symmetric keys are much more apparent to the EKM administrators. The key IDs are not meant to be private or sensitive information.

The expected administrative steps to populating a keystore are:

1. Import a certificate and private key for EKM-to-EKM communications.
2. Generate a set of symmetric encryption keys.

For each EKM that you employ, change the EKM configuration to refer to the key aliases and ranges of the newly created keys by using the new config environment variable, *symmetricKeySet*.

Although the symmetricKeySet parameter that we used for our EKM reflects a single range of keys, the parameter can also specify a value for a single keyalias as shown in Example 6-56 where we specify, in addition to the keyrange `LTO0000-1f`, two single key aliases `redbook123` and `IBMtape01`

Example 6-56 SymmetricKeySet

```
symmetricKeySet = lto0000-1f, redbook123, IBMtape01
```

6.5.6 Start EKM

Now, you can start the EKM. Starting the EKM is a two-step process:

1. Start the EKM Admin Console. Because we have our Windows path statement set correctly to find the correct version of Java, we issue the command from the EKM directory that we created earlier. When it starts, EKM creates any additional files that it needs within this directory. The command that we used to start the EKM Admin Console is shown in Example 6-57. Because the command to start the EKM Admin console is long, we created a `SEKM.BAT` file.

   Example 6-57 Starting EKM Admin Console

   ```
   C:\ekm\ekm1>sekm

   C:\ekm\ekm1>java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig.properties
   #
   ```

2. Start the EKM server by issuing the **`startekm`** command, which results in Example 6-58.

   Example 6-58 Starting the EKM server with the startekm command

   ```
   # startekm
   Loaded drive key store successfully
   Starting the Encryption Key Manager 2.0-20070503
   Processing Arguments
   Processing
   Server is started
   #
   ```

Figure 6-20 shows our EKM directory after the server has been started.

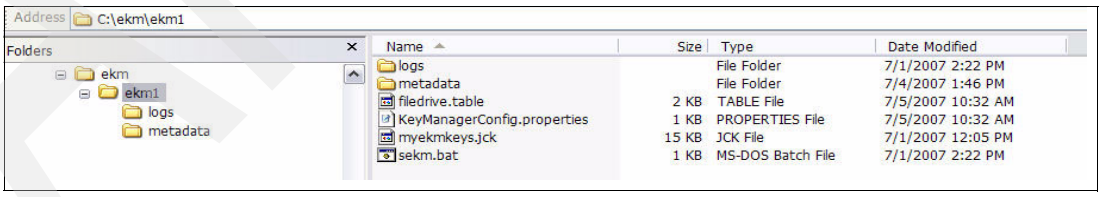


Figure 6-20 Our EKM directory after the server has been started

6.5.7 Starting EKM as a windows Service

On Windows, EKM 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, manually download the binaries for `LaunchEKMService.exe` from the EKM Web site at:

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

Refer to the `readme` file on this Web site for instructions.

Set the system variables JAVA_HOME and PATH:

1. Create a system variable called JAVA_HOME:
   a. From the Start menu, select Control Panel.
   b. Double-click System.
   c. Click the Advanced tab.
   d. Click Environment Variables.
   e. Under the list of System Variables click New.
   f. Specify JAVA_HOME as the variable name and enter the IBM JVM directory, for example `C:\ibm-sdk1.4.2`.
   g. Click OK.
2. Edit the system PATH variable using the following procedure. Setting the PATH variable from the command line will not work.
   a. From the Start menu, select Control Panel.
   b. Double-click System.
   c. Click the Advanced tab.
   d. Click Environment Variables.
   e. Scroll the list of System Variables for the Path variable and click Edit.
   f. Add the IBM JVM to the beginning of the Path variable and click OK.

Note: You must start the EKM Windows Service manually the first time it is used by using the control panel.

The **`LaunchEKMService.exe`** command has the following format (see Example 6-59):

```
LaunchEKMService [-help | -i config_file | -u] -help
```

The options include:

`-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. After the service is installed, you can start and stop the EKM Windows Service from Control Panel. The configuration file has to be specified with this option. If the configuration file does not have all keystore passwords specified, you are prompted for them. When the Windows Service is started, all the passwords are 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.

Example 6-59 The LaunchEKMService command example

```
LaunchEKMService -i KeyManagerConfig.properties
```

Be sure to specify properties with full path names in the `KeyManagerConfig.properties` file if EKM will be installed and used as a Windows Service. After EKM 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 EKM Windows Service runs to a network user. By default, the EKM Windows Service is created to run under the LocalSystem use, which has no access to the network.

To make the 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 the Log On tab.
5. Select this account.
6. Enter the 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 EKMServer properties.

The EKM Windows Service starts successfully.

Tip: The registry keys that the EKM uses as properties when it starts as a service are located in:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EKMServer\Parameters
```

Example 6-60 shows an example of the **EKMDataParser** command.

Example 6-60 EKMDataParser metadata report command

```
C:\eekm\ekm1\metadata>java com.ibm.keymanager.tools.EKMDataParser -filename
EKMData.xml -volser ATS015 >file1
```

Figure 6-21 shows the metadata file report from the **EKMDataParser** command that was issued in Example 6-60.

```
keyAlias1              keyAlias2      volSer  dateTime                      driveSSN      dki
-------------------------------------------------------------------------------------------------------------------

lto000000000000000001                 ATS015  Wed Jul 04 12:35:43  2007  100001350170  6C746F000000000000000001
```

Figure 6-21 Metadata file report from EKMDataParser

6.6 LTO4 Library-Managed Encryption implementation

In this section, we describe how to implement Library-Managed Encryption (LME) on a TS3500 with LTO4 drives to illustrate the differences between 3592 and LTO4. We use the EKM and keystore that we created and installed in the preceding sections.

6.6.1 Barcode Encryption Policy

You can implement LME on the TS3500 in one of two methods: a Barcode Encryption Policy (BEP) or an Internal Library Encryption Policy (ILEP). We provide detail about the BEP.

The logical library that we use is called LTO4 4Gb Fibre and is shown as the last logical library in the list in Figure 6-22. Note that in the encryption method column, our logical library reflects *none*.

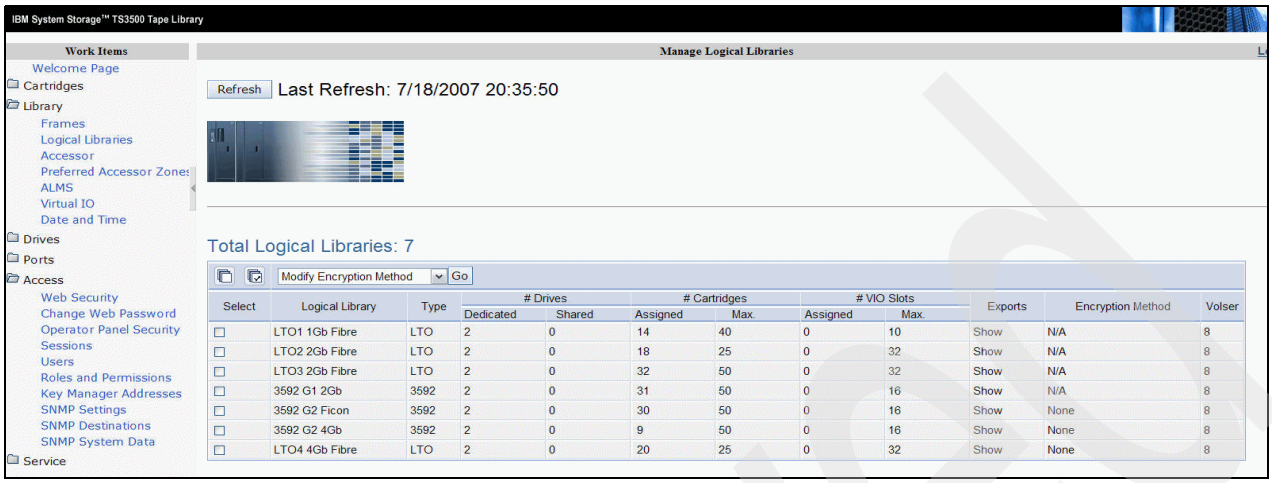


Figure 6-22 Our LTO4 logical library prior to any changes

Note that our logical library, LTO4 4Gb Fibre, has two drives and 20 cartridges assigned to it.

To view the encryption method for your library, on the left, select **Library** → **Logical Libraries** and press Enter.

To change the encryption method for your library from the pull-down list:

1. Refer to Figure 6-23. Select **Modify Encryption Method** and press Enter.

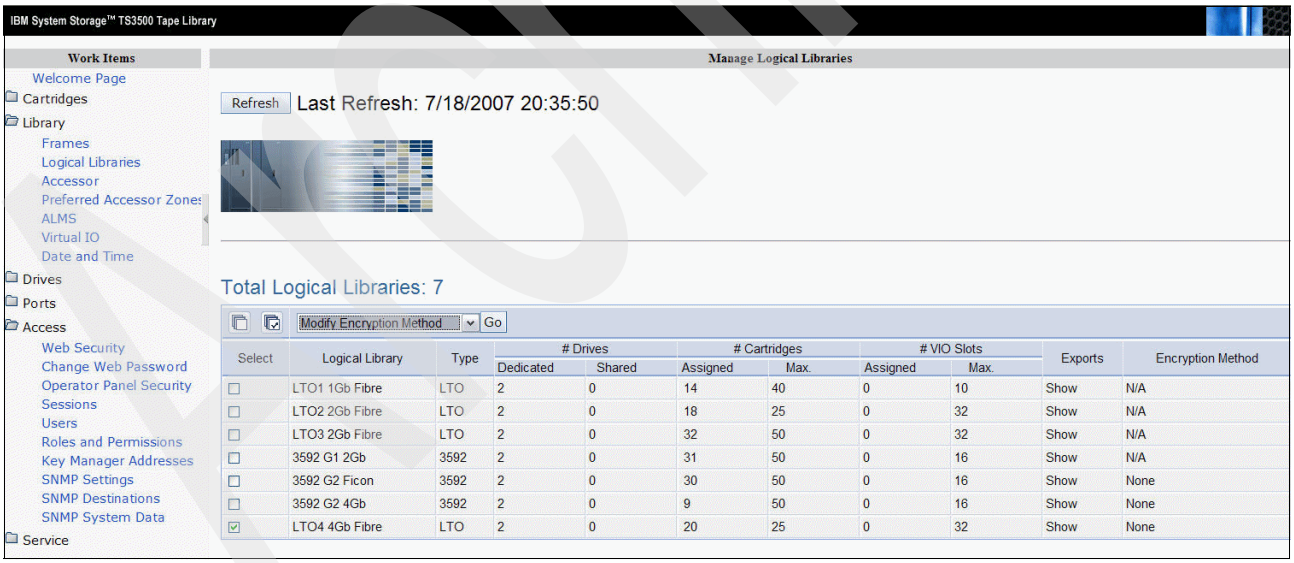


Figure 6-23 Modify the encryption method for a logical library

2. Refer to Figure 6-24 on page 219. From the pull-down list, you choose the Encryption Method, select **Library-Managed** and then select **Apply**. Note that from the TS3500, you can enable encryption for all three encryption methods: LME, SME, or AME. This TS3500 function eliminates the need for the IBM SSR to enable encryption on tape drives that are encryption-capable but not encryption-enabled.

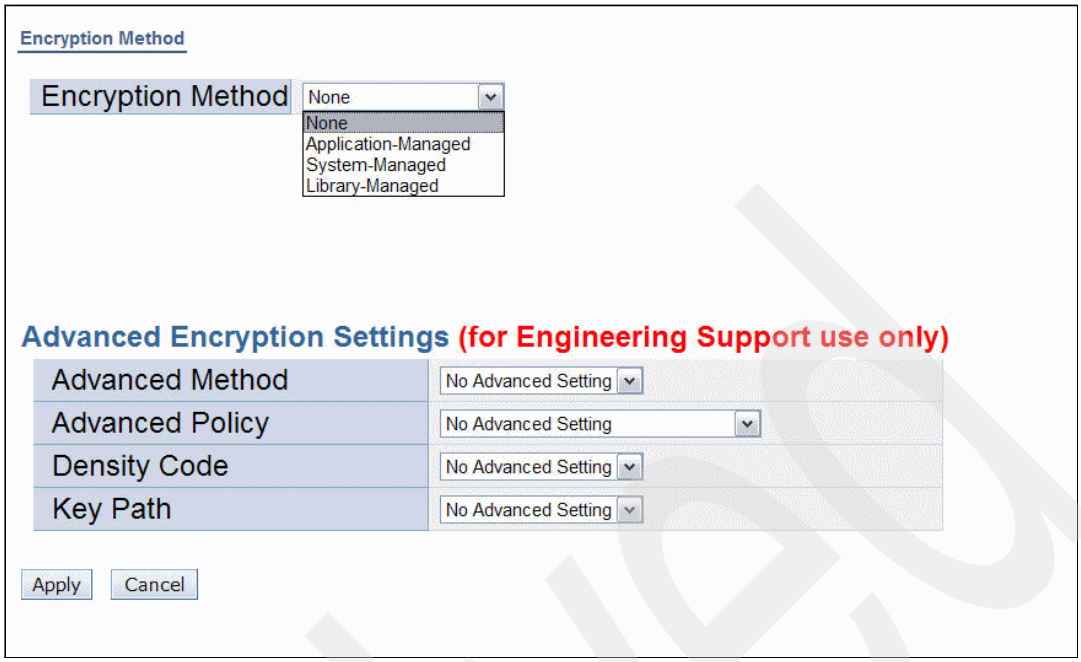


Figure 6-24 Setting the encryption method

3. Refer to Figure 6-25. After selecting **Apply** to enable Library-Managed Encryption, you then select the scratch encryption policy to implement. Select **Barcode (Default)**, and then click **Apply**.

Figure 6-25 Setting the scratch encryption policy

4. Take note of the warning shown in Figure 6-26 on page 220. You might have to restart your operating system so it can rediscover your tape devices. We did not have to do this in our testing; however, you might receive the message shown.

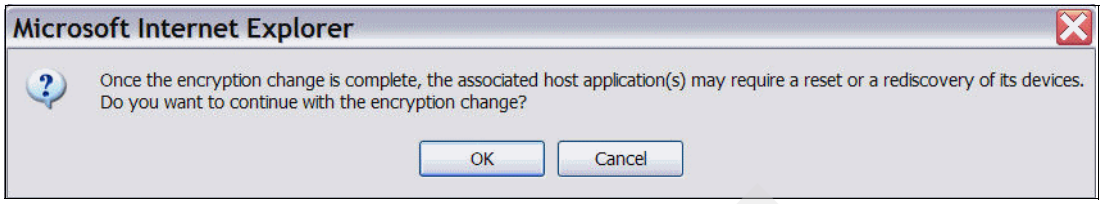


Figure 6-26 Rediscovery warning

5. If you receive that message, click **OK**. Then, you see the panel shown in Figure 6-27, which reflects that the logical library encryption setting change request has completed.

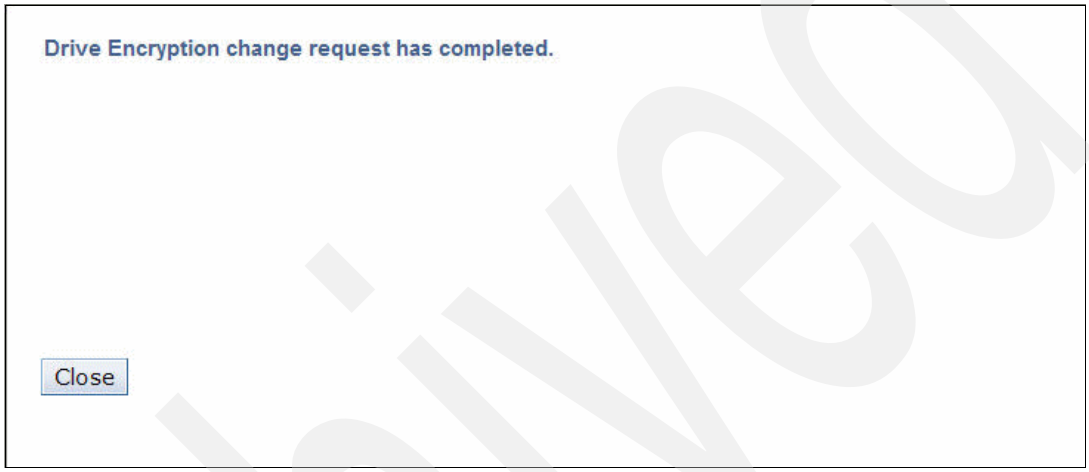


Figure 6-27 Logical library encryption change completed

Taking another look at our logical library, as shown in Figure 6-28, note that it is now configured to use Library-Managed Encryption by using Barcode Encryption Policies.

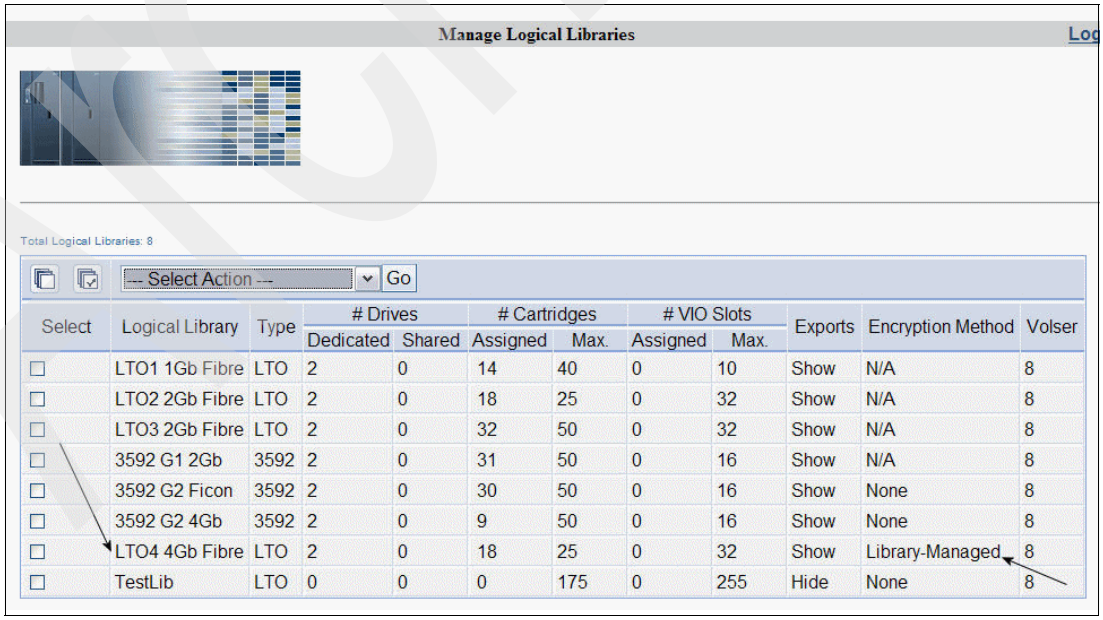


Figure 6-28 Our LME ready logical library

6.6.2 Specifying a Barcode Encryption Policy

Now that our logical library, *LTO4 4GB Fibre,* is set to encrypt data using a barcode policy, it is time to set up the BEPs that we use.

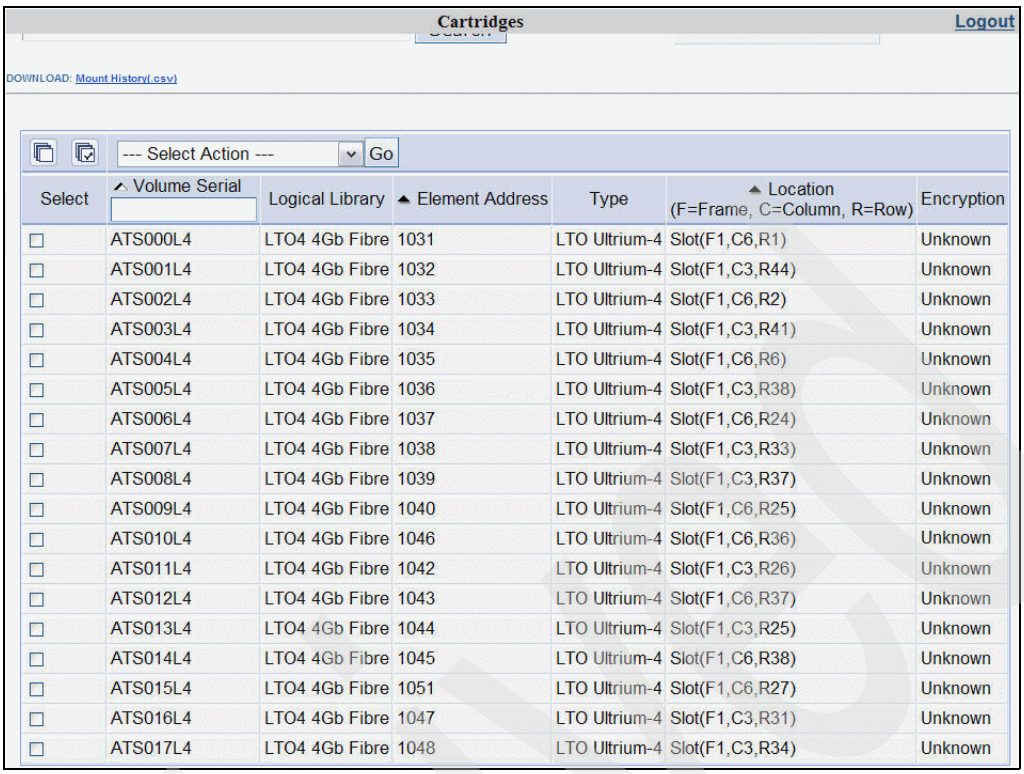
A *Barcode Encryption Policy* (BEP) is a range of cartridge volume serial numbers (VOLSERs), and it specifies which scratch cartridges to encrypt on the next attempt to write from the beginning of the tape. It does not indicate which cartridges are currently encrypted. When used with Library-Managed Encryption (LME), a Barcode Encryption Policy controls cartridge encryption by VOLSER ranges in *all* logical libraries that have LME with BEP selected. If you have defined multiple policies (overlap is not allowed), they all are effective simultaneously, and they affect which cartridges are to be encrypted in *every* defined library-managed logical library in a TS3500 that is enabled to encrypt using BEPs. All BEPs are shared.

When implementing a Library-Managed BEP with LTO4 drives, you specify which cartridges to encrypt and which key to use to encrypt the data.

Important: Determining by VOLSER range which cartridges to encrypt works the same for both LTO4 and 3592 cartridges. Note however, that the key labels within BEPS are used differently for LTO4 than for 3592. For 3592, the BEPs determine what key is used to encrypt the data key that was used to encrypt client data. For LTO4, the BEPs determine what data key is used to encrypt client data.

Both 3592 and LTO4 BEPs are managed similarly in that by using VOLSER ranges, you direct tape allocations to certain Barcode Encryption Policies. The BEPs that you select depend on the intended cartridge destination.

Let us review which cartridges are available to our library before we create a BEP. Select **Cartridges** → **Data Cartridges**, and then by using the pull-down menu, we select our logical library and click **Search** to produce the list of VOLSERs that you see in Figure 6-29 on page 222.



Select	Volume Serial	Logical Library	Element Address	Type	Location (F=Frame, C=Column, R=Row)	Encryption
☐	ATS000L4	LTO4 4Gb Fibre	1031	LTO Ultrium-4	Slot(F1,C6,R1)	Unknown
☐	ATS001L4	LTO4 4Gb Fibre	1032	LTO Ultrium-4	Slot(F1,C3,R44)	Unknown
☐	ATS002L4	LTO4 4Gb Fibre	1033	LTO Ultrium-4	Slot(F1,C6,R2)	Unknown
☐	ATS003L4	LTO4 4Gb Fibre	1034	LTO Ultrium-4	Slot(F1,C3,R41)	Unknown
☐	ATS004L4	LTO4 4Gb Fibre	1035	LTO Ultrium-4	Slot(F1,C6,R6)	Unknown
☐	ATS005L4	LTO4 4Gb Fibre	1036	LTO Ultrium-4	Slot(F1,C3,R38)	Unknown
☐	ATS006L4	LTO4 4Gb Fibre	1037	LTO Ultrium-4	Slot(F1,C6,R24)	Unknown
☐	ATS007L4	LTO4 4Gb Fibre	1038	LTO Ultrium-4	Slot(F1,C3,R33)	Unknown
☐	ATS008L4	LTO4 4Gb Fibre	1039	LTO Ultrium-4	Slot(F1,C3,R37)	Unknown
☐	ATS009L4	LTO4 4Gb Fibre	1040	LTO Ultrium-4	Slot(F1,C6,R25)	Unknown
☐	ATS010L4	LTO4 4Gb Fibre	1046	LTO Ultrium-4	Slot(F1,C6,R36)	Unknown
☐	ATS011L4	LTO4 4Gb Fibre	1042	LTO Ultrium-4	Slot(F1,C3,R26)	Unknown
☐	ATS012L4	LTO4 4Gb Fibre	1043	LTO Ultrium-4	Slot(F1,C6,R37)	Unknown
☐	ATS013L4	LTO4 4Gb Fibre	1044	LTO Ultrium-4	Slot(F1,C3,R25)	Unknown
☐	ATS014L4	LTO4 4Gb Fibre	1045	LTO Ultrium-4	Slot(F1,C6,R38)	Unknown
☐	ATS015L4	LTO4 4Gb Fibre	1051	LTO Ultrium-4	Slot(F1,C6,R27)	Unknown
☐	ATS016L4	LTO4 4Gb Fibre	1047	LTO Ultrium-4	Slot(F1,C3,R31)	Unknown
☐	ATS017L4	LTO4 4Gb Fibre	1048	LTO Ultrium-4	Slot(F1,C3,R34)	Unknown

Figure 6-29 The cartridges in our logical library

We have 18 cartridges in our logical library, which are VOLSERs `ATS000` through `ATS017`. Note that in the Encryption column, all of the cartridges reflect an *unknown* status. The encryption column contains one of three types of status: Encrypted, Not Encrypted, or Unknown. An *Unknown* cartridge has not been previously mounted in a 3592 tape drive or an Ultrium 4 tape drive.

Select **Cartridges** → **Barcode Encryption Policy**, which results in the display shown in Figure 6-30 on page 223. In this example, you see four BEPs that are already defined. The first BEP is for TS1120 and the last three BEPs, which refer to ATS cartridges, are for LTO4. If you did not know which VOLSERs were TS1120 compared to LTO4, how can you know which BEP refers to which drive type, TS1120 or LTO4?

Notice that the first BEP for VOLSERs `JAG000-JAG999` has two key-labels defined and that the ATS VOLSER BEPs only have one key label defined. Because 3592 wraps the data key and using two key labels, which point to public keys, uses the two keys to create two EEDKS on each cartridge; therefore, there are two key labels specified in this example.

Directing your attention to the ATS BEPs, you notice that the first BEP, which is for `ATS000-ATS003`, is defined with a mode of Direct-Default Set while the BEP for `ATS015-ATS017` has a mode of Direct-Specific. The Direct-Default Set setting calls for EKM to select a key from a pregenerated key alias range, which we defined as `LT00000-LT0001f` in 6.5.3, “Create a JCEKS keystore” on page 209 and was specifically shown in Example 6-51 on page 211. The ATS015 BEP actually refers to the key label for EKM to use to obtain the necessary DK in which to encrypt client data.

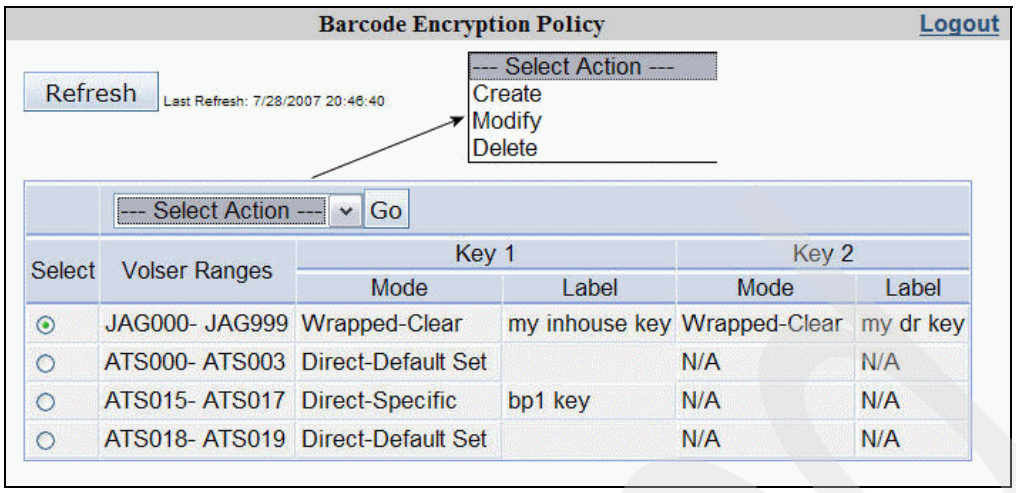


Figure 6-30 Existing Barcode Encryption Policies

Let us take a look at how to define a new BEP, which is shown in Figure 6-31.

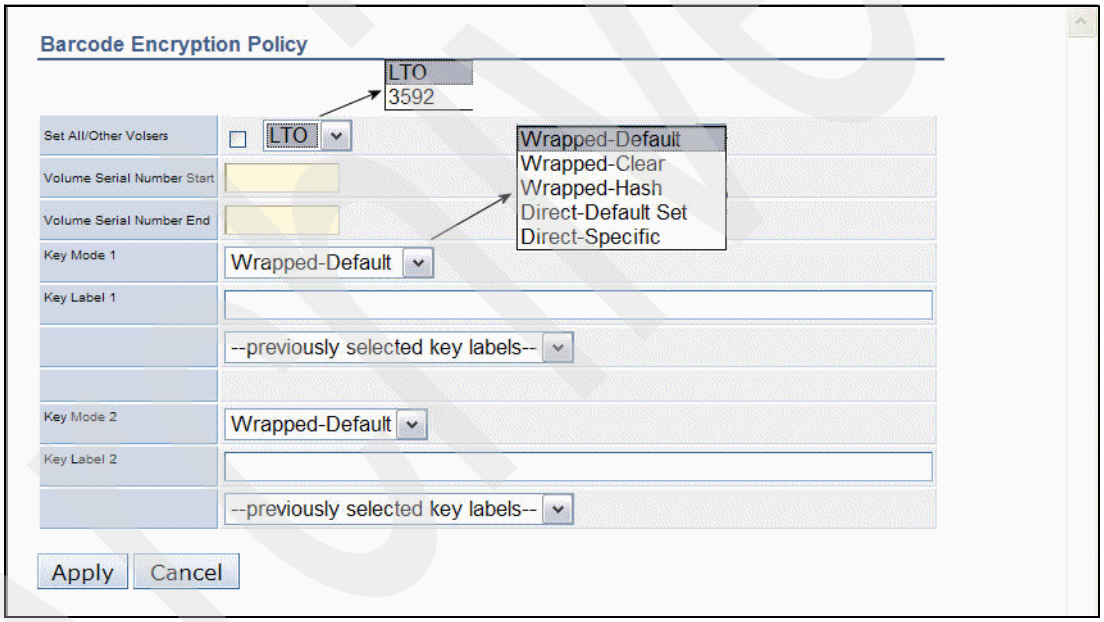


Figure 6-31 How to define a BEP

In Figure 6-31, select either 3592 or LTO from the All/Other Volsers drop-down list box. In our example, we select **LTO**. When LTO is selected, you do not select a Key Label 2, because that option is grayed out.

Enter the volume serial numbers of the scratch cartridges that begin and end the range to which you want to assign the barcode encryption policy. Because we cannot have BEP overlap, we select `ATS020-ATS029`.

Select the key mode from the drop-down list box. The choices are:

Wrapped-Default Default key label from either the Device drive table or the EKM configuration file. This is for 3592 cartridges only.

Wrapped-Clear The externally encoded data key (EEDK) is referenced by the specified key label. This is for 3592 cartridges only.

Wrapped-Hash The EEDK is referenced by a computer value, which corresponds to the public key that is referenced by the specified key label. This is also for 3592 cartridges only. For a good explanation of clear label compared to hash label, refer to “Creating a BEP using Hash values” on page 472.

Direct-Default Set Select a key in a round-robin manner from the alias range defined to the keystore. This is for LTO cartridges only and, in our example, results in a key referenced through a label from `LT00000-LT0001f`. We select this key mode for our example.

Direct-Specific The specified key label references a predefined symmetric data key. This is also LTO only and used to select a specific key instead of using a round-robin selection from a range of keys.

If this scenario were a TS1120 BEP instead, you would select two key modes for the policy. The key mode is the method by which public-private key pairs encrypt a data key to form an externally encoded data key.

Our selections are shown in Figure 6-32. After selecting **APPLY,** the new BEP is created.

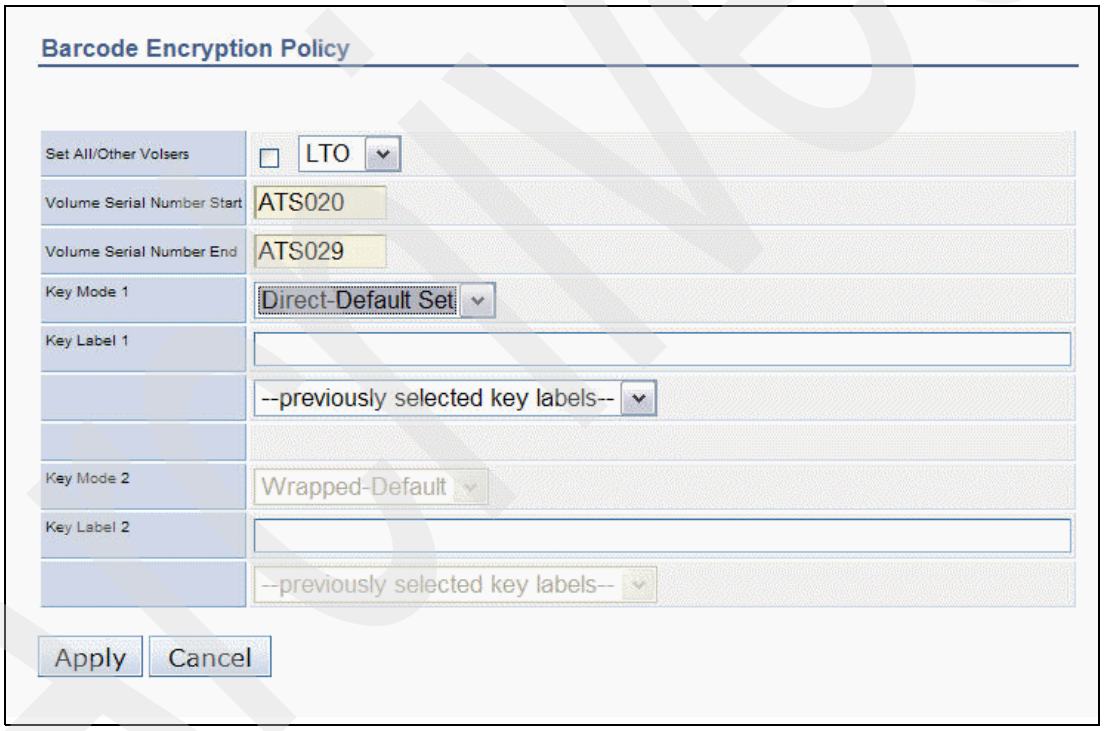


Figure 6-32 Creating a BEP

The new BEP we created is shown in Figure 6-33 on page 225.

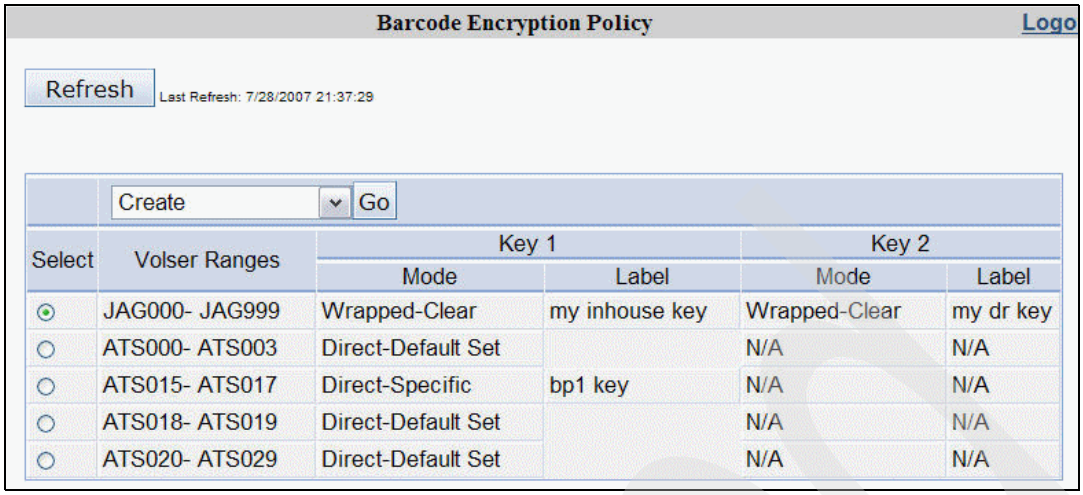


Figure 6-33 New BEP created

6.6.3 TS3500 Library-Managed Encryption differences from TS3310, TS3200, TS3100, and TS2900

The TS3500 is capable of managing significantly more cartridges than the other IBM LTO tape libraries because of this it offers more fine grained control over what cartridges to encrypt. When using library managed encryption with the TS3310, TS3200, TS3100 or TS2900 encryption is either on or off by Library. However, because of the partitioned nature of the TS3310, TS3200, and TS3100, you can still maintain a pool of unencrypted cartridges and a pool of encrypted cartridges. This may be accomplished by configuring the tape library into multiple library partition's.

- TS3310, TS3200, TS3100, and TS2900 do not support a library managed barcode encryption policy library managed encryption done on all cartridges in the logical library or as in the case of the TS2900 the whole library.
- For the TS3310, TS3200, TS3100 and TS2900 Library Managed Encryption is a licensed feature.
- The TS3310, TS3200, TS3100 and TS2900 support an SSL connection to the key manager.
- When ALMS is enabled, the TS3500 can be configured to use different key managers for each logical library.

6.7 LTO4 System-Managed Encryption implementation

In this section, we describe the implementation for System-Managed Encryption (SME) on the TS3500 library with LTO4 drives on the Windows platform. At this time, on Open Systems, SME is available on Windows, Linux, AIX, and Solaris platforms.

Note: At the time of this writing Library-Managed Encryption is the current best practice for Open Systems hosts

Encryption policies specifying when to use encryption are set up in the IBM tape device driver. System-Managed Encryption (SME) and Library-Managed Encryption (LME) interoperate with each other. A tape encrypted using SME can be decrypted using LME, and

the reverse is also true provided that they both have access to the same keys and certificates. Otherwise, this might not be feasible. For details about setting up SME, refer to *IBM Tape Device Drivers Installation and User's Guide,* GC27-2130, and the *Planning and Operator Guide* for your tape library.

6.7.1 LTO4 SME implementation checklist for Windows

Note: Best practice is to use Library-Managed Encryption on Windows unless using a drive not mounted in a tape library or autoloader.

To implement LTO4 SME for Windows on each host that will be accessing the drive:

1. Install the version of the device driver that supports encryption on the Windows operating system.
2. Configure the device driver encryption configuration file.
3. Edit the Windows registry for drives to be System-Managed.
4. At the TS3500, modify the encryption method for the logical library.

7

Planning and managing your keys

In this chapter, we outline the keystores than you can use in conjunction with the Encryption Key Manager (EKM) to complete a tape data encryption solution.

We intend that the discussions of keystores in this chapter be used as supplements to the implementation chapters. Here, we outline the steps and commands for configuring keystores. In Part 4, “Implementing tape data encryption” on page 379, we show you how to use a specific keystore. If the keystore used in the implementation chapters does not fit with your particular environmental requirements, the information that this chapter provides might be sufficient to help you create a keystore that suits your requirements.

7.1 Keystore and SAF Digital Certificates (keyrings)

A *keystore* is a database that stores a collection of symmetric and asymmetric keys protected by triple Data Encryption Standard (TDES or triple DES). Key entries hold sensitive cryptographic key information. Typically, key entries are secret keys or a private key and the certificate chain for the corresponding public key. Private keys and certificate chains are used by a given entity for self-authentication. A *Trusted Certificate Entry* contains a single public key certificate belonging to another party. A trusted certificate means that the keystore owner trusts that the public key in the certificate belongs to the identity specified in the subject of the certificate. This entry can be used to authenticate other parties.

System Authorization Facility (SAF) Digital Certificates are created and stored in the System Authorization Facility. It is common to refer to a SAF Digital Certificate as a *keystore* or *keyring*.

Storing certificate and keymaterial in a SAF provider adds another layer of security with which to protect keys in addition to implementing SAF interfaces for certificate and key management.

Certificates stored in a RACF keyring are always accompanied by a public key and optionally accompanied by a private key to create an asymmetric key pair. Symmetric key is not supported in SAF; Digital Signature Algorithm (DSA) keymaterial is not supported.

SAF Digital Certificates can also take advantage of IBM Integrated Cryptographic Service Facility (ICSF) and hardware cryptographic functions.

Refer to “MVS Open Edition tips” on page 570 and “Advanced security hwkeytool and keytool scripts” on page 573 for UNIX and UNIX System Services tips and advanced shell scripts for creating keystores with an added level of security.

7.2 JCEKS

Java Cryptographic Extension Keystore (JCEKS) is a UNIX System Services Java file-based keystore that is supported on all platforms where the EKM runs. This keystore is password-protected and TDES-encrypted. Several ways exist to transport and share a JCEKS, including but not limited to FTP and e-mail. Public and private keys can be exported from a JCEKS keystore and imported into a JCEKS or RACF keyring, JCE4758KS, or JCECCAKS. JCEKS also supports symmetric keys for encryption on LTO4 tape drives.

To manipulate a JCEKS keystore, use:

- Java keytool utility: The k*eytool* utility (key and certificate management tool) has been enhanced to generate aliases and symmetric keys for encryption on LTO4 tape drives using LTO4 media. It also provides for the import and export of symmetric keys to and from a JCEKS keystore. For more information about keytool, go to:

  `ftp://ftp.software.ibm.com/s390/java/jce/keytool.html`

- iKeyman utility: Currently, iKeyman cannot generate, import, or export symmetric keys. You can use iKeyman to list and delete symmetric keys, though.

We discuss the management of symmetric keys in 7.2.2, “Managing symmetric keys in a JCEKS keystore” on page 232.

The next section provides examples for generating key-pairs, creating a certificate, using a CA, and importing the certificate back into JCEKS keystore.

7.2.1 Examples of managing public-private key pairs

In this section, we generate a public-private key, create a self-signed certificate, and export a self-signed certificate. The exported self-signed certificate can be submitted to a third-party certificate authority. When the certificate authority sends back a certificate response, it can then be imported back into the JCEKS keystore.

You may cut and paste the following examples into script files and execute them. If you plan to run them from the command line, make sure to strip out the backslash characters (\). If you plan to enter these commands in the OMVS command line, you might run out of space. If that happens, either run the commands from a script or create environment variables to shorten the commands. The UNIX System Services environment, when entered using a Telnet daemon, rlogin daemon, or SSH daemon, does not present these problems. Refer to “MVS Open Edition tips” on page 570 for more information about working in the UNIX System Services environment.

Example of creating a public-private key pair

In Example 7-1, we generate an Rivest-Shamir-Adleman (RSA) algorithm public-private key pair. The keypass flag is used to protect the private key with a password and the storepass value flag is used to set the password of the keystore itself. In Example 7-1, the key alias is rsa2048Cert, the distinguished name of the subject is IBM, and the keystore file name is testkeys.jcks. The password used to protect this keystore is “passphrase”, and the expiration of the certificate is 999 days. The key size generated is 2048 bits in length.

Example 7-1 Keytool generating public-private key pair

```
keytool  -genkey  \
-alias  rsa2048Cert  \
-dname  "CN=IBM"
-keystore  testkeys.jcks  \
-provider  IBMJCE  \
-keyalg  RSA  \
-keysize  2048
-keypass  "passphrase"  \
-storepass  "passphrase" \
-storetype  JCEKS \
-validity  999
```

Tip: Keytool commands must be typed on a single line. If you choose to copy these examples into a script file, the backslash character (\) can be used to span lines and improve readability. In OMVS when you use OEDIT, watch out for hidden characters that might break the script. You can use the **`hex on`** command to view all hidden characters.

The caveat, however is that **`hex on`** does not show whether spaces exist at the end of the line. If you use Telnet, rlogin, or ssh to connect to OMVS vi or emacs, make sure that no characters, including spaces, exist after the backslash character.

Example of exporting a certificate

In Example 7-2 on page 230, the certificate with an alias of rsa2048Cert1 is exported to the file rsa2048Cert1.cer. This is a binary certificate format and does not include private key information. If the `-rfc` flag is used, the certificate is exported in a printable encoding format, as defined by the Internet RFC 1421 standard. If the `-pkcs12` flag is used, the certificate that is created conforms to the pkcs12 file format, and both public and private keymaterial are exported and symmetrically encrypted.

Example 7-2 Export public key into a file

```
keytool  -export \
-file  rsa2048Cert1.cer \
-keystore  testkeys.jcks \
-alias  rsa2048Cert1 \
-storepass  passphrase \
-storetype  JCEKS  \
-provider  IBMJCE  \
-keypass  passphrase
```

Example of importing a certificate

In Example 7-3, we import our certificate back into the keystore as a trusted certificate entry. If we had sent the exported certificate to a third-party certificate authority (CA), we import the fulfilled certificate that was returned to us here. This certificate file can be imported into other keystores.

Example 7-3 Import a trusted certificate

```
keytool  -import  \
-file  rsa2048Cert1.cer \
-keystore  testkeys.jcks \
-alias  CArsa2048 \
-storepass  passphrase \
-storetype  JCEKS \
-provider  IBMJCE \
-keypass  passphrase
```

Example of the keytool -list command

Example 7-4 shows the command to list a JCEKS keystore. This command does not list private key information. Use the program in 7.8, “ShowPrivateTool” on page 267 to list private key information.

Example 7-4 List a keystore

```
keytool  -list \
-keystore  testkeys.jcks \
-storetype jceks \
-storepass passphrase
```

Output from the list command looks similar to Example 7-5.

Example 7-5 Output from keytool list

```
Keystore type: jceks
Keystore provider: IBMJCE

Your keystore contains 2 entries

rsa2048Cert1, Sep 18, 2006, keyEntry,
Certificate fingerprint (MD5): 93:CB:BB:04:B4:A5:9B:42:D6:C9:3C:05:FF:8B:E8:15
CArsa2048, Sep 18, 2006, trustedCertEntry,
Certificate fingerprint (MD5): E0:BD:75:2B:50:06:EA:C9:F7:BE:5E:8D:EC:4B:11:A3
```

Example of the keytool using pkcs12 format

PKCS12 is a standard format defined by RSA that describes a file for moving public and private keymaterial. A PKCS12 file is used to store an X.509 certificate, the public and private keys associated with that certificate, the chain of Certificate Authorities that signs that certificate. The file is encrypted with a symmetric key based on a passphrase. You use PKCS12 files to move your keys from site to site, or to backup keys in situations where moving a complete JCEKS does not satisfy the key management procedures.

In Example 7-6, we show the keytool commands to export out a certificate with an alias of `TestCert` from our keystore `SourceKeys.jck` into a PKCS12 file. We are using a password of `CLEARTEXTPASS`.

Example 7-6 export pkcs12 file

```
keytool -export -alias "TestCert" \
-file export.p12 \
-keypass "CLEARTEXTPASS" \
-storepass "CLEARTEXTPASS" \
-pkcs12 \
-keystore SourceKeys.jck \
-storetype JCEKS \
-provider IBMJCE
```

In Example 7-7, we take the PkCS12 file we created in Example 7-6 and store it with an alias of `TestCert` into our `destinationKeys.jck` keystore. If our certificate had been signed by a certificate authority (CA) instead of being a self-signed certificate, this import would have also imported the CA certificates into the keystore. When moving certificates into a JCEKS keystore, if you do not have the signing certificates already in the keystore, you will not be able to import a non-self-signed certificate into the keystore because the trust status of the certificate would not be known.

Example 7-7 Import PKCS12 file into JCEKS keystore

```
keytool -import -noprompt -trustcacerts \
-alias "TestCert" \
-file export.p12 \
-keypass "CLEARTEXTPASS" \
-storepass "CLEARTEXTPASS" \
-pkcs12 \
-keystore destinationKeys.jck \
-storetype JCEKS \
-provider IBMJCE
```

Example 7-8 on page 232 shows the commands we used to list both the source keystore and the destination keystore.

Example 7-8 Listing the keystore

```
keytool -list \
-storepass "CLEARTEXTPASS" \
-keystore SourceKeys.jck \
-storetype JCEKS \
-provider IBMJCE



keytool -list \
-storepass "CLEARTEXTPASS" \
-keystore destinationKeys.jck \
-storetype JCEKS \
-provider IBMJCE
```

Finally, in Example 7-9, we have a listing of our source keystore, and in Example 7-10 we have the destination keystore that now has two certificates and keypairs in it. If we were to run the ShowPrivate utility from 7.8, “ShowPrivateTool” on page 267, it would show that both entries in the destination keystore have private keys.

Example 7-9 Source Keystore listing

```
Keystore provider: IBMJCE

Your keystore contains 1 entry

testcert, Nov 19, 2008, keyEntry,
Certificate fingerprint (MD5): 00:51:A8:93:CD:49:EA:4C:2D:E3:55:2B:55:68:00:5D

Keystore type: JCEKS
Keystore provider: IBMJCE
```

Example 7-10 Destination keystore listing

```
Your keystore contains 2 entries

gumbycert, Nov 19, 2008, keyEntry,
Certificate fingerprint (MD5): 80:AD:0A:AC:C0:76:AD:7E:B1:88:4E:38:26:C6:1C:E2
testcert, Nov 19, 2008, keyEntry,
Certificate fingerprint (MD5): 00:51:A8:93:CD:49:EA:4C:2D:E3:55:2B:55:68:00:5D
```

7.2.2 Managing symmetric keys in a JCEKS keystore

The keytool utility has been enhanced with three command parameters:

keytool -genseckey	Generate symmetric key.
keytool -importseckey	Import a symmetric key.
keytool -exportseckey	Export a symmetric key.

Each data key in the keystore is accessed through a unique alias. An *alias* is a string of characters, such as ibm123tape. In JCEKS as in most other keystores, aliases are not case-sensitive. IBM123Tape is equivalent to ibm123tape and allows access to the same entry in the keystore. 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 for use in writing and reading encrypted data on LTO4 tape.

In the following sections, we detail the commands to generate, import, and export keys. We discuss the command parameters and provide examples for their usage.

Generating aliases and data keys using keytool -genseckey

Use the **keytool -genseckey** command to generate one or more secret keys and store them in a specified keystore. The **keytool -genseckey** command takes the following parameters:

```
keytool -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>]
```

The following parameters are of particular importance when generating data keys for EKM to serve to the LTO4 drives for tape encryption:

-alias Specifies an alias value for a single data key with up to 12 printable characters (for example, `abcfrg` or `ibm123tape`).

-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 that are 21 characters in length. For example, specifying an aliasrange value of `ekm1-a` yields a series of aliases from `ekm0000000000000001` through `ekm000000000000000a`. Specifying an aliasrange value of `ekm01-ff` yields `ekm0000000000000001` through `ekm00000000000000ff`.

-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. The keypass must be at least six characters long.

-keyalg Specifies the algorithm to be used to generate the data key. On a distributed platform, the key algorithm must be specified as AES. If the encrypted tape will be shared with systems on the z/OS platform (the zOSCompatibility property is set to `true`), the key algorithm must be specified as DESede.

-keysize Specifies the size of the data key to be generated. For Open Systems platforms, key size must be specified as 256. Do not use this parameter when the zOSCompatibility property is set to true.

You do not have to specify the parameters `-providerClass` and `-providerArg` when generating symmetric keys for a JCEKS keystore. The IBMJCE provider is used by default. You have to set these parameters when you are using a hardware-based keystore rather than JCEKS.

Example of generating a range of symmetric keys

You may cut and paste Example 7-11 on page 234 into a script file and execute it. If you plan to run it from the command line, make sure to strip out the backslash characters (\).

In this example, we generate ten symmetric keys from `ekm0000000000000001` through `ekm000000000000000a`.

Example 7-11 Generating a range of symmetric keys

```
keytool -genseckey \
-aliasrange ekm1-a \
-keyalg AES \
-keysize 256 \
-keystore EKMKeys.jceks \
-storetype jceks
```

After generating keys and aliases, be sure to update the symmetricKeySet property in the KeyManagerConfig.properties file to specify the new alias or range of aliases. To specify the ten symmetric keys generated in the previous example as the active key set, you set the symmetricKeySet parameter in the configuration file:

```
symmetricKeySet = ekm01-a
```

You may add aliases of key ranges and specific keys to the `symmetricKeySet` parameter. Add the aliases at the end of the `symmetricKeySet` statement separating them by a comma.

If the `symmetricKeySet` statement is missing or the syntax is wrong, EKM prints the following message when starting:

```
No symmetric keys in symmetricKeySet, LTO drives cannot be supported.
```

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, EKM does not start, an audit record is created, and you get an error message:

```
Error: Unable to find Secretkey in the config keystore with alias: ekm1-a
Server failed to start
```

EKM uses the keys in the active key set in a round robin fashion for encrypting data. Though EKM only uses keys in the active key set for encrypting data, keys for decrypting data do not have to be in the active key set. EKM uses symmetric keys to decrypt data that was previously encrypted with these keys regardless of the active key set, as long as the keys are in the keystore.

Note: For EKM to use the new alias or range of aliases, you have to to update the symmetricKeySet property in the KeyManagerConfig.properties file.

Importing symmetric keys using keytool -importseckey

Use the **keytool -importseckey** command to import a symmetric key from an import file. The **keytool -importseckey** command takes the following parameters:

```
keytool -importseckey
   [-v] [-keyalias <keyalias>] [-keypass <keypass>]
   [-keystore <keystore>] [-storepass <storepass>]
   [-storetype <storetype>] [-providerName <name>]
   [-importfile <importfile>]
```

The following parameters are of particular importance when importing data keys for EKM to serve to LTO4 drives for tape encryption:

-keyalias Specifies the alias of a private key in the keystore to decrypt all the data keys in the import file.

-importfile Specifies the keystore file that contains the data keys to be imported.

Example of importing symmetric keys

You may cut and paste Example 7-12 into a script file and execute it. If you plan to run it from the command line, make sure to strip out the backslash characters (\).

In this example, we import symmetric keys from the file `import.key` into the keystore `EKMKeys.jceks` using the private key with the alias `ekmcert` to decrypt the symmetric keys.

Example 7-12 Importing symmetric keys

```
keytool -importseckey \
-keyalias ekmcert \
-keystore EKMKeys.jceks \
-storetype jceks \
-importfile import.key
```

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. The **`keytool -exportseckey`** command takes the following parameters:

```
keytool -exportseckey
   [-v] [-alias <alias> | aliasrange <aliasRange>] [-keyalias <keyalias>]
   [-keystore <keystore>] [-storepass <storepass>]
   [-storetype <storetype>] [-providerName <name>]
   [-exportfile <exportfile>]
```

The following parameters are of particular importance when exporting data keys for EKM to serve to LTO4 drives for tape encryption:

-alias Specifies an alias value for a single data key with up to 12 printable characters (for example, `abcfrg` or `ibm123tape`).

-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 that are 21 characters in length. For example, specifying an aliasrange value of `ekm1-a` yields a series of aliases from `ekm000000000000000001` through `ekm0000000000000000a`. Specifying an aliasrange value of `xyz01-ff` yields `xyz000000000000000001` through `xyz0000000000000000ff`.

-keyalias Specifies the alias of a public key in a keystore to encrypt all data keys.

-exportfile Specifies the keystore file to contain the data keys to be exported.

Example of exporting a range of symmetric keys

You may cut and paste Example 7-13 on page 236 into a script file and execute it. If you plan to run it from the command line, make sure to strip out the backslash characters (\).

In this example, we export ten symmetric keys from `ekm000000000000000001` through `ekm00000000000000000a` from the keystore `EKMKeys.jceks` into a file export.key using the public key with the alias ekmcert to encrypt the symmetric keys.

Example 7-13 Exporting a range of symmetric keys

```
keytool -exportseckey \
-aliasrange ekm1-a \
-keystore EKMKeys.jceks \
-storetype jceks \
-exportfile export.key \
-keyalias ekmcert
```

7.2.3 Example using IKEYMAN

IKEYMAN (IBM Key Management) is a graphical user interface that is included in Java Software Developer Kits (SDKs). You use it for working with keystores. IKEYMAN program code lives in `$JAVA_HOME/jre/bin`:

- On Windows systems, the executable is IKEYMAN.EXE.
- On UNIX systems, the binary is **`ikeyman`**.

In this section, we present a similar set of examples using IKEYMAN.

Note: IKEYMAN is useful for working with X.509 certificates, however to perform symmetric key operations on a JCEKS you must use keytool CLI commands.

To use IKEYMAN:

1. Issue the **`ikeyman`** command:

   ```
   java  com.ibm.gsk.ikeyman.Ikeyman
   ```

2. The IBM Key Management (IKEMAN) window opens, which is shown in Figure 7-1 on page 237. Click the new file (circled) icon beneath the menu bar, or select **Key Database** → **File** → **New**.

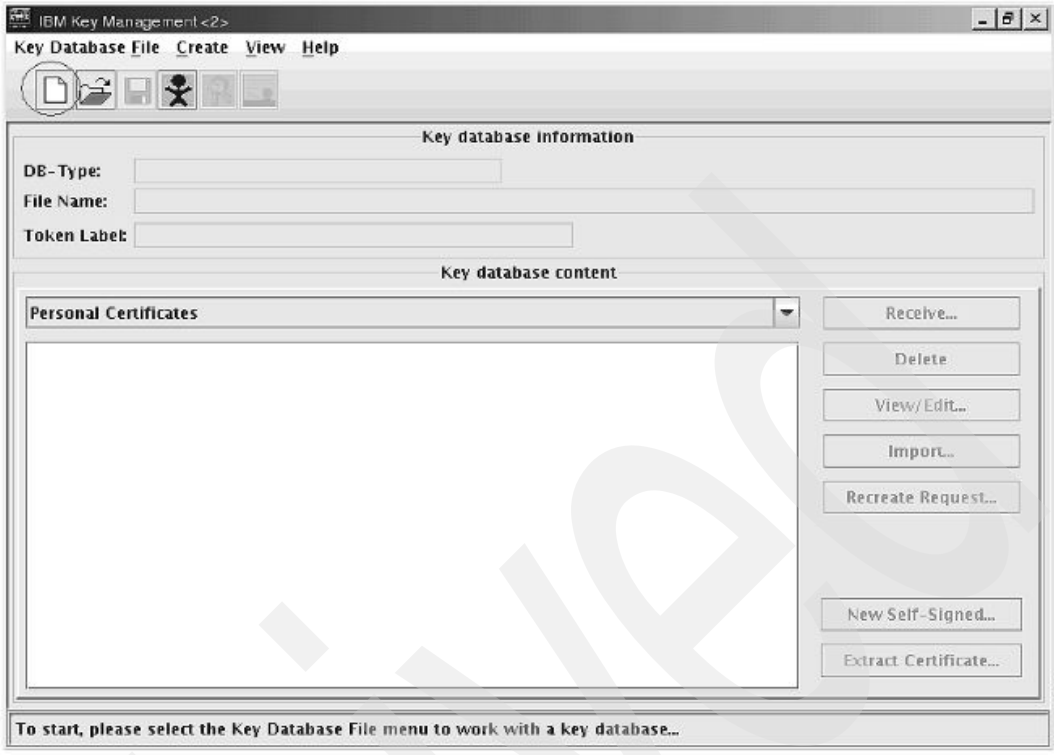


Figure 7-1 IKEYMAN main window

The New key database window opens as shown in Figure 7-2.

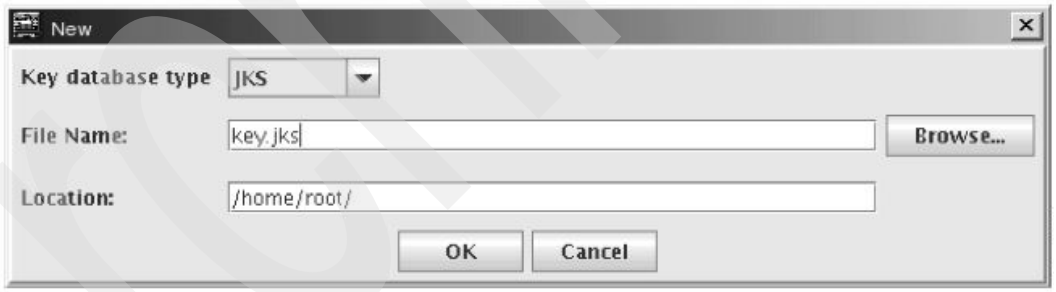


Figure 7-2 New key database window

3. Use the Key database type list box to click **JCEKS** and enter the keystore file name and location (Figure 7-3). You also are required to specify this file name and location in the EKM configuration file. Click **OK**.

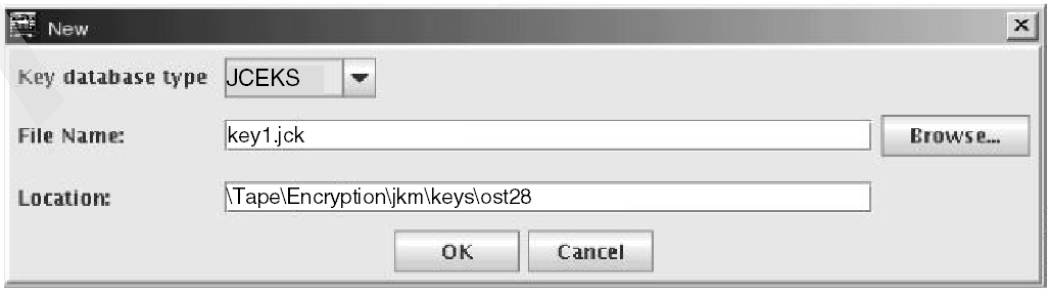


Figure 7-3 New Key database JCEKS

4. The Password Prompt window opens that is shown in Figure 7-4. Enter a password. You specify this same password in the EKM configuration file. Click **OK**.

Figure 7-4 Password Prompt dialog

5. In the IBM Key Management window in Figure 7-5, click the certificate (circled) icon, or select **Create** → **Create New Self-Signed Certificate**.

Figure 7-5 Signing certificate dialog

6. Specify the following values in the Create New Self-Signed Certificate window, shown in Figure 7-6:
   a. Key label can be any value but must not contain any blanks. The key label values correspond to the values that you specify for alias1 and alias2 when editing the EKM configuration file.
   b. Click **X509 V3** in the Version list box.
   c. Click **1024** in the Key Size list box.
   d. The Common Name field defaults to the name of the host that launched the iKeyman application. Any name can be specified.
   e. You must specify an organization name to create a certificate. The remaining fields are optional or default values.
   f. Click **OK**.

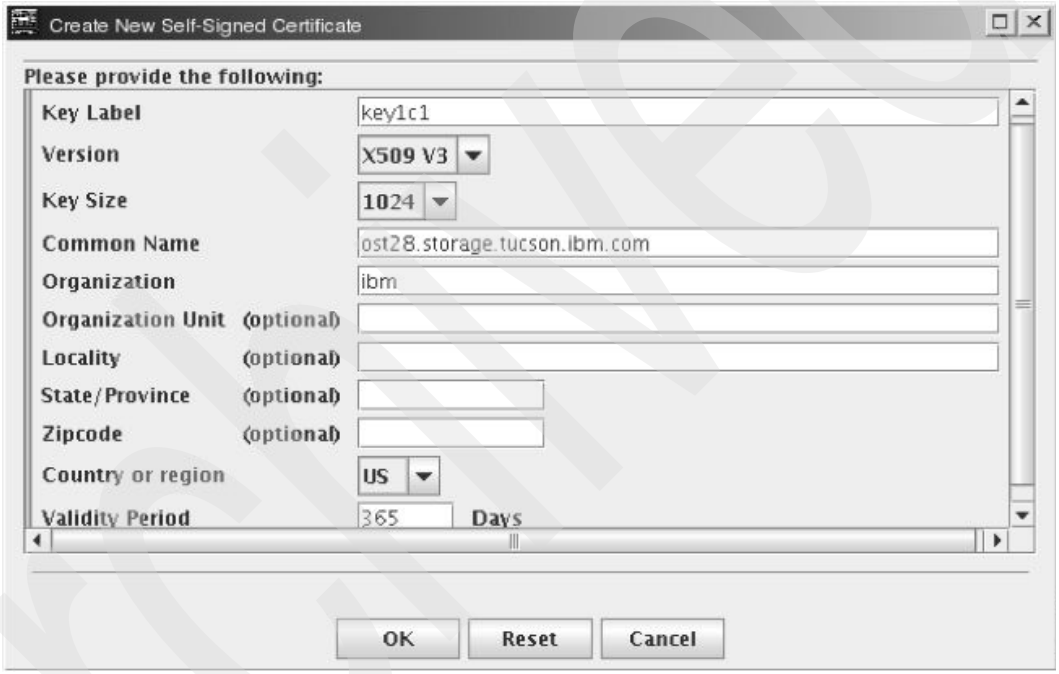


Figure 7-6 Create New Self-Signed Certificate

The IBM Key Management window in Figure 7-7 on page 240 opens, showing the new certificate.

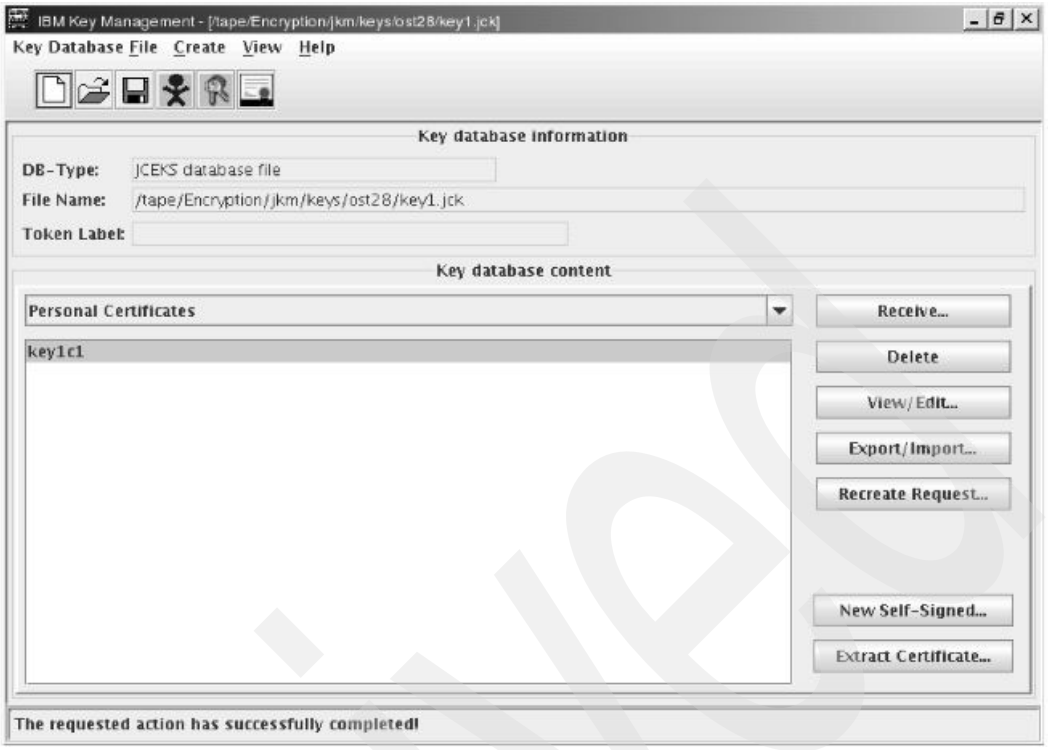


Figure 7-7 Certificate added

7. Create a second self-signed certificate using the dialogs from Figure 7-5 on page 238 and Figure 7-6 on page 239. We created two certificates, which are shown in Figure 7-8.

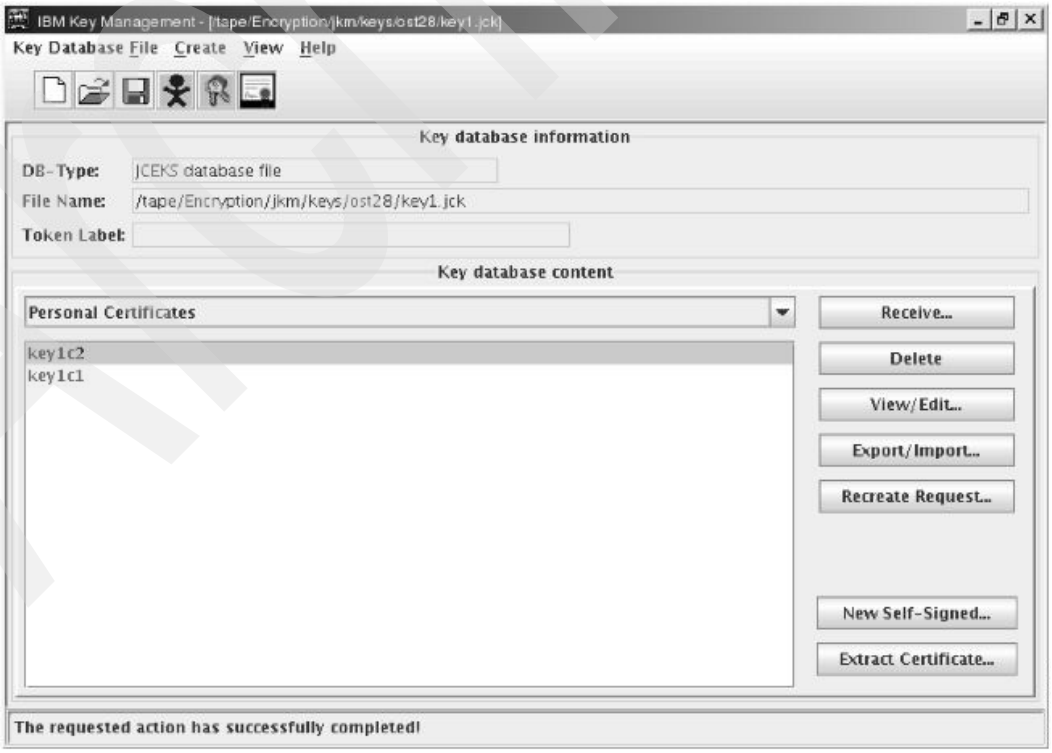


Figure 7-8 Certificates that we created

8. To import certificates that were created in another instance of iKeyman, click Export/Import on the right. In the Export/Import Key window in Figure 7-9:
   a. Select **Import Key**
   b. Select key file type **JCEKS**
   c. Specify the file name of the keystore from which to import keys (`keys5.jck`, in our example) and its location.

Figure 7-9 Export/Import Key dialog

9. The Password Prompt window shown in Figure 7-10 opens. Enter the required password and click **OK**.

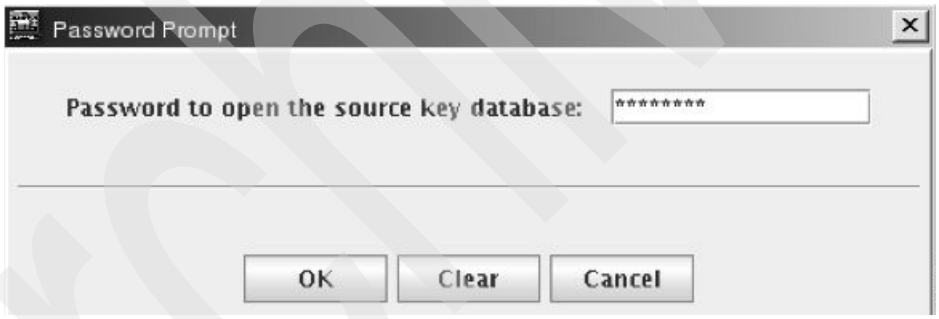


Figure 7-10 Password dialog for source key database

A list of key labels in the specified keystore displays in Figure 7-11.

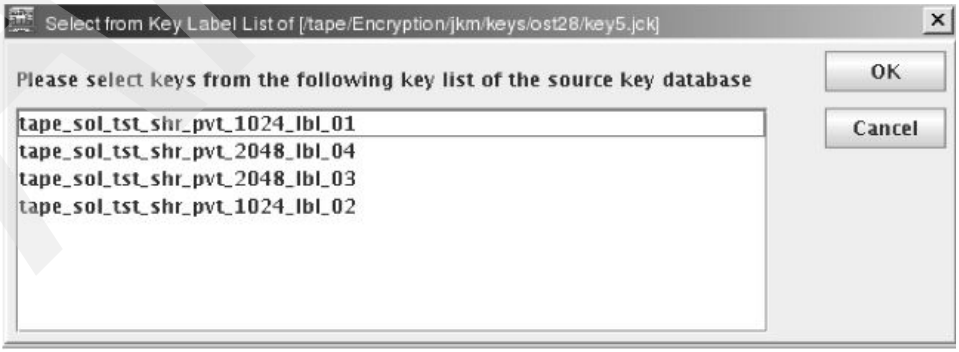


Figure 7-11 Keystore label list

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

11. The Change Labels window opens in Figure 7-12. If you wish to change the key label, select it, and then enter a new label name in the following field. Click **Apply**.

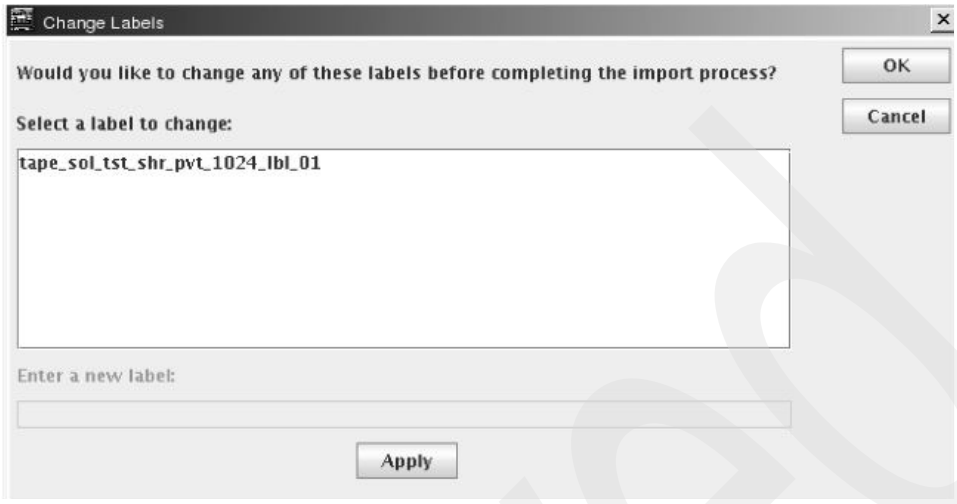


Figure 7-12 Change Labels dialog

12. Click **OK**. The IBM Key Management window appears in Figure 7-13 and displays the two certificates that we created and the one that we imported.

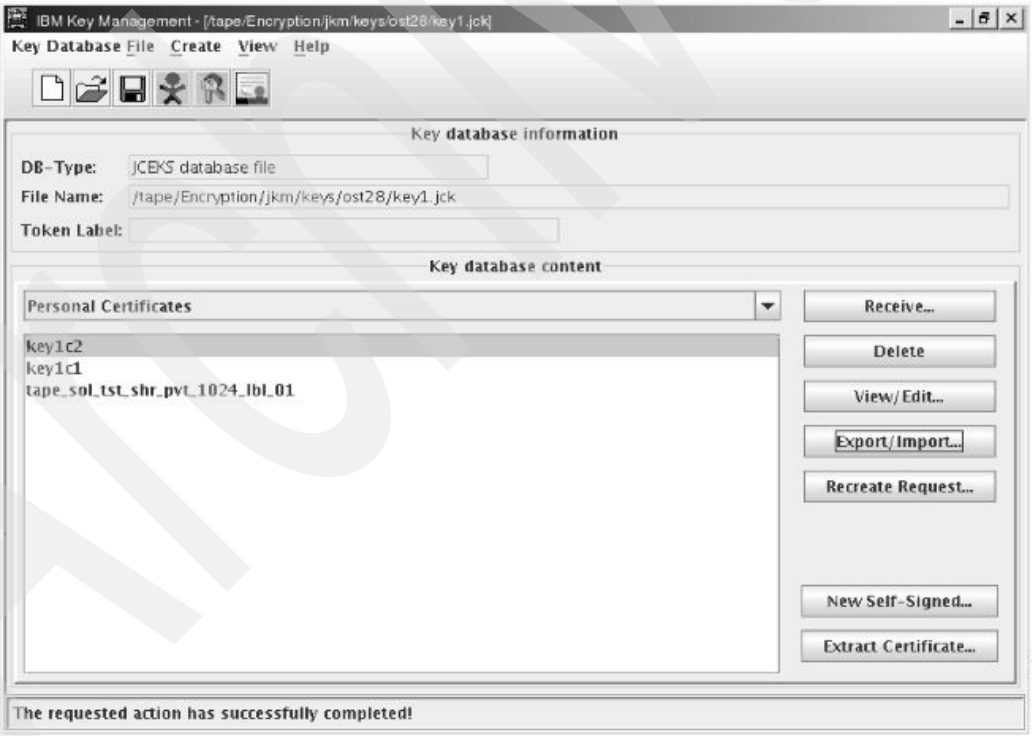


Figure 7-13 Keystore listing

On Open Systems, a JCEKS keystore can be created and manipulated with either IKEYMAN or command line-entered keytool commands. On z/OS systems, only keytool commands are available.

7.3 JCE4758KS and JCECCAKS

JCE4758KS is a Java file keystore type, which is supported by Java Developer Kit (JDK) 1.4 and 5.0, that takes advantage of ICSF. This keystore is password-protected and triple DES-encrypted, and the keymaterial is protected by hardware in one of three ways:

- CLEAR (or LABEL)
- PKDS
- RETAINED

We suggest that you do not use RETAINED with EKM. Refer to 7.10, “Hardware cryptography” on page 272 for more information. You can use Example 7-14 on page 244 also to create JCECCAKS keystores. To use this example, you have to replace the provider with the IBMJCECCA provider and replace the key type with JCECCAKS. You can change the hardware type from PKDS to CLEAR or RETAINED depending on what type and level of hardware security you want. For more information about the hwkeytool, go to:

`ftp://ftp.software.ibm.com/s390/java/jce4758/hwkeytool.html`

If a JCECCAKS is specified with the CLEAR option, a public-private key pair can be exported in a pkcs12 file using the IBMJCECCA provider. This is only supported in JDK 5.0 and higher.

7.3.1 Script notes

You may cut and paste the following examples into script files and execute them. If you plan to run them from the command line, make sure to strip out the backslash characters (/). If you plan to enter these commands on the OMVS command line, you might run out of space. If that happens, either run the commands from a script or create environment variables to shorten the commands. The UNIX System Services environment, when entered using a Telnet daemon, rlogin daemon, or SSH daemon, does not present these problems. Refer to “MVS Open Edition tips” on page 570 for more information about working in the UNIX System Services environment.

Note: To create and use keystores of type JCE4758KS, the `java.security` file located in `$JAVA_HOME/lib/security/` has to include the JCE4758 provider in the list. Similar to:

```
security.provider.1=com.ibm.jsse.IBMJSSEProvider
security.provider.2=com.ibm.crypto.hdwrCCA.provider.IBMJCE4758
security.provider.3=com.ibm.crypto.provider.IBMJCE
```

The script in Example 7-14 on page 244 demonstrates how to generate a public-private key pair using the hwkeytool. Here, we create an alias of rsa2048hdwrCert1 with a distinguished name of IBM and store the certificate in the keystore `testkeys.cca`. The IBMJCE4758 provider is used with the RSA algorithm and a key size of 2048 bits. The store type is JCE4758KS; the private keymaterial is stored in a PKDS. The keypass and keystore password are both passphrase.

Example 7-14 Generate a JCE4758KS public-private key pair

```
hwkeytool  -genkey \
-alias  rsa2048hdwrCert1 \
-dname  "CN=IBM" \
-keystore  testkeys.cca \
-provider  IBMJCE4758 \
-keyalg  RSA \
-keysize  2048 \
-storetype  JCE4758KS \
-keypass  "passphrase" \
-storepass  "passphrase" \
-hardwaretype  PKDS
```

In Example 7-15, the certificate with an alias of rsa2048hdwrCert1 is exported to the rsa2048hdwrCert1.cer file. This is a binary certificate format and does not include private key information. If the -rfc flag is used, the certificate is exported in a printable encoding format, as defined by the Internet RFC 1421 standard. If the -pkcs12 flag is used, the certificate created conforms to the pkcs12 file format, and both public and private keymaterial is exported, but only if the specified alias is of type CLEAR.

Example 7-15 Exporting a certificate from a JCE4758KS

```
hwkeytool  -genkey \
-alias  rsa2048hdwrCert1 \
-dname  "CN=IBM" \
-keystore  testkeys.cca \
-provider  IBMJCE4758 \
-keyalg  RSA \
-keysize  2048 \
-storetype  JCE4758KS \
-keypass  "passphrase" \
-storepass  "passphrase" \
-hardwaretype  PKDS
```

In Example 7-16, we import our certificate back into the keystore as a trusted certificate entry. If we had sent the exported certificate to a third-party CA, we import the fulfilled certificate that was returned to us here. This certificate file can be imported into other keystores. The noprompt flag tells the hwkeytool that we will import the trusted entry without prompting, even though a copy of the public key is already in the keystore.

Example 7-16 Importing a trusted certificate using hwkeytool

```
hwkeytool  -import  \
-alias  CArsa2048hdwr \
-noprompt \
-file  rsa2048hdwr.cer \
-storetype  jce4758ks \
-storepass  passphrase \
-keypass  passphrase \
-keystore  testkeys.cca \
-hardwaretype  PKDS \
-provider  IBMJCE4758  -v
```

Example 7-17 on page 245 shows the command for listing a JCE4758KS keystore. Use the -v flag to print extended information. This command does not list private key information. Use

the program in 7.8, “ShowPrivateTool” on page 267 to list private key information. If the private key information is stored in PKDS or RETAINED, the tool prints the pointer to the key entry.

Example 7-17 Listing a keystore with hwkeytool

```
hwkeytool  -list  \
-keystore  testkeys.cca \
-storetype  jce4758ks
```

Sample output from the hwkeytool is shown in Example 7-18.

Example 7-18 Output from hwkeytool

```
Keystore type: JCE4758KS
Keystore provider: IBMJCE4758

Your keystore contains 2 entries

Alias name: rsa2048ccacert1
Creation date: Sep 18, 2006
Entry type: keyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=rsa4758Cert1
Issuer: CN=rsa4758Cert1
Serial number: 450f2c79
Valid from: Mon Sep 18 16:32:09 MST 2006 until: Sun Dec 17 16:32:09 MST 2006
Certificate fingerprints:
    MD5:  07:0A:29:00:F7:3D:AB:02:1E:2A:A5:A8:ED:F9:E3:C4
    SHA1: E3:80:25:32:8E:A3:C4:05:5B:EF:3F:5E:F8:40:ED:C1:6C:5A:47:5E

*****************************************
*****************************************


Alias name: carsa2048
Creation date: Sep 18, 2006
Entry type: trustedCertEntry

Owner: CN=rsa4758Cert1
Issuer: CN=rsa4758Cert1
Serial number: 450f2c79
Valid from: Mon Sep 18 16:32:09 MST 2006 until: Sun Dec 17 16:32:09 MST 2006
Certificate fingerprints:
    MD5:  07:0A:29:00:F7:3D:AB:02:1E:2A:A5:A8:ED:F9:E3:C4
    SHA1: E3:80:25:32:8E:A3:C4:05:5B:EF:3F:5E:F8:40:ED:C1:6C:5A:47:5E
*****************************************
*****************************************
```

7.3.2 Symmetric keys in a JCECCAKS

The hwkeytool commands now support creating keys in a CKDS and mapping those keys by label to a JCECCAKS. A clever java programmer could have used CKDS keylabel mapping and a JCECCAKS to use SECURE symmetric keys for LTO4 drives. Now in Example 7-19 on

page 246 we have a hwkeytool command similar to the keytool command. Here we are creating a range of five symmetric keys in a JCECCAKS keystore where the key material is stored in a CKDS. The algorithm being used is TDES and the keysize is 192. The CEX2C does not support SECURE 256 bit AES keys. Using TDES with a keysize of 192 will work for tape encryption if the EKM has the `zoscompatability = true` setting.

Example 7-19 Creating a range of keys

```
hwkeytool -genseckey -v \
-aliasRange EKM2008101-2008105 \
-keypass "CLEARTEXTPASS" \
-keyalg TDES \
-keysize 192 \
-hardwaretype CKDS \
-keystore ekmkeys.cca \
-storepass "CLEARTEXTPASS" \
-storetype JCECCAKS \
-provider IBMJCECCA
```

7.4 JCERACFKS

The JCERACFKS uses SAF and RACF services to protect keymaterial and certificates. For SAF and RACF-stored keyrings, 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*, SA22-7687, which is available from:

http://publibz.boulder.ibm.com/epubs/pdf/ichza460.pdf

The following facility classes control access to RACDCERT command:

- IRR.DIGTCERT.ADD IRR.DIGTCERT.ALTER
- IRR.DIGTCERT.DELETE
- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.ADDRING
- IRR.DIGTCERT.DELRING
- IRR.DIGTCERT.LISTRING
- IRR.DIGTCERT.CONNECT
- IRR.DIGTCERT.REMOVE
- IRR.DIGTCERT.MAP
- IRR.DIGTCERT.LISTMAP
- IRR.DIGTCERT.ALTMAP
- IRR.DIGTCERT.DELMAP
- IRR.DIGTCERT.REKEY
- IRR.DIGTCERT.ROLLOVER
- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.LISTRING

The command in Example 7-20 generates a self-signed certificate authority certificate. You can use the MatchKeys tool in 7.9, “MatchKeys tool” on page 269 to check the public-private key pairs.

Example 7-20 Generate a CA

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('ITSO')O('IBM')C('US'))
WITHLABEL('LocalCertAuth') SIZE(1024)
```

Tip: In OMVS, a single quotation mark and a parenthesis must be prefixed with a backslash. The command listed in Example 7-20 looks like:

```
SYS1> tso "RACDCERT CERTAUTH GENCERT
SUBJECTSDN\(CN\(\'ITSO\'\)O\(\'IBM\'\)C\(\'US\'\)\)
WITHLABEL\(\'LocalCertAuth\'\) SIZE\(1024\)
```

The command in Example 7-21 generates an RSA key pair signed with the local certificate authority certificate that was created with the previous command.

Example 7-21 Generate a personal certificate

```
RACDCERT GENCERT SUBJECTSDN(CN('ITSO') O('IBM') C('US')) WITHLABEL('RSACert1')
SIZE(1024) SIGNWITH(CERTAUTH LABEL('LocalCertAuth'))
```

If you will be sending the certificate for third-party verification, it has to be exported as a certificate request. The command in Example 7-22 generates the request and stores it in the data set hlq.PUBKEY.REQUEST.ITSO.

Example 7-22 Generate a certificate request

```
RACDCERT GENREQ (LABEL('RSACert1')) DSN('hlq.PUBKEY.REQUEST.ITSO')
```

After this certificate is sent to a third-party and verified, a response is sent and we can receive it into `hlq.THIRD.PARTY.CERT`. You have to also add the CA that was used to sign this certificate to the ring. The command in Example 7-23 adds the certificate response into RACF.

Example 7-23 Import a certificate from a data set

```
RACDCERT ADD('hlq.THIRD.PARTY.CERT') TRUST WITHLABEL('RSACert1')
```

Next, we must create a keyring and connect our certificates to the keyring. The command in Example 7-24 creates a keyring named `ITSOring`.

Example 7-24 Create a new ring

```
RACDCERT ADDRING(ITSOring)
```

Now that a keyring exists, we can add our certificates to it using the two commands in Example 7-25.

Example 7-25 Connect certificates to a ring

```
RACDCERT CONNECT(CERTAUTH LABEL('LocalCertAuth') RING(ITSOring))
RACDCERT CONNECT(LABEL('RSACert1')RING(ITSOring) USAGE(PERSONAL) DEFAULT)
```

Note: When using a RACF keyring with Java applications, such as EKM, note that a keyring *is not* read successfully, unless there is a CA on the ring. It instead generates the following error:

```
CertPath not verified
```

This error also occurs if a SITE certificate that signed the personal certificate is on the ring. There must be a CA on the ring.

To share data with business partners, their public keys are imported into RACF and connected to the keyring that the EKM server is set up to use. To export a public key to send to a business partner, issue the command in Example 7-26.

Example 7-26 Export a certificate with a public key

```
RACDCERT EXPORT (LABEL('RSACert1')) DSN('hlq.PUBKEY.1024.ITSO') FORMAT(CERTDER)
```

This data set is then sent to a business partner, and the business partner then adds it to their EKM keystore and uses the label to encrypt tapes that they send to you. You have to validate with your business partners or remote sites that you trust a common CA, whether third-party or self-signed, depending on your business and security practices. This certificate can be imported into the keystore that is used by the EKM at the locations of your business partners. This lets everyone in the trust network verify that the keys that are used to encrypt data are in fact part of the trust network.

Note: The RACDCERT command can be used to list certificates and keyrings of other users when the appropriate permissions to the `irr.digtcert.*` classes are added. However, another user's private key information can never be read. When setting up EKM to run under a specific user account, remember that you cannot read another user's private key information.

Best practice

When connecting certificates to a keyring, the whole certificate chain must be connected to the ring. This is what allows the certificate path to be verified.

TIP: To verify that the EKM server has sufficient authority, you can issue the following **`hwkeytool`** command from OMVS:

```
hwkeytool -debug -J-Djava.protocol.handler.pkgs=com.ibm.crypto.provider -list
-keystore safkeyring://USERID/ITSOring -storetype JCERACFKS
```

7.5 JCE4758RACFKS and JCECCARACFKS

The JCE4758RACFKS and JCECCARACFKS SAF keyrings store certificate information in RACF, securing private keymaterial in a PKDS protected by ICSF. These keyrings are available on z/OS only. These keyrings are created and managed through the RACDCERT or equivalent SAF certificate management command.

JCE4758RACFKS and JCECCARACFKS are similar to keystores created with the PKDS option of the **`hwkeytool`** command. Instead of storing certificates in a file stored in UNIX System Services, the certificates are stored in a RACF keyring, protected by and managed with RACF or an equivalent SAF provider. The JCE4758RACKFKS keyring is supported on JDK 1.4.2. The JCECCARACFKS keyring is supported on JDK 5.0. And, JCE4758RACFS supports JVM 1.4.2.

The following facility classes control access to RACDCERT command:

- IRR.DIGTCERT.ADD IRR.DIGTCERT.ALTER
- IRR.DIGTCERT.DELETE
- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.ADDRING
- IRR.DIGTCERT.DELRING
- IRR.DIGTCERT.LISTRING

- IRR.DIGTCERT.CONNECT
- IRR.DIGTCERT.REMOVE
- IRR.DIGTCERT.MAP
- IRR.DIGTCERT.LISTMAP
- IRR.DIGTCERT.ALTMAP
- IRR.DIGTCERT.DELMAP
- IRR.DIGTCERT.REKEY
- IRR.DIGTCERT.ROLLOVER
- IRR.DIGTCERT.LIST
- IRR.DIGTCERT.LISTRING

7.5.1 RACDCERT keywords

The two keywords used in conjunction with the RACDCERT command to take advantage of cryptographic hardware are:

- ICSF
- PCICC

They specify how RACF generates the key pair and how the private key is stored for future use. If PCICC is specified, the key pair is generated using the PCI, PCI X, or Express2 cryptographic coprocessor. If ICSF is specified, the key pair is generated using software. In either case, the resulting private key is generated with the RSA algorithm and stored in the ICSF PKDS.

If either keyword, PCICC or ICSF, is specified and ICSF is not operational or is not configured for PKA operations, processing stops and an error message displays. If the PCICC keyword is specified, a PCI, PCI X, or Express2 cryptographic coprocessor must also be present and operational. Otherwise, processing stops and an error message displays.

If the key is stored as either an ICSF key or a PCICC key, any system using the key in the future is required to have ICSF operational and configured for PKA operations with this PKDS. For a PCICC key, any system using the key in the future will also need to have a PCI, PCI X, or Express2 cryptographic coprocessor present and operational with this PKDS.

The ICSF keyword adds private keymaterial to the PKDS with a generated label. The PCICC keyword lets you specify the PKDS label. The command in Example 7-27 generates a personal certificate that is signed with the LocalRACF CA and stores it in a PKDS with the PKDS label of `ITOPS.EKM.CERT` and with an alias of EKMServer. The MatchKeys tool (refer to 7.9, “MatchKeys tool” on page 269) can be used to check public-private key pairs.

Example 7-27 Generate a personal certificate with an alias of EKMServer

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

Note that the options of the GENCERT command are release-dependent. The options are:

For z/OS R1.8 `[ PCICC[(pkds-label | * )] | ICSF[(pkds-label | * )] | DSA ]`

For z/OS R1.7 `[ PCICC | ICSF | DSA]`

In the following examples, the key label is auto-generated.

The following command (Example 7-28 on page 250) generates a self-signed CA certificate.

Example 7-28 Generate a CA

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('ITSO')O('IBM')C('US')) PCICC
WITHLABEL('LocalCertAuth') SIZE(1024)
```

Tip: In OMVS, a single quotation mark and a parenthesis must be prefixed with a backslash. The command then looks as follows:

```
SYS1> tso "RACDCERT CERTAUTH GENCERT
SUBJECTSDN\(CN\(\'ITSO\'\)O\(\'IBM\'\)C\(\'US\'\)\) PCICC
WITHLABEL\(\'LocalCertAuth\'\) SIZE\(1024\)
```

The command in Example 7-29 generates an RSA key-pair signed with the local CA certificate created with the previous command.

Example 7-29 Generate a personal certificate

```
RACDCERT GENCERT SUBJECTSDN(CN('ITSO') O('IBM') C('US')) PCICC
WITHLABEL('RSACert1') SIZE(1024) SIGNWITH(CERTAUTH LABEL('LocalCertAuth'))
```

If you plan to send the certificate to third-party verification, it has to be exported as a certificate request. The command in Example 7-30 generates the request and stores it in the data set `hlq.PUBKEY.REQUEST.ITSO`.

Example 7-30 Generate a certificate request

```
RACDCERT GENREQ (LABEL('RSACert1')) DSN('hlq.PUBKEY.REQUEST.ITSO')
```

After this certificate is sent to a third-party and verified, a response is sent, and we can receive it into `hlq.THIRD.PARTY.CERT`. The command in Example 7-31 adds the certificate response into RACF. You must also add the CA that was used to sign this certificate to the ring.

Example 7-31 Import a certificate from a data set

```
RACDCERT ADD('hlq.THIRD.PARTY.CERT') TRUST ICSF WITHLABEL('RSACert1')
```

Next, we must create a keyring and connect our certificates to the keyring. The command in Example 7-32 creates a keyring named `ITSOring`.

Example 7-32 Create a new ring

```
RACDCERT ADDRING('ITSOring')
```

Now that there is a keyring, we can add our certificates to it using the two command, as shown in Example 7-33.

Example 7-33 Connect certificates to a ring

```
RACDCERT CONNECT(CERTAUTH LABEL('LocalCertAuth') RING('ITSOring'))
RACDCERT CONNECT(LABEL('RSACert1')RING(ITSOring) USAGE(PERSONAL) DEFAULT)
```

Note: When using a RACF keyring with Java applications, such as EKM, note that a keyring *is not* read successfully unless there is a CA on the ring. It instead generates the following error:

```
CertPath not verified
```

This error also occurs if a SITE certificate that signed the personal certificate is on the ring. There must be a CA on the ring.

To share data with business partners, their public keys are imported into RACF and connected to the keyring that the EKM server is set up to use. To export a public key to send to a business partner, issue the command in Example 7-34.

Example 7-34 Export a certificate with a public key

```
RACDCERT EXPORT (LABEL('RSACert1')) DSN('hlq.PUBKEY.1024.ITSO') FORMAT(CERTDER)
```

This data set is then sent to a business partner. The business partner then adds it to the business partner's EKM keystore and uses the label to encrypt tapes that the business partner sends to you.

Note: Use the RACDCERT command to list certificates and keyrings of other users when the appropriate permissions to the `irr.digtcert.*` classes are added. *However*, another user's private key information can never be read. You must consider this when you set up EKM to run under a specific user account.

You must validate a common CA, whether third-party or self-signed (depending on your business and security practices), with your business partners or remote sites that you trust. This CA certificate can be imported into the keystore that is used by the EKM at the locations of your business partners. This lets everyone in the trust network verify that the keys that are used to encrypt data are in fact part of the trust network.

7.5.2 Best practice

When connecting certificates to a keyring, the whole certificate chain must be connected to the ring, allowing the *certificate path* to be verified.

Tip: To verify that the EKM server has sufficient authority, the following **`hwkeytool`** command can be issued from OMVS:

```
hwkeytool -debug -J-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider
-list -keystore safkeyring://USERID/ITSOring -storetype JCERACFKS
```

The provider list in the `java.security` file (in `$JAVA_HOME/lib/security/java.security`) must contain the corresponding hardware provider:

```
com.ibm.crypto.hdwrCCA.provider.IBMJCECCA for a JCECCARACFKS
com.ibm.crypto.hdwrCCA.provider.IBMJCE4758 for JCE4758RACFKS
```

This is similar to the following list:

```
security.provider.1=com.ibm.jsse.IBMJSSEProvider
security.provider.2=com.ibm.crypto.hdwrCCA.provider.IBMJCECCA
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.security.cert.IBMCertPath
```

7.6 PKCS#11

The IBMPKCS11Impl provider uses the Java Cryptography Extension (JCE) and Java Cryptography Architecture (JCA) framework to seamlessly add hardware cryptography capabilities using the Public Key Cryptographic Standards # 11(PKCS#11) standard. This new provider takes advantage of hardware cryptography within the existing JCE architecture and gives Java 2 programmers the significant security and performance advantages of hardware cryptography with minimal changes to existing Java applications. Because the complexities of hardware cryptography are taken care of within the normal JCE, advanced security and performance using hardware cryptographic devices are made easily available.

PKCS#11support for symmetric keys depends on hardware vendors of cryptographic devices.

Supported platforms

The PKCS11Impl provider supports a subset of the platforms that the JVM supports at the 5.0 level (Refer to the IBM JVM for 5.0 specific documentation for the supported operating systems and any other JVM specific requirements). The supported platforms for Java 5.0 are:

- Win 32
- AIX 5.2/5.3 (32-bit and 64-bit)
- Linux (PPC 32-bit and 64-bit)
- Linux (Intel 32)
- Solaris (32-bit and 64-bit) SPARC only
- Linux on zSeries (32-bit and 64-bit)

7.7 IBMi5OSKeyStore

The contents of this keystore are based on an i5OS certificate store file and the password to access that file. This keystore class allows the modification of the certificate store. You can make changes without using the Digital Certificate Manager (DCM).

This keystore is new for Java SDK 5.0. It allows DCM certificate stores to be updated. Java-based tools can be used to create DCM certificate stores. Hardware key support is not available with this keystore, and there is no application ID support.

The IBMi5OSKeyStore implementation conforms to the Sun Microsystems, Inc., specification for the Java keystore application programming interface (API). You can find more information in the keystore javadoc information by Sun Microsystems, Inc., at:

`http://java.sun.com/j2se/1.5.0/docs/api/java/security/KeyStore.html`

Currently, the IBMi5OSKeyStore does not support symmetric keys. The types of encrypting tape drives that you support determines the type of keystore that you have to create:

- If you support only TS1120 tape drives, you can create either a JCEKS or a IBMi5OSKeystore keystore.
- If you support only LTO4 tape drives, or you have one tape library that supports both LTO4 and TS1120 tape drives, you must create a JCEKS keystore.
- If you have separate tape libraries for the LTO4 and TS1120 tape drives, you can set up two EKM servers for each of the tape libraries. One EKM server can run using an IBMi5OSKeystore for use with the TS1120 tape drive and one EKM server can run using a JCEKS for use with the LTO4 tape drive. The two EKM servers can run on the same system as long as they listen on different ports.

For details about managing a JCEKS keystore, refer to 7.2, “JCEKS” on page 228.

Note: IBMi5OSKeyStore does not support symmetric keys. You cannot use this keystore to encrypt data on LTO4 tape drives.

7.7.1 Digital Certificate Manager

A *digital certificate* is an electronic credential that you can use to establish proof of identity in an electronic transaction. An increasing number of uses are available for digital certificates to provide enhanced network security measures. For example, digital certificates are essential to configuring and using the Secure Sockets Layer (SSL). Using SSL allows you to create secure connections between users and server applications across an untrusted network, such as the Internet. SSL provides one of the best solutions for protecting the privacy of sensitive data, such as user names and passwords, over the Internet. Many System i services and applications, such as FTP, Telnet, and HTTP Server, provide SSL support to ensure data privacy.

System i provides extensive digital certificate support that allows you to use digital certificates as credentials in a number of security applications. In addition to using certificates to configure SSL, you can use them as credentials for client authentication in both SSL and virtual private network (VPN) transactions. Also, you can use digital certificates and their associated security keys to sign objects. *Signing objects* allows you to detect changes or possible tampering with object contents by verifying signatures on the objects to ensure their integrity.

Capitalizing on the System i support for certificates is easy when you use Digital Certificate Manager (DCM), a free feature, to centrally manage certificates for your applications. DCM allows you to manage certificates that you obtain from any CA. Also, you can use DCM to create and operate your own local CA to issue private certificates to applications and users in your organization.

You can obtain more information at:

`http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp?topic=/rzaha/rzahajssenative15.htm`

7.7.2 How to set up an IBMi5OSKeyStore

The following instructions show how to create a keystore with one self-signed certificate. For information about installing Digital Certificate Manager (DCM), visit the Information Center:

`http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp`

The following tasks assume that DCM is started and the home window opens.

To create a keyring/keystore instance:

1. Select **Create New Certificate Store** from the menu on the left. In the Create a New Certificate Store window, click **Other System Certificate Store** and click **Continue**. In the Certificate Store Name and Password window, shown in Figure 7-14 on page 254, specify a certificate store path and filename and click **Create**. This filename is also specified in your EKM configuration file.

Figure 7-14 DCM Certificate Store Name and Password dialog

2. The Certificate Store Created window opens. See Figure 7-15. Select a Certificate Store menu item to access the new keystore that you have just created.

Figure 7-15 Certificate Store Created

Generate public-private key pairs

Perform the following 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 organizational identity. These steps create a public-private key pair as well as a certificate request.

To generate public-private key pairs:

1. Click **Select a Certificate Store**. In the Select a Certificate Store window, as shown in Figure 7-16, click **Other System Certificate Store**. Click **Continue**.

Figure 7-16 Select Certificate Store window

2. The Certificate Store and Password window opens, as shown in Figure 7-17 on page 256. Specify the path and file name you entered in Create keyring/keystore instance in Figure 7-14 on page 254. Click **Continue**.

Figure 7-17 Certificate Store and Password

3. 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.
4. Select **Create Certificate**. Select VeriSign or another Internet certificate authority (CA) for the CA to sign the certificate and click **Continue**. Figure 7-18 shows this dialog.

Figure 7-18 Select a Certificate Authority (CA)

5. In the Create Certificate window, shown in Figure 7-19 on page 257, select a Key size of **1024** and enter a Certificate label value that corresponds to the alias1 key label value in your EKM configuration file. Complete the other fields as appropriate. Click **Continue**.

Figure 7-19 Create Certificate

6. The Certificate Request Created window opens (shown in Figure 7-20). Copy the contents of the certificate request and paste the contents into the certificate request form provided by the CA. When the signed certificate is received from the CA, copy it into a file on the System i server. Click **OK**.

Figure 7-20 Certificate Request

7. Select **Work with Server and Client Certificates**. The Work with Server and Client Certificates window opens, as shown in Figure 7-21 on page 258.

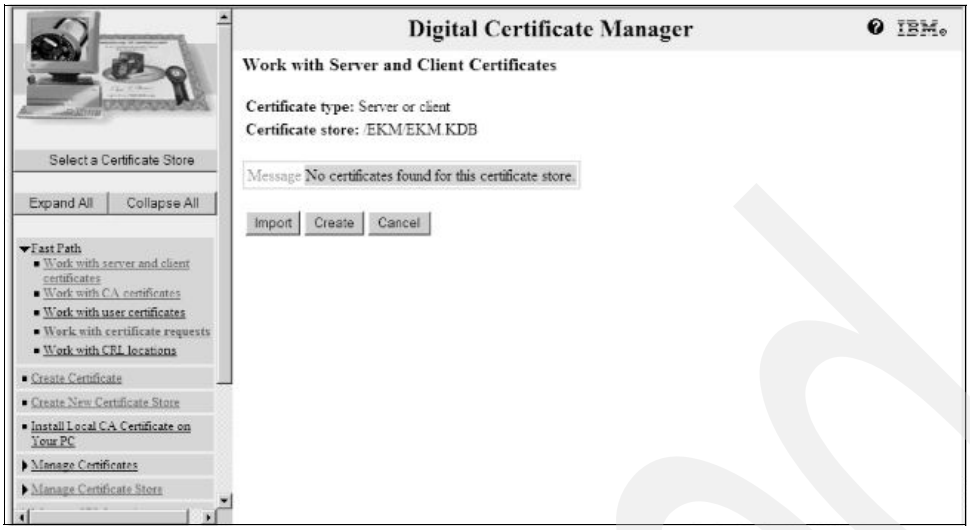


Figure 7-21 Work with Server and Client Certificates

8. Click **Import** to receive the signed certificate. The Import Server or Client Certificate window opens, as shown in Figure 7-22. Specify the name of the file into which you copied the signed certificate. Click **Continue**.

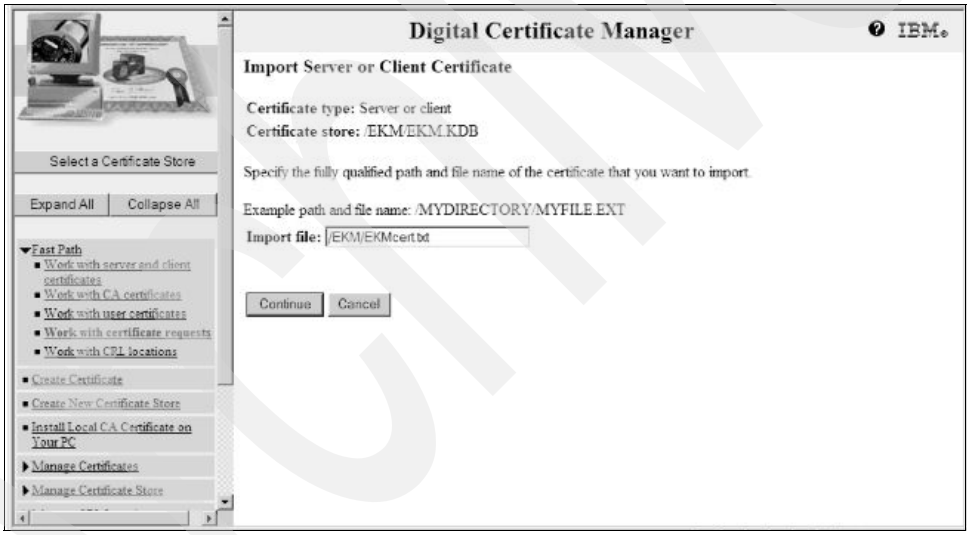


Figure 7-22 Import Server or Client Certificate

9. The Work with Server and Client Certificates window opens, as shown in Figure 7-23 on page 259.

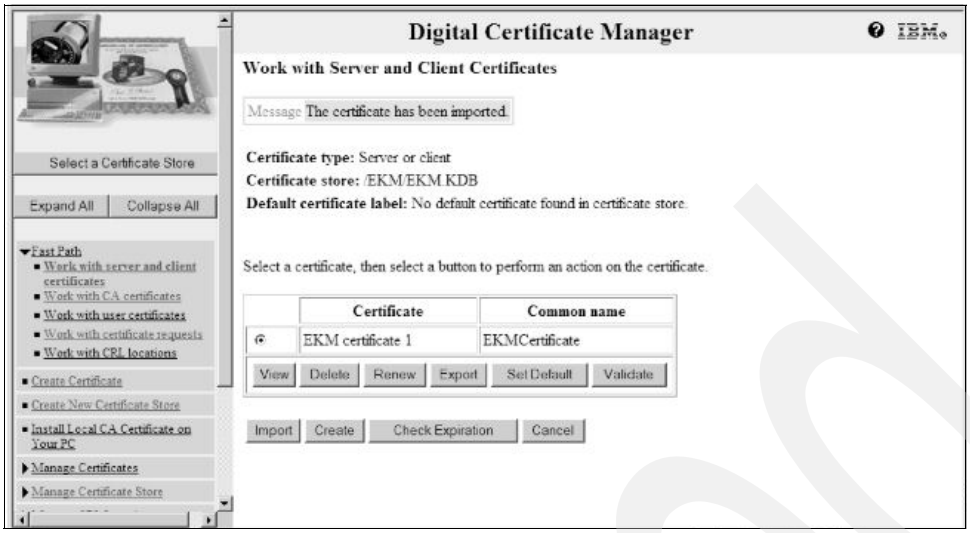


Figure 7-23 Work with server and client certificates

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.

To create a self-signed certificate:

1. Click **Create a Local CA Certificate**. In the Create a Certificate Authority (CA) window, as shown in Figure 7-24, specify a Key size of **1024** and enter the keystore password. Fill in the required certificate information. Click **Continue**.

Figure 7-24 Create a certificate authority (CA)

2. Click **Select Certificate Store**. In the Select Certificate Store window, select **Other System Certificate Store**. Click **Continue**. The Certificate Store Name and Password window opens, as shown in Figure 7-25 on page 260. Specify the path and filename that

you entered in the Create keyring/keystore instance window, as shown in Figure 7-14 on page 254. Click **Continue**.

Figure 7-25 Certificate Store and Password window

3. In the Install Local CA Certificate window, as shown in Figure 7-26, click the **Install certificate** link to install on your browser. Click **Continue**.

Figure 7-26 Install Local CA Certificate

4. The Certificate Authority (CA) Policy Data window opens, as shown in Figure 7-27 on page 261. Click **Yes** to allow creation of user certificates and specify the validity period of

the certificates that are issued by this certificate authority (number of days). Click **Continue.**

Figure 7-27 Certificate authority (CA) Policy Data window

5. The Work with Server and Client Certificates window opens, as shown in Figure 7-28, click **Create**. You can also navigate to this window by the Work with server and client certificates option of the Fast Path menu.

Figure 7-28 Work with Server and Client Certificates

6. The Select a Certificate Authority (CA) window opens, as shown in Figure 7-29 on page 262. Select **Local Certificate Authority (CA)**. Click **Continue**.

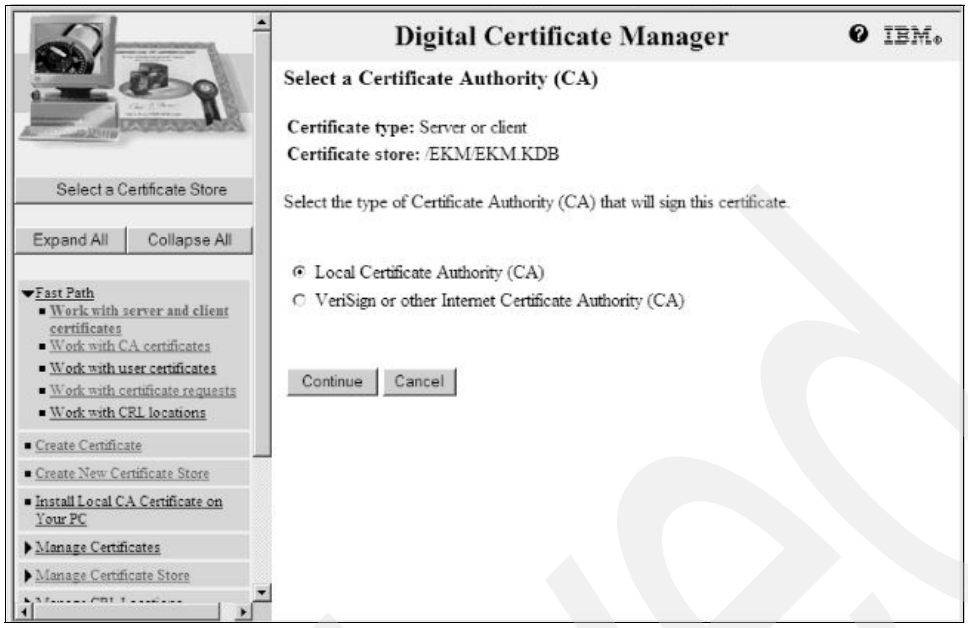


Figure 7-29 Select a Certificate of Authority (CA) window

7. In the Create Certificate window, as shown in Figure 7-30, specify a Key size of **1024** and enter a Certificate label value that corresponds to the alias2 value in your EKM configuration file. Complete the other required fields as appropriate. Click **Continue**.

Figure 7-30 Create Certificate window

The Work with Server and Client Certificates window opens showing the newly created certificate in the list.

Receive the 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. Because this certificate does not have a private key, it must be imported as a CA certificate through DCM.

To receive the certificate:

1. Select **Work with CA certificates** and click **Import** when the Work with CA Certificates window opens.
2. In the Import Certificate Authority (CA) Certificate window, as shown in Figure 7-31, specify the fully qualified path and filename of the certificate that you want to import. Click **Continue**.

Figure 7-31 Import a Certificate Authority (CA) Certificate window

3. Click **Continue**.
4. Enter the password when prompted. Click **Continue**. The Certificate Received window opens, verifying your certificate.

Export private key and certificate

Perform these steps when copying or moving a key, a key and a certificate, or only a certificate from a source keystore.

To export the private key and certificate:

1. Select **Work with Server and Client Certificates**. On the Work with Server and Client Certificates window, shown in Figure 7-32 on page 264, select the desired certificate and click **Export**.

Figure 7-32 Work with Server and Client Certificates window

2. In the Export Destination window, shown in Figure 7-33, select **File - Export to a file** to export the certificate to a file. Click **Continue**.

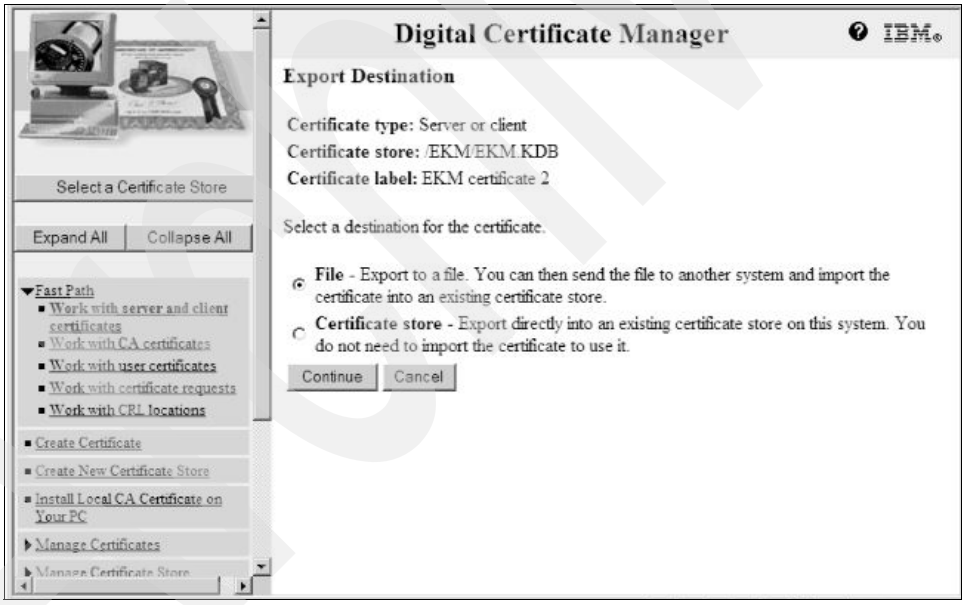


Figure 7-33 Export Destination window

3. In the Export Server or Client Certificate window, shown in Figure 7-34 on page 265, specify a file name to which you want to export the certificate and enter the Password. Click **Continue**.

Figure 7-34 Export Server or Client Certificate window

The Certificate Exported window opens.

Import private key and certificate

Perform these steps when copying or moving a key, a key and a certificate, or only a certificate into a target keystore

To import the private key and certificate:

1. Click **Work with Server and Client Certificates**.
2. In the Work with Server and Client Certificates window, as shown in Figure 7-35, select the desired certificate and click **Import**.

Figure 7-35 Work with Server and Client Certificates window

3. The Import Server or Client Certificate window opens, as shown in Figure 7-36. Specify the fully qualified path and file name of the certificate file to be imported. Click **Continue**.

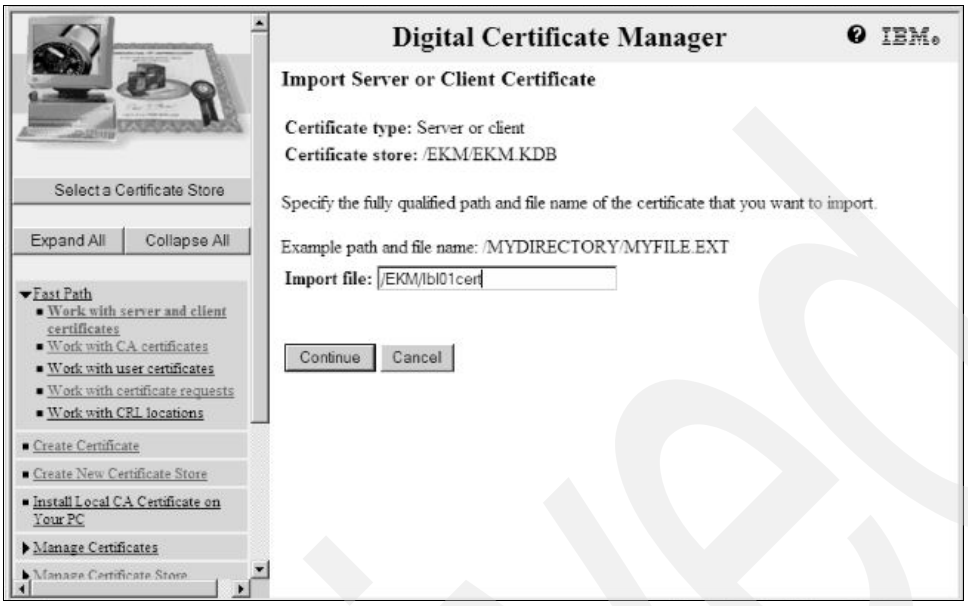


Figure 7-36 Import Server or Client Certificate window

4. Specify the Password when prompted (see Figure 7-37). Click **Continue**.

Figure 7-37 Import Server or Client Certificate password window

The Work with Server and Client Certificates window reopens, showing the imported certificate, as shown in Figure 7-38 on page 267.

Figure 7-38 Work with Server and Client Certificates window showing imported certificate

7.8 ShowPrivateTool

When working with JCEKS, JCE4758KS, and JCECCAKS, no keytool command or **`ikeyman`** command is available to display private key information. Not being able to display private keymaterial can create difficulties when importing keymaterial into backup and secondary keystores.

In Example 7-35 on page 268, we introduce a Java application that takes as input: a keystore file, a keystore, the password of the keystore, and the keystore format.

Format choices are:

- JCEKS
- JKS
- JCE4758KS
- JCECCAKS

Note: The correct providers must be in the `$JAVA_HOME/lib/security/java.security` file when executing this program.

In the example, we first set our command-line arguments. The keystore file is then instantiated into the keystore object using the `getInstance` method with the `stortype` as an argument. We then instantiate a `FileInputStream` object using the keystore file as input. After the keystore object is instantiated and the FileInputStream object is created, we can call the load method from the keystore object, using the `FileInputStream` and keystore password as arguments. This loads keystore database information into memory. After that is accomplished, we simply iterate through all aliases in the keystore to determine whether each alias has private keymaterial associated with it. If all aliases do, we print all the private keymaterial.

Note: If you would like to see more information about the private key, locate the following line:

```
System.out.println("alias: " + alias + "Private Key=True");
```

Replace that line with the following line:

```
System.out.println("alias: " + alias + "Private Key=True" + "\n" +
privKey.toString());
```

However, your results might vary when listing hardware keystores, because the private key in a hardware keystore is protected.

To run this program, you must first compile it using the **javac** command:

```
javac ShowPrivate.java
```

The **javac** command has to be in the $PATH. The **javac** command is located in $JAVA_HOME/bin. After the source is compiled, a ShowPrivate.class file is created. To run the program, execute the command:

```
java ShowPrivate keystore password storetype
```

Example 7-35 Java application to show private key information

```
import java.io.FileInputStream;
import java.security.KeyStore;
import java.security.interfaces.RSAPrivateKey;

public class ShowPrivate {
   static String keystore = null;
   static char[] password = null;
   static String storeType = null;
   static RSAPrivateKey privKey = null;

   static public void main(String[] args) {
      try {
         if (args.length != 3) {
            System.err.println("<keystore> <password> <storetype>");
            System.exit(-1);
         }
         keystore = args[0];
         password = args[1].toCharArray();
         storeType = args[2];
         printPrivate();
      } catch (Throwable t) {
         t.printStackTrace();
         System.err.println("Prog failed. Exiting...");
      }
   }

   static void printPrivate() throws Exception {
      // Load a keystore, with arguments from command line
      KeyStore ks = KeyStore.getInstance(storeType);
      FileInputStream fis = new FileInputStream(keystore);
      ks.load(fis, password);
      fis.close();

      // Iterate through all keys in a keystore
      String alias;
```

```
      for (java.util.Enumeration e = ks.aliases(); e.hasMoreElements();) {
         alias = (String) e.nextElement();

         if (ks.isKeyEntry(alias)) {
            privKey = (RSAPrivateKey) ks.getKey(alias, password);
            if (privKey == null) {
               System.out.println("alias: " + alias + "Private Key=NULL");

            } else{
               System.out.println("alias: " + alias + "Private Key=True");
                           }
         }
      }

   }

}
```

The output from the command is similar to Example 7-36.

Example 7-36 Output from ShowPrivate

```
alias: rsakey1 Private Key=True
```

7.9 MatchKeys tool

The following Java application example shows that a public key and a private key loaded from a keyring do in fact complement each other. The source in Example 7-37 on page 270 works with keyrings of the type JCERACFKS. The source involves changing the RACFInputStream being used to load the keyring from:

```
com.ibm.crypto.provider.RACFInputStream
```

The keyring is loaded to:

```
com.ibm.crypto.provider.hdwrCCA.provider.RACFInputStream.
```

Changing the provider to the correct hardware provider and the keystore type to the corresponding keystore type causes this tool to work with JCECCARACFKS and JCE4758RACFKS.

In Example 7-37 on page 270, first, the program loads the arguments from the command line into string variables. After the arguments are loaded, we can use RACFInputStream to load the keyring into memory. With the keyring in memory, we can access the keyring and load the public and private key objects corresponding to the certificate that we chose at the command line.

Now that we have key objects, we can encrypt a byte stream test message, using the public key. Then, we can decrypt the enciphered byte stream with our private key. If the original clear byte stream matches our decrypted message, we know that the public key and the private key are a matched set. If they do not match, or an exception is thrown, something is wrong with our certificate, and this certificate cannot be used for encrypting data, because we will not be able to decrypt the data.

To run this program, you must first compile it using the **javac** command:

```
javac MatchKeys.java
```

The **javac** command has to be in the $PATH. The **javac** command is located in $JAVA_HOME/bin. After the source is compiled, a ShowPrivate.class file is created. To run the program, execute the command:

java ShowPrivate *personalAlias keyring user ID*

Example 7-37 MatchKeys source

```
import java.security.Key;
import java.security.KeyStore;
import java.security.PublicKey;

import javax.crypto.Cipher;

public class MatchKeys {

   public static void main(String argv[]) {
      String aliasPersonal = "";
      String keyRing = "";
      String userid = "";

      try {
         if (argv.length != 3) {
            System.err.println("aliasPersonal keyring userid");
            System.exit(-1);
         }
         aliasPersonal = argv[0];
         keyRing = argv[1];
         userid = argv[2];
      } catch (Throwable t) {
         t.printStackTrace();
         System.err.println("Prog failed. Exiting...");
      }

      try {

         String keystoreType = new String("JCERACFKS");
         String algorithm = new String("RSA");

         String pass = new String("password");
         byte[] plainText = "testmessage_a very long test message"
               .getBytes("8859_1");

         /* Get certificates and keys from RACF */
         KeyStore keyStore = null;
         String provider = null;
         System.out.println("Using the IBMJCE provider");
         provider = new String("IBMJCE");
         com.ibm.crypto.provider.RACFInputStream ksStream = new
com.ibm.crypto.provider.RACFInputStream(
               userid, keyRing, pass.toCharArray());
         keyStore = KeyStore.getInstance(keystoreType, provider);
         keyStore.load(ksStream, pass.toCharArray());

         /*
          * Get the private and public key associated with the specified
          * alias
```

```
         */
        Key privKey = keyStore.getKey(aliasPersonal, pass.toCharArray());
        System.out.println("Obtained the private key for alias "
              + aliasPersonal + ".");
        PublicKey pubKey = keyStore.getCertificate(aliasPersonal)
              .getPublicKey();
        System.out.println("Obtained the public key for alias "
              + aliasPersonal + ".");

        /* Use the above keys to encrypt and decrypt a message */
        Cipher cp = Cipher.getInstance(algorithm, provider);

        System.out.println("Doing Encrypt");
        cp.init(Cipher.ENCRYPT_MODE, pubKey);
        cp.update(plainText);
        byte[] cipherText = cp.doFinal();

        System.out.println("Doing Decrypt");
        cp.init(Cipher.DECRYPT_MODE, privKey);
        cp.update(cipherText);
        byte[] newPlainText = cp.doFinal();

        System.out.println("Plain Text messages should match below");
        System.out.println("plainText    = "
              + new String(plainText, "8859_1") + "\n"
              + "newPlainText = " + new String(newPlainText, "8859_1"));
      } catch (Exception ex) {
        System.out.println("Unexpected exception: " + ex.getMessage());
        ex.printStackTrace();
      }
    }
}
```

We can see the output from MatchKeys in Example 7-38. Here, we have executed the MatchKeys application on one of our keyrings with a personal certificate in it. We can see from the output that the public key and the private key were retrieved from the certificate, and then a message was successfully encrypted and then decrypted. If the public key does not match the private key in this certificate, the message is not decrypted. This tool can be useful for troubleshooting issues where a private key might not match a public key. This situation can happen when you incorrectly delete certificates and then regenerate the certificate while private key information is still around.

Example 7-38 Output from MatchKeys

```
JBARNEY:/u/jbarney: >java -cp test.jar com.johann.MatchKeys RSA_1024_Cert1 EKMRing
JBARNEY
Using the IBMJCE provider
Obtained the private key for alias RSA_1024_Cert1.
Obtained the public key for alias RSA_1024_Cert1.
Doing Encrypt
Doing Decrypt
Plain Text messages should match below
plainText    = testmessage_a very long test message
newPlainText = testmessage_a very long test message
```

7.10 Hardware cryptography

Using hardware cryptographic services on z/OS, the three ways to secure keymaterial are:

CLEAR Stores the key in an external hardware key structure where clear text is tokenized into a CCA external token. This way has the greatest hardware performance and the lowest hardware security.

PKDS This public-private key data set (PKDS) encrypts the private key with the system master key. The clear text version of this key can never be viewed or retrieved. The key pair is stored in a system key storage area. Compared with CLEAR and RETAINED key types, PKDS has medium hardware security and medium throughput.

RETAINED Stores the private key actual hardware device and is never allowed to be viewed or retrieved in the clear. This hardware type offers the maximum security for keys. It is also the slowest, because cryptographic calls for a specific key pair must go to the hardware device that holds that key pair. When using this hardware type, applications are completely tied to the physical device.

Hardware cryptographic devices on z/OS

Hardware cryptography is the addition of a cryptographic coprocessor such as the IBM PCI-X Cryptographic Coprocessor (PCIXCC). For more information about IBM cryptographic hardware, refer to:

http://www-03.ibm.com/security/cryptocards/

Use of a cryptographic coprocessor can free processing cycles by offloading cryptographic processing to the cryptographic device. In addition, use of a cryptographic device can increase security by storing keys on the hardware itself, because they are never available in the clear, nor when in storage or in use. The devices include:

- PCIXCC:
  - Supported on System z 990 servers and z9
  - Replacement for both the PCICC and the Cryptographic Coprocessor Facility (CCF)
  - Supports DES, TDES, RSA, and SHA-1 cryptographic operations
  - Can perform modular-exponentiation for RSA and DSA
  - System (master) Key and retained key storage
- PCICC:
  - Available on CMOS G5, G6, and System z servers, except System z 990 and z9.
  - Supports DES, RSA, and DSA cryptographic operations.
  - System (master) Key and retained key storage.
- PCICA:
  - Supports DES, RSA, and DSA cryptographic operations
  - SSL Handshake Acceleration

Note: PCICA supports DSA cryptographic operations; RACF does not support DSA keys.

- CP Assist for Cryptographic Function (CPACF):
  - IBM System z 990 servers and z9
  - DES, TDES, MAC, and SHA-1 cryptographic operations
- Cryptographic Coprocessor Facility (CCF):
  - IBM CMOS G5, G6, and System z servers, except the new System z 990 and z9
  - DES, TDES, RSA, and various Finance industry-specific cryptographic operations

Supported hardware cryptographic cards for Java 5.0 on Open Systems

Support for these cards through the IBMPKCS11Impl provider begins after the card, its driver, and any manufacturer's support software have been installed and are functioning properly. Refer any issues regarding installation and configuration of these cards and software to the manufacturer.

The following cards are supported on Windows (32-bit), AIX, Solaris 9 (32-bit and 64-bit, SPARC only), and Linux:

- nCipher nForce 4000 PCI (OB4033P-4K0)
- nCipher nForce 1600 PCI (nC3033P-1k6)
- nCipher nForce 150 PCI (nc3033P-150)
- nCipher nShield 800 PCI (nC4033P-800)
- nCipher nShield 150 SCSI (nc4032W-150) Note: This card is going out of support.
- nCipher nShield 150 SCSI (nF300KM-1c) Note: This card is going out of support.
- nCipher netHSM 1600 (nH1956)
- Eracom Orange (CSA8000)
- SafeNet Luna SA

The following cards are supported on Windows (32-bit), AIX, Solaris 9 (32-bit, 64-bit, SPARC only), and Linux. These specific model cards have not been tested by IBM, but support is assumed, because other cards in the same family have been tested successfully:

- nCipher nForce 300 PCI
- nCipher nForce 400 PCI
- nCipher nForce 400 SCSI
- nCipher nShield 400 SCSI
- nCipher nShield 150 PCI
- nCipher nShield 300 PCI
- nCipher netHSM 300 PCI
- nCipher netHSM 800 PCI

Alias and symmetric key setup for LTO4 encryption (PKCS11ImplKS keystore)

In Example 7-39, we show you to create a range of symmetric keys in a hardware-based PKCS11ImplKS keystore. This example is similar to the example in 7.2, "JCEKS" on page 228 with the exception that a hardware keystore is involved. As opposed to the example using JCEKS, we have to specify the `providerClass` and `providerArg` parameters. Invoke the **`keytool`** with the **`-aliasrange`** option.

Example 7-39 Generating symmetric keys in a PKCS11ImplKS keystore

```
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 just used and the filename under which the symmetric keys were stored. Note that EKM might not start if an invalid alias is specified. Other causes for validation check failure can include incorrect bit size (AES keysize *must* be 256) or an invalid algorithm for the platform. On z/OS, -keyalg must be DESede (3-DES). On distributed, -keyalg must be AES and -keysize must be 256. The filename specified in the config.keystore.file must 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 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, EKM will not start, an audit record will be created, and an error message appears.

Note: If you plan to use a hardware-based keystore for encryption on LTO4 tape drives, check with the hardware vendor for support of symmetric keys. Not all PKCS11ImplKS keystores support symmetric keys.

8

EKM operational considerations

In this chapter, we discuss EKM commands and operational considerations, including the following topics:

- Synchronizing primary EKM configuration data, including drive tables
- Maintenance, including adding and removing drives
- Backup procedures on Open Systems
- Backup procedures on z/OS
- Considerations for backing up keystores for:
  - JCEKS
  - JCERACFKS
  - JCECCARACFKS

We also include a mixed mode data sharing example, in which we describe the steps required to share encrypted tapes with a business partner.

8.1 EKM commands

The following sections discuss the important **sync** command and other EKM commands used with the EKM command line interface.

8.1.1 The EKM sync command and EKM properties file

The **sync** command synchronizes the configuration file properties, drive table information, or both on another EKM server with those on the EKM server issuing the command. The EKM **sync** command creates a Secure Sockets Layer (SSL) connection between two EKMs. The EKM making the request is considered the client, and the EKM being synchronized with is the server. By default, the EKM is set to perform server-only authentication. That is to say, the client checks its trust store to see if the server is trusted. This behavior can be modified by changing the following line:

```
TransportListener.ssl.clientauthentication  =  0
```

Changing the `0` to `1` modifies the EKM behavior to the setting `want client auth`, which means that the server requests the client's certificate for verification; if it does not send it, it still tries to establish an SSL connection. Setting the `0` to `2` changes the behavior to the setting `need client auth`, which means that the server requests the client's certificates. If the server does not get them or does not verify them, it ceases SSL handshaking, and synchronization does not happen.

Note: Make sure to change the line (`TransportListener.ssl.clientauthentication`) on the EKM acting as the server during the sync.

In the EKM properties file, the following lines set up the SSL configuration from a keystore point of view:

- Admin.ssl.keystore.name = /keymanager/testkeys
- Admin.ssl.truststore.name = /keymanager/testkeys
- TransportListener.ssl.ciphersuites = JSSE_ALL
- TransportListener.ssl.clientauthentication = 0
- TransportListener.ssl.keystore.name = /keymanager/testkeys
- TransportListener.ssl.protocols = SSL_TLS
- TransportListener.ssl.truststore.name = /keymanager/testkeys

The SSL server setup is done with the Admin.ssl directives, and SSL client setup is done with the TransportListener.ssl directives. An EKM can act as either a client or a server when performing **sync** commands.

Assuming `clientauthentication` is set to `2`, we perform an SSL handshake with `Server Needs Authentication`, and the following chain of events occurs:

1. The EKM acting as SSL server sends certificates from its `Admin.ssl.keystore` to the client requesting a sync.
2. The EKM acting as client then searches its `TransportListener.ssl.truststore` for matching certificates to validate whether it trusts the server.
3. If it does, the server then requests that the client sends its certificates from the `TransportListener.ssl.keystore`.
4. The server searches its `Admin.ssl.truststore` for matching certificates.
5. Both server and client verify that they trust each other.

6. They then negotiate a common cipher from `TransportListener.ssl.ciphersuites`.
7. Finally, the requested **`sync`** command happens.

Refer to “sync” on page 281 for the detailed command syntax.

8.1.2 EKM command line interface and command set

EKM provides a command-line interface command called **`KMSAdminCmd`** that includes the command set described in the following sections. For more information, see:

http://www.fdesecurityleaders.com/files/ibm_encryption_key_manager_component_for_the_java.pdf

adddrive

Use the **`adddrive`** command to add a new drive to the EKM drive table, or you can use the EKM **`acceptunknown`** function to have the drives added automatically. Example 8-1 shows the **`adddrive`** command.

Example 8-1 The adddrive command

```
adddrive -drivename drivename [ -rec1 alias -rec2 alias]

-drivename     Specifies the 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.

Example: adddrive -drivename 000123456789 -rec1 alias1 -rec2 alias2
```

deletedrive

Use the **`deletedrive`** command to delete a drive from EKM drive table. Equivalent commands are **`deldrive`** and **`removedrive`**. Example 8-2 shows the **`deletedrive`** command.

Example 8-2 The deletedrive command

```
deletedrive -drivename drivename

-drivename     Specifies the serial number of the drive to be added.

Example: deletedrive -drivename 000123456789
```

exit

Use the **`exit`** command to exit the EKM **`admin`** command and stop the EKM server. An equivalent command is **`quit`**. Example 8-3 shows the **`exit`** command.

Example 8-3 The exit command

```
exit
```

export

Use the **`export`** command to export a drive table or configuration file to the specified URL. Example 8-4 on page 278 shows the **`export`** command.

Example 8-4 The export command

```
export {-drivetab|-config} -url urlname

-drivetab   Specifies to export the drive table.
-config     Specifies to export the configuration file.
-url        urlname specifies the location where the file is to be written.

Example: export -drivetab -url FILE:///keymanager/data/export.tabl
```

help

Use the **help** command to display EKM command line interface command names and syntax. An equivalent command is the question mark character (**?**). Example 8-5 shows the **help** command.

Example 8-5 The help command

```
help
```

import

Use the **import** command to import a drive table or a configuration file from a specified URL. Example 8-6 shows the **import** command.

Example 8-6 The import command

```
import {-merge|-rewrite} {-drivetab|-config} -url urlname

-merge      Specifies to merge the new data with the current data.
-rewrite    Specifies to replace the current data with new data.
-drivetab   Specifies to import the drive table.
-config     Specifies to 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

Use the **list** command to list certificates contained in the keystore named by the config.keystore.file property. Example 8-7 shows the **list** command.

Example 8-7 The list command

```
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    Specifies to display more information about the certificates.

Example: list -alias mycert -v
```

listcerts

Use the **listcerts** command to list certificates contained in a keystore named by config.keystore.file property. Example 8-8 on page 279 shows the **listcerts** command.

Example 8-8 The listcerts command

```
listcerts [-alias alias -verbose |-v]

-alias        alias specifies a specific certificate to list.
-verbose|-v   Specifies to display more information about the certificates.

Example: listcerts -alias alias1 -v
```

listconfig

Use the **listconfig** command to list the EKM configuration file properties (Example 8-9).

Example 8-9 The listconfig command

```
listconfig
```

listdrives

Use the **listdrives** command to list the drives in the drive table (Example 8-10).

Example 8-10 The listdrives command

```
listdrives [-drivename drivename -verbose |-v]]

-drivename       drivename specifies the serial number of the tape drive to list.
-verbose|-v      Specifies to display more information about the tape drives.

Example: listdrives -drivename 000123456789
```

logout

Use the **logout** command to log off the current user (Example 8-11). Equivalent command is **logoff**. These commands are only useful when the LoginModule is enabled.

Example 8-11 The logout command

```
logout
```

modconfig

Use the **modconfig** command to modify a configuration file property (Example 8-12). Equivalent command is **modifyconfig**.

Example 8-12 The modconfig command

```
modconfig {-set | -unset} -property name -value value

-set          Specifies to set the specified property to the specified value.
-unset        Specifies to remove the specified property.
-property     name specifies the name of the property.
-value        value specifies the new value for the property when -set is
               specified.

Example: modconfig -set -property sync.timeinhours -value 24
```

moddrive

Use the **moddrive** command to modify drive information in the drive table (Example 8-13). An equivalent command is **modifydrive**.

Example 8-13 The moddrive command

```
moddrive -drivename drivename [ -rec1 alias -rec2 alias]

-drivename    Specifies the serial number of the 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.

Example: moddrive -drivename 000123456789 -rec1 newalias1
```

refresh

Use the **refresh** command to have EKM refresh the debug, audit, and drive table values with the latest configuration parameters (Example 8-14).

Example 8-14 The refresh command

```
refresh
```

refreshks

Use the **refreshks** command to refresh the keystore (Example 8-15). Use this command to reload the keystore specified in config.keystore.file if it was modified while the EKM server was running. Use this command only when needed, because it can degrade performance.

Example 8-15 The refreshks command

```
refreshks
```

startekm

Use the **startekm** command to start the EKM server (Example 8-16).

Example 8-16 The startekm command

```
startekm
```

status

Use the **status** command to display the status of whether the EKM server is started or stopped (Example 8-17).

Example 8-17 The status command

```
status
```

stopekm

Use the **stopekm** command to stop the EKM server (Example 8-18).

Example 8-18 The stopekm command

```
stopekm
```

sync

Use the **sync** command to synchronize the configuration file properties, drive table information, or both on another EKM server with those on the EKM server issuing the command (Example 8-19). For more information, refer to 8.1.1, “The EKM sync command and EKM properties file” on page 276.

Example 8-19 The sync command

```
sync {-all | -config | -drivetab} -ipaddr ip_addr:ssl:port [-merge | -rewrite]

-all        Sends both the configuration file properties and the drive table
             information to the other EKM server.
-config     Sends only the configuration file properties to the other EKM server.
-drivetab   Sends only the drive table information to the other EKM server.
-ipaddr     ip_addr:ssl:port specifies the address of the other EKM server.
-merge      Specifies to merge new drive table data with current data. (The
             configuration file is always a rewrite.) This is the default.
-rewrite    Specifies to replace the current data with new data.

Example: sync -drivetab -ipaddr remoteekm.ibm.com:443 -merge
```

Important: If your EKM setup uses Shared HFS function for UNIX Systems Services *and* you use shared directories for debug and error logs, we recommend that you *do not use* the **sync -all** or **sync -config** command, because these commands change the log settings on synchronized systems to use the same directories.

Use only the **sync -drivetable** command for this type of configuration.

version

Use the **version** command to display the version of the EKM server (Example 8-20).

Example 8-20 The version command

```
version
```

8.2 Backup procedures

Maintaining primary and secondary EKM servers is desirable for maximum availability of encrypted backup and recovery.

8.2.1 EKM file system backup

You must save the EKM and its associated data regularly without encryption. If the keystore password is specified on the strEKM script call (and not stored in the `KeyManagerConfig.properties` file), you must keep a copy of the password in a secure location. The keystore password must be available to recover the EKM. Encrypted save or archive operations must not be performed on the partition or system where the EKM server is running. If data on the system where the EKM is running is encrypted, the EKM cannot be recovered without the availability of a secondary EKM server.

If the keystore password is entered into EKM through a script (that is, the EKM config file does not contain the keystore password), when the EKM is backed up, the files (configuration file, drive table, and keystore backup file) do not need to necessarily be treated as secret.

However, the script that contains the keystore password must be stored securely and resiliently (for example, multiple copies in multiple locations). The keystore password is confidential information and must be treated as confidential information.

Backing up the script file securely has the same options that exist for backing up the configuration file that contains the keystore password. But the scripts might be backed up and stored/transmitted secretly and separately from the EKM backup files, which adds an additional level of security. Finally, we must emphasize that however the keystore password is stored (in a script or in the EKM's configuration file), it must be stored securely and resiliently so that the keystore password can always be recovered. Loss of all copies of the keystore password causes loss of all of the keys in the keystore, and there is no recovery path for this situation.

Example 8-21 shows a sample job to back up a z/OS aggregate.

Example 8-21 Job to back up a z/OS aggregate

```
//ZFSBKUP1 JOB (OS390),'PROGRAMMER',CLASS=A,
// MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//* THIS JOB QUIESCES A ZFS AGGREGATE, DUMPS IT, THEN UNQUIESCES IT.
//*-------------------------------------------------------------------
//DUMP    EXEC PGM=ADRDSSU,REGION=4096K
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//OUT      DD  DSN=hlq.AGGR004.BACKUP,
//             DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(5,1),RLSE)
//SYSIN    DD  *
DUMP DATASET(INCLUDE(hlq.ZFS.AGGR004)) -
      CONCURRENT -
      OUTDD(OUT)
```

The zFS aggregate can be restored using DFSMSdss logical restore. It is restored into a new aggregate (in this case, OMVS.PRV.AGGR005.LDS0005) if the original aggregate (in this case, hlq.ZFS.AGGR004) still exists. Example 8-22 shows a restore job.

Example 8-22 Job to restore a zFS aggregate

```
//ZFSREST1 JOB (OS390),'PROGRAMMER',CLASS=A,
//             MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------------------------
//* THIS JOB RESTORES A ZFS AGGREGATE.
//*-------------------------------------------------------------------
//ZFSREST EXEC PGM=ADRDSSU,REGION=0M
//SYSPRINT DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//INDS     DD  DISP=SHR,DSN=SUIMGUR.ZFS.DUMP1
//SYSIN    DD  *
       RESTORE DATASET(INCLUDE(**)) -
               CATALOG -
               RENAMEU( -
                (hlq.ZFS.AGGR004, -
                   OMVS.PRV.AGGR005.LDS0005) -
                      ) -
               WRITECHECK -
               INDD(INDS)
```

After the aggregate is restored, you must perform the following steps for a compatibility mode aggregate:

1. Unmount the original aggregate (in this case, `hlq.ZFS.AGGR004`) if it still exists (this also detaches it).
2. Mount the file system in the restored aggregate (in this case, `OMVS.PRV.AGGR005.LDS0005`).

After the aggregate is restored, you must perform the following steps for a multi-file system aggregate:

1. Unmount the file systems in the original aggregate (if any are mounted).
2. Detach the original aggregate (in this case, `hlq.ZFS.AGGR004`) if it still exists.
3. Attach the restored aggregate (in this case, `OMVS.PRV.AGGR005.LDS0005`).
4. Mount the file systems in the restored aggregate.

Another example of a logical restore of a zFS aggregate using DFSMSdss by replacing the existing aggregate is shown. The backup is restored into the original aggregate (in this case, `hlq.ZFS.AGGR004`). The aggregate cannot be mounted (or attached) during the restore operation. Example 8-23 is an example of a *restore replace* job.

Example 8-23 Job to restore a zFS aggregate with replace

```
//ZFSREST2 JOB (OS390),'PROGRAMMER',CLASS=A,
// MSGCLASS=X,MSGLEVEL=(1,1)
//*-------------------------------------------------
//* THIS JOB RESTORES A ZFS AGGREGATE.
//*-------------------------------------------------
//ZFSREST EXEC PGM=ADRDSSU,REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//INDS DD DISP=SHR,DSN=SUIMGUR.ZFS.DUMP1
//SYSIN DD *
       RESTORE DATASET(INCLUDE(hlq.ZFS.AGGR004)) -
               CATALOG -
               REPLACE  -
               WRITECHECK -
               INDD(INDS)
```

DFSMShsm processes zFS data sets. The zFS data sets are VSAM linear data sets (LDS) that provide a function similar to HFS data sets. DFSMShsm does not process individual file systems within a zFS data set.

8.2.2 Identifying DFSMShsm to z/OS UNIX System Services

If you plan to use DFSMShsm to back up HFS or zFS data sets mounted by z/OS UNIX System Services, you have to define DFSMShsm to z/OS UNIX System Services as a super user. This action provides the required authorization to quiesce or unquiesce a file system. DFSMShsm must have a RACF user ID associated with it. That user ID must have a default RACF group, which has an OMVS segment with a group id (GID). The user ID must also have an OMVS segment with the following parameters:

```
UID(0) HOME('/')
```

For additional information, refer to *z/OS UNIX System Services Planning*, GA22-7800.

8.2.3 Keystore backup

The maintenance, backup, and restoring of key and certificate information depend on which keyring/keystore implementation is used. One method is:

- Create copies of the keystores that the EKM will use.
- Retain a PKCS12 format file for *each* key-certificate combination and store this file in a secure location (for example, on read-only media in a locked cabinet).
- Retain a copy of the PKCS12 format and the keystore at your disaster recovery (DR) sites.

This method allows keystores to be re-created if absolutely necessary.

Note: Simply backing up the keystore file does not work with keystore types JCE4758KS and JCECCAKS. These keystores can store private key material in a protect PKDS. Follow the PKDS backup and restoration procedures in 8.3, "ICSF disaster recovery procedures" on page 286 in addition to backing up the keystore file for these types of keystores.

8.2.4 RACF

In this section, we describe how to copy a RACF database to prepare it for backup. Refer to *z/OS V1R8.0 Security Server RACF System Programmer's Guide,* SA22-7681, for more details about RACF, the IRRUT200 utility, and the IRRUT400 utility.

As a general rule, use IRRUT200 to create a database copy if the output database is the same size and on a device with the same track geometry as the input database. However, if you need to produce a copy of a database of a different size from your original database or on a different device type (for example, 3390 to 3380), you must use IRRUT400.

In cases where IRRUT200 has detected errors on upper level blocks only, or an analysis of IRRUT200's BAM block mappings has shown that significant fragmentation has occurred, use IRRUT400 to perform the copy. When IRRUT400 copies a database, it rebuilds the database, re-creating upper level index blocks and reorganizing profiles to eliminate fragmentation. The profile reorganization makes all the segments of a single profile (for example, a user profile's base, TSO, and CICS® segments) contiguous.

The reorganization that IRRUT400 performs can improve performance by reducing the number of database reads required to read profiles. As a profile is updated over time, its segments are likely to be written to different physical blocks in the database. You can see this by looking at the output of the IRRUT200 INDEX FORMAT function and noting the relative byte address (RBA) of each profile segment. RACF reads the database one 4K block at a time, so the fewer the number of 4K blocks that a profile's segments are spread across, the fewer the number of reads required to access all of them and the better the performance of RACF functions that require database profile access.

For RACF databases consisting of multiple data sets, one IRRUT400 invocation can process one or more of the data sets.

The target of the copy cannot be an active RACF database. If you specify an active primary or backup data set on the system on which IRRUT400 is running, the utility fails. If you need to refresh an active RACF database, use RVARY to deactivate the database before running IRRUT400. After utility processing completes, use RVARY to activate the database.

You can copy an active RACF database, but if you do, you must either specify LOCKINPUT or guarantee that no updates occur to the input data sets from any system.

The three ways to copy an active database using this utility are:

- Specify the LOCKINPUT parameter.

  This method is preferred. It creates an accurate output database and guarantees that no information is lost before you are able to use the new copy as your active database. Using the LOCKINPUT parameter stops you from writing information, other than statistical updates, to the input database. If you attempt to write to the database while IRRUT400 is running, RACF generates ABEND483 RC50 or ABEND485 RC50 errors.

  Attempts to write to the database result from explicit commands, such as RALTER, and also from a specific logon attempt. For instance, a logon causes a write to the database and fails if:

  - This is your first logon of the day, and RACF is not in data sharing mode.
  - The password is being changed.
  - You are entering the correct password after previously entering an incorrect password.

  If the LOCKINPUT keyword is specified, you are unable to update the input data sets after the execution of this utility. LOCKINPUT leaves the input database locked to prevent any updates to the input database. If the input database was unlocked when IRRUT400 completed, it might get updated and, therefore, be out of sync with the new copy. If you do not want to switch to the new copy, you must invoke IRRUT400 again, this time with the UNLOCKINPUT parameter, to unlock the input database so that it can be updated.

- Specify the NOLOCKINPUT parameter.

  Specifying this parameter does not prevent you from updating the input database:

  - If updates occur to the input database *during* the copy operation, the results of the utility and the content of the output database are unpredictable. The updates might be successful, an abend might occur, or the output database might become corrupted.
  - If updates occur to the input database *after* the copy completes, the output database is complete and consistent. However, it does not reflect any of the updates you made to the input database. If you plan to use the output database and want to avoid losing information, be sure that no changes are made between the time that you make the copy and the time RACF begins using it.

- Use IRRUT200 first, then use IRRUT400, in a two-stage process:

  - Stage 1: Use IRRUT200

    Use IRRUT200 to make a copy of a data set from the input database. This copy must be the same size and on a device with the same geometry as the input data set. You can use IRRUT200 only to copy one data set at a time. If the RACF database is comprised of three data sets, for example, you must invoke the utility three times to copy all the data sets. Because IRRUT200 uses ENQ or RESERVE serialization while it copies a data set, updates to the data set are delayed briefly until the copy is completed.

  - Stage 2: Use IRRUT400

    Use IRRUT400 against the new copy of the database. You can specify the NOLOCKINPUT parameter, because the copy is not an active RACF database. This option avoids the errors that are possible by using the first option and avoids the unpredictable results that might occur by using the second option. However, to avoid losing information, you must be sure that no changes are made between the time that you make the copy and the time that RACF begins using it.

  If you have a split database, you must not issue any user or group administration commands until all the IRRUT200 copies are complete. Issuing these commands can cause inconsistencies between user and group profiles on the IRRUT400 output database.

8.3 ICSF disaster recovery procedures

This section outlines the steps required to restore IBM Integrated Cryptographic Service Facility (ICSF) services in the event that the Cryptographic Domains have been zeroed out. This can occur when performing a Disaster Recovery/Business Continuity test, in case of a hardware outage or upgrade, or of a real disaster recovery.

8.3.1 Key recovery checklist

Depending on the way that your organization secures keys or key parts, recovering cryptographic keys might involve multiple people and key parts. Before starting the recovery, all required parties must be contacted and have their key parts available. Then, the required parties must obtain a secure connection to the z/OS systems, which will have their cryptographic services restored. At this point, the recovery can proceed.

The following list summarizes the required steps:

1. Verify that the corresponding CDKS/PKDS/key part files or data are available.
2. Confirm that all required people are available and have retrieved their key part. The key part can be stored on paper, a USB device, or any other storage medium.
3. Disable PKA services and PKDS read/write access.

   In a Sysplex environment, this must be done on each logical partition (LPAR) in the Sysplex before continuing to the next step.
4. Update the PKDS:
   - Enter the SYM-MK key parts.
   - Set the SYM-MK.
   - Enter the ASYM-MK key parts.
   - Activate the PKDS.

   In a Sysplex environment, updating must be done on each LPAR in the Sysplex before continuing to the next step.
5. Complete the recovery:
   - Enable all services and PKDS read/write updates.
   - Verify cryptographic services are available.

   In a Sysplex environment, this must be done on each LPAR in the Sysplex.

8.3.2 Prerequisites

The key or key part owners must know which CKDS and PKDS they will be recovering before retrieving their keys or key parts. Review the documentation of your environment to determine which LPARs need to have keys changed.

Suggestions for naming conventions for the CKDS, PKDS, and key part files:

CKDS: *hlq*.CSF.*sysplex*.D*yymmdd*.CSFCKDS
PKDS: *hlq*.CSF.*sysplex*.D*yymmdd*.CSFPKDS
SYM-MK: SYM-MK.*sysplex*.*part*.D*yymmdd*.txt
ASYM-MK: ASYM-MK.*sysplex*.*part*.D*yymmdd*.txt

In the naming conventions, descriptions are:

- *hlq* is your system high-level qualifier.
- *sysplex* is the name of the Sysplex.
- *part* can be FIRST, MIDDLE, or FINAL.
- *yymmdd* is a six-character creation date.

Note that this example assumes that the symmetric and asymmetric key parts are stored as text files (.txt).

8.3.3 Pre-key change: All LPARs in the Sysplex

Be sure the correct CKDS/PKDSs are available and that the installation options data set points to the right CKDS/PKDS. The initial system check must be done on each system in the Sysplex before continuing on to 8.3.6, “Entering Master Keys for all LPARs in the Sysplex” on page 291. Refer to your environment’s documentation for a list of all LPARs in the Sysplex.

Verify which CKDS and PKDS are available for recovery

To verify which CKDS and PKDS are available for recovery:

1. From the main ISPF panel (Figure 8-1), enter option `3 Utilities`.

```
                        - ISPF Primary Options Menu-
OPTION ==>  3
             --- My Options ---                    .----- All The Options -----.
LOG  SPF Options          CP   Copy/Move           !  ----- Top Of Data ------ !
1    Browse               DU   Dataset Utility     !  0    Settings            !
2    Edit                 LU   Library Utility     !  1    View                !
3    Utilities            TE   Dialog Test         !  2    Edit                !
4    SPF Foreground       V    VTOC Utility        !  3    Utilities           !
5    SPF Background                                !  4    Foreground          !
6    TSO                                           !  5    Batch               !
7    Tutorial                                      !  6    Command             !
SD   SDSF                                          !  7    Dialog Test         !
                                                   !  8    LM Facility         !
                                                   !  9    IBM Products        !
                                                   !  10   SCLM                !
                                                   '---------------------------'
                                                   ****************************
                                                   *     Workbench Options    *
                                                   * XX   Change Colours/Title *
                                                   * YY   Set Standard Options *
                                                   * ZZ   Use Personal Options *
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
```

Figure 8-1 ISPF Primary Options Menu panel

2. From the Utility Selection Panel, shown in Figure 8-2 on page 288, enter option `4 Dslist utility`.

```
 Menu  Help
──────────────────────────────────────────────────────────────────────────────

                          Utility Selection Panel

1  Library     Compress or print data set.  Print index listing.  Print,
                 rename, delete, browse, edit or view members
2  Data Set    Allocate, rename, delete, catalog, uncatalog, or display
                 information of an entire data set
3  Move/Copy   Move, or copy members or data sets
4  Dslist      Print or display (to process) list of data set names.
                 Print or display VTOC information
5  Reset       Reset statistics for members of ISPF library
6  Hardcopy    Initiate hardcopy output
7  Transfer    Download ISPF Client/Server or Transfer data set
8  Outlist     Display, delete, or print held job output
9  Commands    Create/change an application command table
11 Format      Format definition for formatted data Edit/Browse
12 SuperC      Compare data sets                          (Standard Dialog)
13 SuperCE     Compare data sets Extended                 (Extended Dialog)
14 Search-For  Search data sets for strings of data       (Standard Dialog)
15 Search-ForE Search data sets for strings of data Extended (Extended Dialog)
Option ===>   4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
```

Figure 8-2 Utility Selection Panel

3. From the Data Set List Utility panel, shown in Figure 8-3, enter the high-level qualifier of the CKDS or PKDS, such as MYSYS.TST.TESTPLEX*.

```
 Menu  RefList  RefMode  Utilities  Help
──────────────────────────────────────────────────────────────────────────────

                        Data Set List Utility
                                                                More:     +
   blank Display data set list               P Print data set list
       V Display VTOC information           PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . . MYSYS.TST.TESTPLEX.*
   Volume serial  . .

Data set list options
   Initial View . . . 1  1. Volume       Enter "/" to select option
                         2. Space        /  Confirm Data Set Delete
                         3. Attrib       /  Confirm Member Delete
                         4. Total        /  Include Additional Qualifiers
                                         /  Display Catalog Name

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
Option ===> 4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure 8-3 Data Set List Utility panel

4. From the panel, shown in Figure 8-4 on page 289, verify that the correct CKDS and PKDSs exist on this system.

```
  Menu  Options  View  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
DSLIST - Data Sets Matching MYSYS.TST.TEXTPLEX                  Row  1 of  1

Command - Enter "/" to select action                Message          Volume
 -----------------------------------------------------------------------------
         MYSYS.TST.TESTPLEX.CKDS                                      *VSAM*
         MYSYS.TST.TESTPLEX.CKDS.DATA                                 TSOE01
         MYSYS.TST.TESTPLEX.CKDS.INDEX                                TSOE01
         MYSYS.TST.TESTPLEX.PKDS                                      *VSAM*
         MYSYS.TST.TESTPLEX.PKDS.DATA                                 TSOE22
         MYSYS.TST.TESTPLEX.PKDS.INDEX                                TSOE22
****************************** End of Data Set list ***************************









Command ===>                                                Scroll ===> CSR
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Figure 8-4 DSLIST listing

8.3.4 Check the ICSF installation options data

To check the options data:

1. From the previous panel, go back one panel by pressing F3 (Figure 8-4). You see the panel shown in Figure 8-3 on page 288. Enter the high-level qualifier of the installation options data set and press Enter.
2. From the menu shown in Figure 8-5, select the installation options data set for editing by placing an E to the left of it.

```
  Menu  Options  View  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
DSLIST - Data Sets Matching SYS1.TEST.PARMLIB                   Row  1 of  1

Command - Enter "/" to select action                Message          Volume
 -----------------------------------------------------------------------------
E        SYS1.TEST.PARMLIB                                           TSOE01
****************************** End of Data Set list ***************************









Command ===>                                                Scroll ===> CSR
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Figure 8-5 Selecting the installation options

3. In the data set (Figure 8-6), verify that the installation options data is correct for the given system. Verify the CKDSN and PKDSN are pointing to the correct CKDS and PKDS.

```
****************************** Top of Data ******************************
/*                                                                     */
/*     LICENSED MATERIALS - PROPERTY OF IBM                            */
/*                                                                     */
/*     "RESTRICTED MATERIALS OF IBM"                                   */
/*     5694-A01                                                        */
/*                                                                     */
/*     (C) COPYRIGHT IBM COPR. 1990, 2003                              */
/*                                                                     */
/*     STATUS = HCR770A                                                */
/*                                                                     */
CKDSN(MYSYS.TST.TESTPLEX.CKDS)
PKDSN(MYSYS.TST.TESTPLEX.PKDS)
COMPAT(NO)
SSM(NO)
KEYAUTH(NO)
CKTAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

Figure 8-6 Verifying the installation options

8.3.5 Disable all services

To disable all services:

1. Go to the Integrated Cryptographic Service Facility (ICSF) main panel.
2. From the main ICSF panel, shown in Figure 8-7, choose option `4 ADMINCNTL`. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
```

Figure 8-7 ICSF panel

3. Disable all of the services by selecting each of them with a `D` (for disable) as shown in Figure 8-8.

   A status message appears in the upper right corner of the panel indicating whether the services have been stopped.

```
------------------- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

         Active CKDS: MYSYS.TST.TESTPLEX.CKDS
         Active PKDS: MYSYS.TST.TESTPLEX.PKDS


 To change the status of a control, enter the appropriate character
 (E - ENABLE, D - DISABLE) and press ENTER.


   FUNCTION                                          STATUS
   --------                                          ------
d  Dynamic CKDS Access                               ENABLED
d  PKA Callable Services                             ENABLED
d  PKDS Read Access                                  ENABLED
d  PKDS Write, Create, and Delete Access             ENABLED
******************************Bottom of data ********************************
```

Figure 8-8 ICSF Administrative Control panel

4. After this is done for each system in the Sysplex, continue to the next section, 8.3.6, “Entering Master Keys for all LPARs in the Sysplex” on page 291.

8.3.6 Entering Master Keys for all LPARs in the Sysplex

After you have completed the steps described in the previous section for all LPARs in the Sysplex, you may start with the tasks outlined in the following sections.

Enter SYM-MK Master Key parts

These steps must be done by Master Key part owners:

1. From the panel shown in Figure 8-8, return to the main ICSF panel by clicking F3. Select option `1 COPROCESSOR MGMT` as shown in Figure 8-9 on page 292. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure 8-9 ICSF panel

2. Select the processors to set the Master Key on with the E option (for ENABLE) and press Enter. See Figure 8-10.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR       SERIAL NUMBER   STATUS
   --------          -------------   ------
e  X00               77712345        ACTIVE
e  X01               77712346        ACTIVE
e  X02               77712347        ACTIVE
e  X03               77712348        ACTIVE
******************************Bottom of data ********************************
```

Figure 8-10 ICSF Coprocessor Management panel

3. Retrieve the key or key part.
4. In the panel shown in Figure 8-11 on page 293, enter SYM-MK, the key part (FIRST, MIDDLE, or FINAL), the checksum, and the key value. Note that the key registers are both empty. Press Enter and you see a status message in the upper right corner indicating the key part has been loaded. Check to be sure that the Verification Pattern (VP) and Hash Pattern (HP) match what is in the Master Key part document.

```
---------------------- ICSF - Clear Master Key Entry ------------------------
COMMAND ===>
              Symmetric-keys new master key register          :  EMPTY
              Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> SYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST           (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> B7

   Key Value ===> 023457CB6E20537B
             ===> 5ED689736749ADF2
             ===> 0000000000000000   (ASYM-MK only)


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure 8-11 ICSF Clear Master Key Entry panel

5. Go back to the beginning of this section (“Enter SYM-MK Master Key parts” on page 291).
6. Enter SYM-MK Master Key parts and repeat the same instructions by each key part owner. For the previous step, enter either MIDDLE or FINAL for the key part as appropriate. Ensure the FINAL key part is indeed entered last.

Set the SYM-MK

After you have completed all steps in the previous section, set the SYM-MK:

1. Return to the main ICSF panel and select option `2 MASTER KEY` as shown in Figure 8-12. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 2
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure 8-12 ICSF panel: Select option 2 MASTER KEY

2. Next, select option `2 SET MK` and press Enter (see Figure 8-13). A message appears in the upper right corner indicating whether the set was successful.

```
------------------------- ICSF - Master Key Management   ------------------
OPTION ===> 2

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initializw a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure 8-13 ISCF Master Key Management: Select option 2 SET MK

Enter ASYM-MK key parts

This section must be performed by Master Key part owners:

1. Go to the ICSF Main panel. From the main ICSF panel, choose option `1 COPROCESSOR MGMT` as shown in Figure 8-14. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure 8-14 Integrated Cryptographic Service Facility: Select COPROCESSOR MGMT

2. Select the coprocessors to set the Master Key on with the `E` option (for ENABLE) as shown in Figure 8-15 on page 295. Press Enter.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR       SERIAL NUMBER   STATUS
   --------          -------------   ------
e  X00               77712345        ACTIVE
e  X01               77712346        ACTIVE
e  X02               77712347        ACTIVE
e  X03               77712348        ACTIVE
*******************************Bottom of data ********************************
```

Figure 8-15 ICSF Coprocessor Management panel

3. In the panel shown in Figure 8-16, enter ASYM-MK, key part (FIRST, MIDDLE, or FINAL), the checksum, and the key value. Note that the key registers are both empty. Press Enter and you see a status message in the upper right corner indicating the key part has been loaded. Check to be sure that the Verification Pattern (VP) and Hash Pattern (HP) are the same as what is recorded in your key part.

```
---------------------- ICSF - Clear Master Key Entry ------------------------
COMMAND ===>
              Symmetric-keys new master key register          :  EMPTY
              Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> ASYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST            (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> BD

   Key Value ===> 123DE98DA620537B
             ===> 5ED58392E04FBDF2
             ===> 5739EA8926375995    (ASYM-MK only)


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure 8-16 ICSF Clear Master Key Entry panel

4. Go back to the beginning of this section, “Enter SYM-MK Master Key parts” on page 291 and repeat the same instructions by each key part owner. For step 3, enter either MIDDLE or FINAL for the key part as appropriate. Ensure the FINAL key part is indeed entered last.

Activate the PKDS

After all of the key parts have been entered, activate the new PKDS:

1. From the main ICSF panel, enter option 2 SET MK as shown in Figure 8-13 on page 294.
2. Enter option **7** ACTIVATE PKDS to activate the new PKDS. See Figure 8-17 on page 296.

```
------------------------- ICSF - Master Key Management   ------------------
OPTION ===> 7

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initialize a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure 8-17 ICSF Master Key Management panel

3. Enter the new PKDS data set as shown in Figure 8-18.

```
----------------- ICSF - Activate PKA Cryptographic Key Data Set --------------
COMMAND ===>

Enter the name of the new PKDS below.

   New PKDS  ===> 'MYSYS.TST.TSTPLEX.PKDS'



Press ENTER to activate the PKDS.
Press END   to exit to the previous menu.
```

Figure 8-18 ICSF Activate PKA Cryptographic Key Data Set panel

8.3.7 Post-key change for all LPARs in the Sysplex

The procedures in this section must be completed on all LPARs in the Sysplex after the CKDS and PKDS have been activated on all LPARs.

Enable all services and PKDS read and write access:

1. Return to the main ICSF panel and select option `4 ADMINCNTL` as shown in Figure 8-19 on page 297.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure 8-19 Integrated Cryptographic Service Facility: Selecting ADMINCNTL

2. Select the services that you want to enable by entering E as shown in Figure 8-20. A message appears in the upper right corner indicating whether the changes were successful.

```
------------------ ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                 SCROLL ===> PAGE

         Active CKDS: MYSYS.TST.TESTPLEX.CKDS
         Active PKDS: MYSYS.TST.TESTPLEX.PKDS


 To change the status of a control, enter the appropriate character
 (E - ENABLE, D - DISABLE) and press ENTER.


    FUNCTION                                          STATUS
    --------                                          ------
e   Dynamic CKDS Access                               DISABLED
e   PKA Callable Services                             DISABLED
e   PKDS Read Access                                  DISABLED
e   PKDS Write, Create, and Delete Access             DISABLED
******************************Bottom of data ********************************
```

Figure 8-20 ICSF Administrative Control Functions panel

Key change verification

Now that the SYM-MK and the ASAM-MK Master Keys have been changed on all LPARs, a sample application or multiple applications that use keys protected by the SYM-MK and ASYM-MK must be run.

8.3.8 Exiting disaster recovery

Refer to your disaster recovery test documentation for processes that must be followed before leaving the disaster recovery/business continuity site.

8.4 Business partner tape-sharing example

A common practice is to share tapes with other organizations for joint development, contracting services, or other purposes. To facilitate this sharing, EKM can store two sets of wrapped encryption keys on a TS1120 tape. This function allows another organization to read that specific tape without your providing them any shared secret information or compromising the security of your certificates and keys.

This capability gives you the flexibility to make a specific tape readable by both your own and another organization. To take advantage of this capability, you must add that other organization's certificate and public key to your keystore.

For LTO4 tape drives, the process is different. As with LTO4 encryption, the key is not stored on the media, you have to pass along the symmetric key in addition to the media. You can export symmetric keys from your keystore to a public key encryption-protected file. To share tapes with a business partner, you export the required keys from your keystore to a file. Your business partner has to import the keys from the file to the business partner's keystore to read the LTO4 media encrypted by you. The next section applies to encryption on TS1120 tape drives only.

8.4.1 Key-sharing steps

Key-sharing is performed by adding the public part of the other organization's public-private certificate and keys to your EKM's keystore using a second alias (or key label). When the TS1120 tape is written, the data key is stored on the tape, protected by two sets of public-private keys: your set and the other organization's set. The other organization is then able to use its EKM and private key to unwrap the data key that allows them to read that specific tape. To reiterate, your EKM must have the certificate of the business partner organization. The other organization must have the associated private key in the keystore used by the other organization's EKM.

This capability gives you the flexibility to make a specific tape readable by both your own and another organization. To take advantage of this capability, you must add that other organization's certificate and public key to your keystore.

For information about sharing encrypted LTO4 tapes with business partners, refer to 8.4.3, "Exporting a symmetric key from a JCEKS keystore" on page 302 and 8.4.6, "Importing symmetric keys to a JCEKS keystore" on page 306.

8.4.2 Exporting a public key and certificate to a business partner

To write encrypted tapes to send to a business partner, you must import a public key/certificate from your business partner (the business partner can read the encrypted tape with the business partner's corresponding private key). The reverse is also true: to have a business partner create encrypted tapes that you can read, you must export a public key/certificate from one of your public-private key pairs. Then, you can read the encrypted tape with your private key.

After you have public-private keys and certificates in place, see 7.1, “Keystore and SAF Digital Certificates (keyrings)” on page 228. Examples in the following sections describe how to export a certificate and public key to a business partner and how the business partner imports it so that the business partner can write encrypted tapes that can be read by your EKM.

You must 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 can be imported into the keystore that is being used by the EKM at your business partner’s locations. To send this certificate, *export* it to a data set.

Note: In the following examples, we are sending the public key and corresponding certificate that was created for use by your EKM. We only send the public key (not the private key), so security is not compromised.

Export from JCEKS keystore

You can use the following commands to export the self-signed certificate and public key alias of `MyEKMServer` from an JCEKS keystore to a file called `ExportedPublicKey.cer` using the keytool utility:

```
keytool -export -file ExportedPublicKey.cer -keystore EKMKeystore -alias
MyEKMServer -storepass “somesecretphrase” -storetype JCEKS -provider IBMJCE
-keypass “somesecretphrase”
```

You may print the newly created certificate file by using the keytool printcert utility:

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

Now, you may send the `ExportedPublicKey.cer` file to your business partner. You might possibly have to tell the business partner the alias `MyEKMServer` if the business partner is not able to use an encoding mechanism of Public Key Hash. We explain this in more detail in 8.4.4, “Importing a public key and a certificate from a business partner” on page 303.

Export from JEC4758KS keystore

You may use the following commands to export the self-signed certificate and public key labeled `MyEKMServer` from a JCE4758KS keystore to a file called `ExportedPublicKey.cer` by using hwkeytool:

```
hwkeytool -export -file ExportedPublicKey.cer -keystore EKMKeystore4758 -alias
MyEKMServer -storepass “somesecretphrase” -storetype JCE4758KS -provider
IBMJCE4758 -keypass “somesecretphrase”
```

You can print the newly created certificate file using the hwkeytool printcert utility.

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

Now, you may send the `ExportedPublicKey.cer` file to your business partner. You might possibly need to tell the business partner the alias `MyEKMServer` if the business partner is not able to use an encoding mechanism of Public Key Hash. This is explained in more detail in 8.4.4, “Importing a public key and a certificate from a business partner” on page 303.

Export from z/OS JCERACFKS or JCE4758RACFKS/JCECCARACFKS

You have to select from one of the following techniques depending on the type of certificate you use. To help make this example clearer, we also include samples of the steps that are used to create the public-private key pair from which the public key and certificate are generated. Refer to 7.1, “Keystore and SAF Digital Certificates (keyrings)” on page 228 for more detail about RACF keystores.

Self-signed certificate

For a self-signed certificate:

1. Generate a Rivest-Shamir-Adleman algorithm (RSA) key-pair and certificate with the following RACDCERT command using ICSF for private key storage. The key size is 2048 bits and the private key is saved in the ICSF PKDS in an encrypted form:

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

   Note: If you are not using ICSF, omit the PCICC keyword and change the key size to 1024.

2. To send this certificate, export it to a data set:

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

3. Ensure that the EKM server certificate is connected to the EKM's keyring. This example shows connecting the certificate that identifies the EKM server to the EKM keyring. You have to modify these command examples to suit your installation's needs.

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

4. If you use ICSF, ensure that the EKM server instance has RACF authority to the key label of the private key that is stored in the ICSF PKDS. Also, be sure to refresh the *in-storage* copies of the CSFKEYS class profiles using the commands shown in Example 8-24.

   Example 8-24 Refresh 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
   ```

5. Send the `ExportedPublicKey.cer` file to your business partner. You might possibly need to tell the business partner the alias `MyEKMServer` if the business partner is not able to use an encoding mechanism of Public Key Hash, explained in 8.4.4, "Importing a public key and a certificate from a business partner" on page 303.

Certificate signed by an internal certificate authority

To generate a certificate signed by an internal certificate authority:

1. Generate a self-signed certificate authority certificate using the following command:

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

   Note: If you are not using ICSF, omit the PCICC keyword and change the key size to 1024.

2. The following RACDCERT command uses ICSF for private key storage, and it is signed with the local certificate authority certificate generated in the previous step:

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

   Note: If you are not using ICSF, omit the PCICC keyword and change the key size to 1024.

3. With your business partners or remote sites that you trust, you must validate a common certificate authority (CA), whether third-party or self-signed, depending on your business and security practices. To send this certificate, export it to a data set:

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

4. Ensure that the EKM server certificate is connected to the EKM's keyring. Example 8-25 shows connecting the certificate that identifies the EKM server to the EKM keyring. You have to modify these command examples to suit your installation's needs.

Example 8-25 Connecting the EKM server certificate the EKM keyring

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

5. If you use ICSF, ensure that the EKM 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 as shown in Example 8-24 on page 300.
6. Send the `ExportedPublicKey.cer` file to your business partner, and you might have to tell the business partner the alias `MyEKMServer` if the business partner is not able to use an encoding mechanism of Public Key Hash, which is explained in 8.4.4, "Importing a public key and a certificate from a business partner" on page 303.

Certificate signed by a third-party certificate authority

To generate a certificate signed by a third-party certificate authority:

1. Generate an RSA key pair and certificate using the following RACDCERT command using ICSF for private key storage:

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

> **Note:** If you are not using ICSF, omit the PCICC keyword and change the key size to 1024.

2. Generate and save a certificate request to a data set (hlq.PUBKEY.REQUEST.ITOPS) using the following command:

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

3. Submit a certificate request, `hlq.PUBKEY.S2048.ITOPS`, to your certificate provider. The response that you receive is an X.509 certificate. This example assumes that the certificate that you receive from your third-party certificate authority is saved in the data set `'hlq.THIRD.PARTY.CERT'` on z/OS. The contents of this data set are imported into RACF.

> **Note:** This data set contains only the signed certificate that identifies the EKM instance running on z/OS and perhaps the certificate authority certificate. The private key for the EKM certificate remains protected in the PKDS by either ICSF or RACF.

4. Receive the response into data set `'hlq.THIRD.PARTY.CERT'` and add the certificate to RACF using the following command:

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

If the CA certificate is not contained in the data set `'hlq.THIRD.PARTY.CERT'`, you will need to acquire the CA certificate that signed the EKM certificate from the External Certificate Authority and add it to RACF as a CERTAUTH.

5. With your business partners or remote sites that you trust, you must validate a common certificate authority (CA), whether third-party or self-signed, depending on your business and security practices. To send this certificate, export it to a data set:

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

6. Ensure that the EKM server certificate is connected to the EKM's keyring. This example shows connecting the certificate that identifies the EKM server to the EKM keyring.

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

   You need to modify these commands to suit your installation's needs.

7. If you use ICSF, ensure that the EKM 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 as shown in Example 8-24 on page 300.

8. Send the `ExportedPublicKey.cer` file to your business partner, and you might need to tell the business partner the alias `MyEKMServer` if the business partner is not able to use an encoding mechanism of Public Key Hash, which is explained in more detail in the following section.

8.4.3 Exporting a symmetric key from a JCEKS keystore

If you want to send encrypted LTO4 media to a business partner, you will also have to send the symmetric AES data keys that were used to encrypt the media. You may export a symmetric key or a range of symmetric keys to a file using the **`keytool -exportseckey`** command.

The keytool utility protects the symmetric keys using public key encryption. For your business partner to be able to import the symmetric keys into the business partner's keystore, you must use one of the business partner's public keys for encrypting the symmetric keys. Before exporting the private keys, you have to make sure that your keystore contains at least one of your business partner's public keys. If this is not the case, request a public key/certificate from your business partner and import it into your keystore as described in "Import to a JCEKS keystore" on page 303. When the business partner receives the file with your exported keys, the business partner can import them with the corresponding private key.

The reverse is also true. To have a business partner export symmetric keys that you can import into your keystore, you must export a public key/certificate from one of your public-private key pairs and send it to your business partner. After importing your certificate, the business partner can export symmetric keys using your public key, and you may import the encrypted symmetric keys into your keystore using the corresponding private key.

In Example 8-26, we export the symmetric key with the alias `ekm00000000000000001` from the keystore `EKMKeys.jceks` to a file named `exportsym.key by` using the public key contained in the certificate with the alias `partnercert.`

Example 8-26 Exporting a symmetric key

```
keytool -exportseckey -alias ekm000000000000000001 -keyalias partnercert
-keystore EKMKeys.jceks -storepass passw0rd -storetype jceks
-exportfile exportsym.key
```

8.4.4 Importing a public key and a certificate from a business partner

We now look at examples of how to import the certificate and public key contained in `ExportedPublicKey.cer` from a business partner to the EKM keystore with an alias CompanyXPublicKey to various keystore types. In this example, the imported alias of CompanyXPublicKey does not match the original business partner's alias of MyEKMServer. 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, your business partner must provide the alias on the import command.

Note: The alias that you specify on the import must match the alias that was used by the business partner (`MyEKMServer` in the examples given in 8.4.2, "Exporting a public key and certificate to a business partner" on page 298) 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 uses a Hash value rather than the KeyLabel to identify the key. Although Hash gives slightly less performance, it allows you to import a certificate/public key from a business partner without knowing the alias/KeyLabel that the business partner used to create and export the key. In addition, it gives you the freedom to specify the label that you want to use to identify your business partner's public key. Therefore, using Public Key Hash is the preferred method.

Import to a JCEKS keystore

You can use the following commands to import the certificate and public key contained in `ExportedPublicKey.cer` from a business partner to an JECKS keystore with an alias CompanyXPublicKey from various keystore types using the keytool utility:

```
keytool -import -file ExportedPublicKey.cer -keystore EKMKeystore -alias
CompanyXPublicKey -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"
```

To list the contents of the keystore to which the certificate was imported, use the **`keytool -list`** command:

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

Import to a JCE4758KS keystore

You may use the following commands to import the certificate and public key from a business partner contained in `ExportedPublicKey.cer` to an JCE4758KS keystore with an alias CompanyXPublicKey from various keystore types using the hwkeytool utility:

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

You may list the contents of the keystore to which the certificate was imported by using the hwkeytool list utility:

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

Import from z/OS JCERACFKS or JCE4758RACFKS/JCECCARACFKS

To import into a z/OS RACF keystore, use RACDCERT to add the certificate to RACF. The public key in the certificate can also be saved in the ICSF PKDS depending on the operands supplied to the RACDCERT command.

The following RADCERT ADD command imports the certificate and public key from the business partner contained in ‘dataset _containing_the_cert_received’ with an alias CompanyXEKMServer. RACDCERT ID(EKMServ):

```
ADD('dataset_containing_the_cert_received') TRUST WITHLABEL('CompanyXEKMServer')
PCICC(companyX.EKMServ.cert) SIZE(2048)
```

Note that in this example the imported alias of CompanyXEKMServer does not match the original business partner's alias of MyEKMServer. 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 has to be provided to you by your business partner.

Note: If you are not using ICSF, omit the PCICC keyword and change the key size to 1024.

The WITHLABEL keyword associates a string or friendly name for the certificate that is being imported, and this name is used by EKM when accessing the certificate. Refer to the *z/OS Security Server RACF Command Language Reference,* GA22-7800, for details about the RACDCERT command.

You also have to ensure that this certificate is connected (or associated) to the EKM server's keyring, which is accomplished using the RACDCERT command as shown in Example 8-27. This example assumes that the EKM keyring on this z/OS system is EKMRing, and that the z/OS user ID associated with the EKM process is EKMSERV.

Example 8-27 RACDCERT command to connect the certificate with the EKM server's keyring

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

Note: Because this certificate contains only a public key, using the USAGE(CERTAUTH) option is extremely important. If it is not specified, EKM does not start, because it believes that the certificate that was added must also contain a private key.

Ensure that the EKM server is authorized to read from its keyring and that the EKM server is authorized to use the ICSF key label. Ensure that the required RACF FACILITY class profiles are defined. If not, issue the RDEFINE commands as shown in Example 8-28 to define these profiles, which protect the use of keyring functions.

Example 8-28 RACF authorizes the ICSF key labels

```
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 EKM 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 using the commands shown in Example 8-29 on page 305.

Example 8-29 Ensure EKM server authority to the ICSF PKDS key labels

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

8.4.5 Tape exchange and verification

Refer to Example 8-30 on page 306.To validate tape exchange with your business partner:

- At your local site, perform the following steps:
  a. Create a key pair and export the public key to your business partner by using techniques discussed in 8.4.2, “Exporting a public key and certificate to a business partner” on page 298 as appropriate for your keystore type.

     Note: This solution does not require that you and your business partner have the same keystore type. Sharing public keys between different types of keystores is possible, if you both use TS1120 drives.

  b. Perform Read/Write to the tape. If your business partner used the Hash encoding mechanism to encode your public key key-label on the tape, you can process it on any system where the EKM has access to the public-private key pair for this label. Otherwise, ensure that the business partner used the same alias that you did as described in the note reference in the next step. If you are running z/OS, you can use, for example, IEBDG DITTO, or IEBGENER.
- Your business partner steps are:
  a. Load the public key certificate to the system by importing it to your keystore using the techniques discussed in the previous section, “Importing a public key and a certificate from a business partner” on page 303, as appropriate for your keystore type.

     Note: The alias you specify on import must match the alias that was used by the business partner (`MyEKMServer` in the examples given in 8.4.2, “Exporting a public key and certificate to a business partner” on page 298) if you plan to specify an encoding mechanism of label “L” when encrypting tapes. Optionally, you may specify an encoding mechanism of Public Key Hash “H” that uses a Hash value rather than the KeyLabel to identify the key. Although Hash gives slightly less performance, it allows you to import a certificate/public key from a business partner without knowing the alias/KeyLabel that the business partner used to create and export the key. In addition, it gives you the freedom to specify the label you want to use to identify your business partner’s public key. Therefore, using Public Key Hash is the preferred method.

  b. Encrypt a tape to share with your business partner using two key labels. One of the labels should be the public key that you imported in the earlier step from your business partner. As just noted, if you do not use Hash to encode this key, you also have to let your partner know the alias of this key. The other label you use must be an alias of a public-private key pair that allows you to read the tape. EKM processing has a safeguard built-in that requires that you must be able to read any tape that you encrypt. If you attempt to write a tape with an alias that points to only a public key, the processing fails. Because you do not have the private key associated with that business partner key, you cannot read the tape yourself and you must provide a second key where you have access to the private key in order for this processing to complete successfully.

Example 8-30 Sample job to create a business partner-encrypted tape

```
//CO2STRW1   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.CO2M5CX2.PC5.NOPOOL.CO2STRS1.MASTER,
//         KEYLABL1='MY_PUBLIC_PRIVATE_KEY',
//         KEYENCD1=L,
//         KEYLABL2='COMPANYXPUBLICKEY',
//         KEYENCD2=H,
//         LABEL=(1,SL),UNIT=CO2M5CX2,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
/*
```

Note: The z/OS Compatibility configuration property should be set to true for the EKM you are using if you plan to exchange tapes between z/OS and non-z/OS systems.

c. Send the tape to the business partner site that provided the public key for processing.

8.4.6 Importing symmetric keys to a JCEKS keystore

If a business partner wants to send you encrypted LTO4 media, you will need the symmetric AES data keys that were used to encrypt the media in order to decrypt them. The business partner can provide you with a file containing one or more symmetric keys. Import these keys from the file into your keystore by using the **keytool -importseckey** command.

The symmetric keys in export files are protected by public key encryption. For you to be able to import the symmetric keys to your keystore, your business partner must use one of your public keys for encrypting the symmetric keys. If your business partner does not have one of your public keys in the business partner's keystore, the business partner must request a public key/certificate from you and import it to the business partner's keystore as described in "Import to a JCEKS keystore" on page 303. When you receive the file with the exported symmetric keys of your business partner, you can import them using the corresponding private key for decryption.

The reverse is also true. In order to enable a business partner to import symmetric keys that you exported from your keystore to a file, you have to use one of your business partner's public keys for export.

In Example 8-31, we import one or more symmetric keys from a file named `importsym.key` to the keystore `EKMKeys.jceks` using a private key with the alias `mycert`.

Example 8-31 Importing a symmetric key

```
keytool -importseckey -keyalias mycert -keypass passw0rd -keystore EKMKeys.jceks
-storepass passw0rd -storetype jceks -importfile importsym.key
```

8.5 RACF export tool for z/OS

There are many ways to transport certificates within a company to ensure that all tapes are encrypted with the same keys. Consider using the Java keytool as a means of obtaining an X509V3 certificate for use by the Encryption Key Manager for z/OS deployment. The Java keytool provides a means to cryptographically clone the certificate and private key through the distribution of the PKCS12 (.p12) file to other z/OS instances. The PKCS12 file is password-protected and contains the certificate, public and private key, and a CA certificate that signed it.

In the PKCS12 password-protected file, the private key is stored with compliance to PKCS#8; the password encryption is compliant with PKCS#5.

For more information about an exported PKCS12, refer to:

`http://www.rsasecurity.com/rsalabs/node.asp?id=2124`

This form of certificate exchange, where even the private key is stored in a file, can raise a security concern that an individual knows the password used to encrypt the contents of the PKCS12 file; therefore, the individual knows the private key that can then be used to decrypt all tapes encrypted with that private key's matching public key. In this situation, the security of the business is based on the trust of the individual exporting and transporting PKCS12 files.

There might be a concern from an audit perspective; the accessibility of the private key in this situation is only as secure as the person who knows the password used to protect the private key.

A tool to satisfy these concerns is the KEYXFER tool, which:

- Generates the private key using z/OS ICSF and hardware cryptography.
- Stores the private key encrypted under the host Master Key in ICSF's PKDS.
- Reads the key record from ICSF (still remains encrypted under host Master Key) and distributes to other z/OS systems.
- Imports into remote z/OS systems in an encrypted form; import processing as suggested can be done only if the same Master Key is used across the systems.
- Associates the EKM certificate with the encrypted private keystore in z/OS ICSF PKDS.

Note: This method for exporting and transporting certificates is only valid on z/OS, and this method is only valid if you are using a hardware-based RACF keystore on z/OS.

In this scenario, no passwords or similar means to externally gain knowledge of the private key is ever available. The private key is moved from System A to System B in this scenario in an encrypted form: it is enciphered by the host Master Key. Therefore, using a password-based encryption scheme to protect the private key is not necessary. This scenario depends on utilizing the same host Master Key on System A and System B.

Note: To successfully export an encrypted ICSF PKDS record entry and import into another PKDS instance using the KEYXFER utility, the same ICSF hardware Master Key must be used between the exporting and importing z/OS images.

8.6 Audit log considerations

The EKM can create audit records, which wrap the log to three files. After the last file is full, the first file is rewritten.

On z/OS, you need to allocate file system space for the EKM audit logs, and if requested by IBM Service, you need to enable the EKM debug log.

You may choose to allocate a file system specifically for use by the EKM for audit and debug file storage. Assume 500 cylinders of space allocated to the EKM's audit and debug log file system until you have observed based on tape and EKM activity how quickly the audit logs wrap. The file system must not be shared by the EKM instances running in a Sysplex environment, but must be private to each EKM instance. This prevents any possible interleaving of audit or debug logs between EKM instances.

Create the EKM configuration file in `/&SYSNAME/etc` and customize it accordingly for your installation, as follows:

- Audit.handler.file.directory

  Modify this to a location where you want EKM to store the audit logs.

- z/OS Compatibility

  Use this option if you intend to exchange tapes between z/OS and non-z/OS systems.

  If the EKM configuration file parameter `requireHardwareProtectionForSymmetricKeys` is set to `true`, this value must be set to `true` also. This applies to *all* supported platforms.

8.6.1 Audit overview

The audit subsystem writes textual audit records to a set of sequential files as various auditable events occur during EKM'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 and renamed based on the current time stamp. 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, and their names are sequenced by the time stamp 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 file system, a script or program must be created which monitors the set of files in the configured audit directory/folder/container. As files are closed and named based on the time stamp, the file's contents must be copied and appended to the chosen long-term, continuous log location and then cleared. Be careful not to remove or alter the file that is having records written to it by EKM while EKM is running (this file does not have a time stamp in the file name).

8.6.2 Audit log parsing tool

In Example 8-32, we introduce a Java application that takes an audit log file as input. The program then parses the audit log for tape keylabel information and outputs it to the panel. If this output is redirected to a file, it can then be read into a spreadsheet as a comma-delimited format, and then you can work with it. This parsing tool is included here as a sample way that you can keep track of tapes and the keylabels on the tapes.

To run this program, you must first compile it:

```
javac AuditCrawler.java
```

The **javac** command has to be in the $PATH; the **javac** command is located in $JAVA_HOME/bin. After the source is compiled, an AuditCrawler.class file is created. To run the program, execute the command:

```
java AuditCrawler auditlog
```

Examples in this section include:

- Example 8-32 shows the application that takes an audit log file as input.
- Example 8-33 on page 313 shows the format of the audit log records.
- Example 8-34 on page 313 shows the output of the AuditCrawler program.

Example 8-32 AuditCrawler source tool

```
   import java.io.BufferedReader;
   import java.io.File;
   import java.io.FileNotFoundException;
   import java.io.FileReader;
   import java.io.IOException;
   import java.util.ArrayList;
   import java.util.Iterator;
   import java.util.List;

/**
 * @author svenjamin
 * @revision johann
 *
 */
public class AuditCrawler {

   private static final String DRIVE_SER_NUM = "resource=[name= Drive Serial Number:";
   private static final String WORLD_WIDE_NUM = "WWN:";
   private static final String VOL_SER = "VolSer:";
   private static final String KEY_ALIAS_LABEL = "Key Alias/Label[";
   private static final String TYPE_FINAL = ";type=file]";
   private static final int DRIVE_SER_NUM_LEN = 12;
   private static final int WORLD_WIDE_NUM_LEN = 16;
   private static final int VOL_SER_LEN = 6;

   public static void main(String[] args) {
      AuditCrawler m = new AuditCrawler();

      File filename = m.validateInputFile(args[0]);
      m.readInputFile(filename);

   }

   private void readInputFile(File filename) {
      List driveList = new ArrayList();
```

```
        List timestampList = new ArrayList();
        String times = null;
        try {

           BufferedReader br = new BufferedReader(new FileReader(filename));

           while (br.ready()) {
              String s = br.readLine();
              // Parse the timestamp and save it until we find
              // an audit record with drive/labels
              if (s.indexOf("timestamp") > 0) {
                 times = s;
              }
              // parse the entry with keylabel info
              // add our timestamp to a list
              // add the drive info to a list
              if (s.indexOf("resource=[name= Drive Serial Number") > 0) {
                 timestampList.add(times);
                 driveList.add(s);
              }

           }
        } catch (FileNotFoundException e) {
           e.printStackTrace();
        } catch (IOException e) {
           e.printStackTrace();
        }
        System.out

.println("DriveSerialNumber,WorldWideNodeName,VolSer,Alias1,Alias2,Day,Mon,Date,time,Timezo
ne,Year");
        if (!driveList.isEmpty()) {
           Iterator it = driveList.iterator();
           int i = 0;
           while (it.hasNext()) {
              String s = (String) it.next();
              try {
                 //process our strings and print them
                 //add some space to line things up pretty
                 System.out.println("     "+processFoundLine(s)
                       + processTimeStamp((String) timestampList.get(i)));
              } catch (Exception e) {

              }
              i++;
           }

        }
     }

     // Check to see if audit file exists and we have read permission
     private File validateInputFile(String filename) {
        File f = new File(filename);
        if (!f.exists()) {
           String s = "File " + filename + " does not exist.";
           System.err.println(s + " Bailing!");
           throw new IllegalArgumentException(s);
        }

        if (!f.canRead()) {
```

```
            String s = "Can't read the file " + filename;
            System.err.println(s + " Bailing!");
            throw new IllegalArgumentException(s);
        }
        return f;
    }
//comma delimit the timestamp

    private String processTimeStamp(String s) throws IllegalArgumentException {
        String timeInfo = s.substring(12);
        String[] splitString = timeInfo.split(" ");
        StringBuffer output = new StringBuffer();
        for (int i = 0; i < splitString.length; i++) {
            output.append(",");
            output.append(splitString[i]);

        }
        return output.toString();

    }

    private String processFoundLine(String s) throws IllegalArgumentException {
        StringBuffer sb;// = new StringBuffer();
        String str_driveSerialNumber = null;
        int int_driveSerialNumberIndex = s.indexOf(DRIVE_SER_NUM);

        String str_worldWideNumber = null;
        int int_worldWideNumberIndex = s.indexOf(WORLD_WIDE_NUM);

        String str_volSer = null;
        int int_volSerIndex = s.indexOf(VOL_SER);
        int int_keyLabelIndex = s.indexOf(KEY_ALIAS_LABEL);
        int int_finalTypeIndex = s.indexOf(TYPE_FINAL);

        // do the processing in reverse order to save cycles
        // process the key labels
        if ((int_keyLabelIndex > 0) && (int_finalTypeIndex > int_keyLabelIndex)) {
            // now we can allocate the StringBuffer...
            sb = new StringBuffer();

            // get a new string for the key label stuff...
            String keylabels = s.substring(int_keyLabelIndex,
                  int_finalTypeIndex + 1);
            String firstCert = keylabels.substring(keylabels.indexOf(":") + 2,
                  keylabels.lastIndexOf(KEY_ALIAS_LABEL)).trim();
            sb.append(firstCert);
            sb.append(",");
            String lastCert = keylabels
                  .substring(keylabels.lastIndexOf(":") + 2);
            sb.append(lastCert);

        } else {
            throw new IllegalArgumentException("No Key Labels to process");
        }

        // Next we process the VOLSER and prepend it to our StringBuffer
        if ((int_volSerIndex > 0) && (int_keyLabelIndex > int_volSerIndex)) {
            str_volSer = s.substring(int_volSerIndex + VOL_SER.length() + 1,
                  int_keyLabelIndex - 1);
```

```
    if (str_volSer.length() != VOL_SER_LEN) {
        System.out.println("VolSer " + str_volSer
                + " is not appropriate!");
        throw new IllegalArgumentException("VolSer "
                + ((str_volSer == null) ? "UNKNOWN" : str_volSer)
                + " is not the right length");

    }
    sb.insert(0, str_volSer + ",");

} else {
    throw new IllegalArgumentException("Problem with VolSer processing");
}

// Next process the world wide number
if ((int_worldWideNumberIndex > 0)
        && (int_volSerIndex > int_worldWideNumberIndex)) {
    str_worldWideNumber = s.substring(int_worldWideNumberIndex
            + WORLD_WIDE_NUM.length() + 1, int_volSerIndex - 1);
    if (str_worldWideNumber.length() != WORLD_WIDE_NUM_LEN) {
        System.out.println("WWN " + str_worldWideNumber
                + "is not appropriate!");
        throw new IllegalArgumentException("WWN "
                + ((str_worldWideNumber == null) ? "UNKNOWN"
                        : str_worldWideNumber)
                + "is not the right length");

    }
    sb.insert(0, str_worldWideNumber + ",");

} else {
    throw new IllegalArgumentException("Problem with WWN processing");
}

// Process the drive serial number
if ((int_driveSerialNumberIndex > 0)
        && (int_worldWideNumberIndex > int_driveSerialNumberIndex)) {

    str_driveSerialNumber = s.substring(int_driveSerialNumberIndex
            + DRIVE_SER_NUM.length() + 1, int_worldWideNumberIndex - 1);

    if (str_driveSerialNumber.length() != DRIVE_SER_NUM_LEN) {
        System.out.println("Drive Serial Number "
                + str_driveSerialNumber + "is not appropriate!");
        throw new IllegalArgumentException("Drive Serial Number "
                + ((str_driveSerialNumber == null) ? "UNKNOWN"
                        : str_driveSerialNumber)
                + "is not the right length");
    }
    sb.insert(0, str_driveSerialNumber + ",");

} else {
    throw new IllegalArgumentException(
            "Problem with Drive Serial Number processing");
}
//remove the last comma
//processTimeStamp will prepend to its
//date information
sb.deleteCharAt(sb.length() - 1);
return sb.toString();
```

```
   }
}
```

In Example 8-33, we see the format of the audit log records. This is what is read by the AuditCrawler program and then subsequently parsed into an organized comma-delimited list shown in Example 8-34.

Example 8-33 Sample audit log

```
Runtime event:[
  timestamp=Thu Sep 28 09:05:19 EDT 2006
  event source=com.ibm.keymanager.c.eb
  outcome=[result=successful]
  event type=SECURITY_RUNTIME
  resource=[name=Process Message;type=file]
  action=stop
  ]
Runtime event:[

  event source=com.ibm.keymanager.c.eb
  outcome=[result=successful]
  event type=SECURITY_RUNTIME
  resource=[name=Process Message;type=file]
  action=stop
  ]
Runtime event:[
  timestamp=Thu Sep 28 09:05:19 EDT 2006
  event source=com.ibm.keymanager.c.fb
  outcome=[result=successful]
  event type=SECURITY_RUNTIME
  resource=[name= Drive Serial Number: YN1B00001388 WWN: 5005076300020216 VolSer:
451AAF Key Alias/Label[0]: cert1 Key Alias/Label[1]: cert2;type=file]
  action=stop
  ]
```

In Example 8-34, we see the output from the AuditCrawler program. The first line is an explanation of the fields. The subsequent lines are the parsed and formatted records. Only records that have the information containing drive key label information are printed here.

Example 8-34 Sample output

```
DriveSerialNumber,WorldWideNodeName,VolSer,Alias1,Alias2,Day,Mon,Date,time,Timezone,Year
     YN1B00001436,5005076300020215,444AAF,cert1,cert2,Tue,Aug,22,11:23:30,EDT,2006
     YN1B00001436,5005076300020215,444AAF,cert1,cert2,Tue,Aug,22,13:11:11,EDT,2006
     YN1B00001388,5005076300020216,444AAF,cert1,cert2,Tue,Aug,22,14:37:09,EDT,2006
```

Note: By following the JZOS example in “EKM and JZOS” on page 567, you can set this up to run as a job that is scheduled to run when a new audit log has been created.

Part 3

Implementing and operating the TKLM

In this part, we provide the information required to plan for, implement, and operate the Tivoli Key Lifecycle Manager (TKLM). We also describe planning considerations for keystores and key management.

9

Planning for TKLM and its keystores

This chapter, we discuss the planning or the Tivoli Key Lifecycle Manager (TKLM). TKLM or Encryption Key Manager (EKM) must be implemented before you can encrypt any tape using System-Managed Encryption (SME) or Library-Managed Encryption (LME). Application-Managed Encryption (AME) does not require a TKLM or EKM implementation. This planning can occur even before your hardware arrives.

Chapter 10, “Implementing TKLM” on page 325 completely describes the implementation of TKLM.

TKLM provides life cycle management for keys and certificates of an enterprise. It may be used as a key store for encryption capable IBM storage such as the TS1120, TS1130, and IBM LTO4 tape drive.

You must decide on what platform (or platforms) the TKLM will run. We suggest that you implement TKLM only on a single operating system type to allow TKLM synchronization between primary and secondary TKLM instances.

In this chapter, we first provide a TKLM planning quick reference, then we describe the requirements for each of the platforms where TKLM can run. Then we discuss various TKLM and keystore implementation considerations.

9.1 TKLM planning quick-reference

Table 9-1 compares the encryption characteristics of the TS1120 drive to the TS1130 and LTO4 drive. Table 9-1 compares the Tivoli Key Lifecycle Manager (TKLM) software prerequisites for each platform. On Open Systems platforms, the TS1120, TS1130 and the LTO4 are almost identical as to which encryption methods they support and the operating system software requirements. These requirements were extracted from *Tivoli Key Lifecycle Manager Installation and Configuration Guide* SC23-9977.

Table 9-1 Encryption implementation characteristics comparison

Description	TS1120 tape drive	TS1130 tape drive	LTO4 tape drive
Encryption standard	AES (256-bit)	AES (256-bit)	AES (256-bit)
Encryption process for data	Symmetric AES (256)	Symmetric AES (256)	Symmetric AES (256)
Encryption key model	Wrapped key	Wrapped key	Direct key
Encryption type for data keys	Public-private key (Asymmetric)	Public-private key (Asymmetric)	None
Data keys used	Unique data key for each cartridge	Unique data key for each cartridge	Keylist: A list or range of data keys used, pregenerated in keystore
Data keys stored?	Wrapped (that is, encrypted) data keys (2) stored on cartridge, called EEDKs	Wrapped (that is, encrypted) data keys (2) stored on cartridge, called EEDKs	Stored in keystore
Keystore	Contains private keys and certificates representing public keys. Preloaded in keystore	Contains private keys and certificates representing public keys. Preloaded in keystore	Contains data keys that have been pregenerated
Keystore types supported by TKLM	JCEKS (All platforms)	JCEKS (All platforms)	JCEKS (All platforms)

Table 9-2 lists the current platforms on which TKLM may be installed.

Table 9-2 TKLM Software Platforms

Description	Patch, maintenance level at time of initial publication
AIX Version 5.3 64-bit, and Version 6.1	For Version 5.3, use Technology Level 5300-04 and Service Pack 5300-0402
Sun Server Solaris 10 (SPARC 64-bit) Note: TKLM runs in a 32-bit JVM.	None
Windows Server 2003 R2 (32-bit Intel)	None
Red Hat Enterprise Linux AS Version 4.0 on x86 32-bit	compat-libstdc++-33-3.2.3-47.3 package. run rpm -qa | grep -i "libstdc" to verify it is present.
SUSE Linux Enterprise Server Version 9 and 10 on x86 (32-bit)	compat-libstdc++-33-3.2.3-47.3 package. run rpm -qa | grep -i "libstdc" to verify it is present.

Table 9-3 lists the hardware configuration for running TKLM.

Table 9-3 TKLM Hardware Re om mended Requirements

System components	Recommended Value
System memory (RAM)	4 GB
Processor speed	Linux/Windows 3.0 GHz dual processor AIX and Sun Solaris: 1.5 GHz (4-way)
Free disk space	20 GB
Disk space free in /tmp or C:\temp	500 MB
Disk space free in /opt	3.5 GB
Disk space free in /home directory for DB2 Database	1.5 GB

Access requirements

To install TKLM you must have different access levels by platform. Windows requires a user with Administrator access. Linux, Solaris and AIX require root access.

Browser support

The TKLM server is accessed using a Web browser, Firefox 2.0.0.14, or Internet Explorer® 6.0.x SP1, or Internet Explorer 7.0. Note that AIX has no supported browser; the TKLM interface must be accessed across the network using one of the supported browsers. With TKLM 1.0, Firefox 3.0.3 did not render all pages correctly, although Firefox 2.0.0.17 did. Therefore, use the supported browser's major version numbers.

9.2 TKLM and keystore considerations

We begin with questions for you to consider:

- On what platforms will you run TKLMs?

  The Tivoli Key Lifecycle Manager is currently supported on:

  – AIX 5.3 Technology Level 5300-04 and Service Pack 5300-04-02
  – AIX 6.1
  – Sun Server Solaris 10 (SPARC 64 bit)
  – Windows Server 2003 R2 (32 bit)
  – Red Hat Enterprise Linux AS Version 4.0 on x86 32–bit
  – SUSE Linux Enterprise Server Versions 9 and 10 on x86 (32–bit)

  You might want to run the TKLM on more than one of these platforms. We do not recommend running TKLM on different OS Platforms. With the TKLM 1.0 release as the backup and restore mechanism used to create redundant TKLM's requires the same platform and configuration. In all cases, you want TKLM to be running on a highly available platform that will be available anytime you need access to the drives. If you have a disaster recovery site, you should also have a TKLM at this site or a way to rapidly deploy one with the current keystore and configuration.

- What keystore deployment model will you employ? At the time of this writing JCEKS (file-based) is the only supported key store. Refer to Table 9-4.

Table 9-4 Comparison of supported keystores

Keystore type	Platforms supported	TS1120, TS1130 (stores keypairs and certificates)	LTO4 (stores symmetric keys)	TS1120, TS1130, LTO4	Symmetric key tools available
JCEKS	ALL	Yes	Yes	Yes	keytool

- Do you want to use secure hardware cryptographic services?

  This consideration is driven by the regulations and requirements your business is required to meet. If you require a hardware solution at this time, use EKM.

- Is your TKLM behind a firewall?

  As part of your centralized key management strategy, the TKLM that your platform has to access might be behind a firewall. If so you should work with your company firewall to determine appropriate rules. TCP/IP port 3801 is the default TKLM port. When using SSL TCP port 441, note that this is different from tape libraries that usually default to port 443.

9.2.1 TKLM configuration planning checklist

Make sure you:

- Have selected a support OS and hardware platform. Note that the machine name cannot start with the following text:

  – IBM
  – SQL

- If you are migrating an existing EKM server, verify that the server meets the TKLM server recommendations:

  – Back up your EKM configuration prior to migration.
  – Write down the path to the EKM; this path name should not contain any spaces.

- Schedule an outage for your EKM server. Note that if you have redundant EKMs, you do not have to stop all EKMs, but you do have to stop the EKM that is being migrated to TKLM. After you have the first EKM migrated to TKLM, we suggest using the TKLM backup/restore function to configure the remaining EKMs.
- Migration types that are not supported:
  - Administrator SSL keystores or truststores
  - PKCS11Impl keystores and truststores

- If this is a new TKLM installation:
  - Know the recipients for the tapes to be written and read. For each recipient:
    - An associated X.509 certificate must exist when using TS1120 or TS1130
    - A key or range of keys must be pregenerated within the keystore when using LTO4
  - Identify the tape drives that will be used.

    For each tape drive, determine the drive serial number for input into the TKLM drive table. You may also configure TKLM to accept unknown drives.
  - Have existing keys and certificates that you plan to use.

Note: For TKLM servers that typically handle requests from the same set of tape drives, the information in the associated keystores *must* be kept the same.

This consistency is required so that no matter which TKLM server is contacted, the necessary information is available for the TKLM server to use to support the requests that the TKLM server receives from the tape drives.

9.2.2 Best security practices for working with keys and certificates

Best practices are:

- *Do not* lose your keys and certificates!
- No really, *do not* lose your keys and certificates.
- Do not leave your keys and certificates for other users to see.

  For example putting them on another user's general purpose mobile computer or USB key is a bad idea.
- Make sure you *back up your keys and certificates* and *verify you can retrieve them from the backup.*

Important: Although IBM has services that can help you to recover data from a damaged tape, if the damaged tape is encrypted, what you receive from the recovery will still be encrypted data. So, if you lose your keys, you have lost your data.

9.2.3 Acting on the advice

Maintenance, backup, and restoration of key and certificate information can be accomplished using the TKLM GUI interface or CLI **`wsadmin`**. See 11.4, "TKLM backup and restore procedures" on page 361 for more information about how to backup TKLM.

9.2.4 Multiple TKLMs for redundancy

To run a replica server, at this time, TLKM requires that both systems be running the same OS, middleware, file system layout, and others. TKLM is not cluster-aware and not enabled for automatic failover. Manual failover is possible. Synchronization is achieved by backing up one server and restoring the backup configuration on the other server. You should plan to do this backup and restore process when the following events take place:

- Initial configuration
- Adding keys or devices
- Key or certificate replacement intervals
- Certificate Authority (CA) requests
- Upgrades to TKLM or middleware, such as DB2

9.3 Other TKLM considerations

In this section, we summarize additional TKLM considerations.

9.3.1 Database selection

TKLM includes DB2 in the installation process:

- If you already have IBM DB2 9, TKLM uses your existing DB2 installation.
- If DB2 9 is less than DB2 9.1 fix pack 4, TKLM installer upgrades your DB2 installation.
- If you have a prior version of DB2 or do not have DB2 on the target machine, TKLM installs DB2 9.1 fix pack 4.

The TKLM backup/restore function backs up the TKLM relevant tables.

9.3.2 EKM to TKLM migration

TKLM supports migration from an EKM installation to a TKLM installation, which requires stopping the EKM server. You can either schedule a maintenance window to shut down the EKM or, if you have redundant EKMs, you can stage this migration. The TKLM Installation and planning guide has an excellent discussion on EKM migration. It can be found in either:

- *Tivoli Key Lifecycle Manager Installation and Configuration Guide* SC23-9977
- http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tklm.doc/install/cpt/cpt_ic_plan_migration.html

The following list is a quick review of what to be aware of when migrating from EKM to TKLM:

- Back up your EKM configuration prior to migration.
- Write down the path to the EKM; this path name should not contain any spaces.
- Schedule an outage for your EKM server. Note that if you have redundant EKMs, you do not have to stop all EKMs, but you do have to stop the EKM that is being migrated to TKLM. After you have the first EKM migrated to TKLM, we suggest using the TKLM backup and restore function to configure the remaining EKMs.
- Migration types that are not supported:
  - Administrator SSL keystores or truststores
  - PKCS11Impl keystores and truststores

9.3.3 Data exchange with business partners or different platforms

TKLM 1.0 does not currently support key group exchange between EKM and TKLM. If you are exchanging LTO4 cartridges with a business partner running EKM, you have to maintain an EKM instance or coordinate a migration to TKLM.

9.3.4 Disaster recovery considerations

Your disaster recovery site must have an operational Key Manager that has recently been synchronized with the primary site. Or, at a minimum a recent backup of the Key Manager based on the criteria used in Section along with your TKLM configuration worksheets and install media. The TKLM worksheets can be found in either:

- *Tivoli Key Lifecycle Manager Installation and Configuration Guide* SC23-9977t
- http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tklm.doc/welcome.htm

10

Implementing TKLM

This chapter is intended as an example to show how the TKLM is installed on a Red Hat Enterprise Advanced Server 2 system. TKLM is supported on other systems, described in Chapter 9, “Planning for TKLM and its keystores” on page 317. A full installation guide for those systems can be downloaded from:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.tklm.doc/cpt/cpt_ic_releas_probsinstallupg.html

We include instructions for both setting up TKLM to serve keys to TS1100 drives through the use of asymmetric keys, and also to LTO4 drives through the use of symmetric keys. The TKLM interface is common across platforms; after TKLM is installed on a system, the keystore setup is quite similar to other TKLM implementations.

10.1 Implementation notes

The Server operating system for this implementation is Red Hat Enterprise Linux Advanced Server 2. We used the GUI installation method, which provides several wizards to ease the implementation.

10.2 Installing TKLM

The TKLM must be installed as root. After downloading the `tklminstall_linux.tar.gz` file, extract it to `/root/tklm`. From there, start the installation script.

To install TKLM as root:

1. Extract the TKLM tar file:

   ```
   tar -zxpvf tklminstall_linux.tar.gz
   ```

   From there, several folders are created.

2. Run the following command from `/root/tklm` to start the GUI installation:

   ```
   ./install.sh
   ```

   Figure 10-1 shows starting the **`install`** script.

```
File  Edit  View  Terminal  Tabs  Help
db2/doc/zh_CN/readme.txt
db2/doc/zh_CN/release.txt
db2/doc/zh_TW/
db2/doc/zh_TW/install.txt
db2/doc/zh_TW/readme.txt
db2/doc/zh_TW/release.txt
db2/db2_install
db2/db2prereqcheck
db2/db2setup
db2/installFixPack
doc/
doc/UsingTIPCharts.doc
doc/UsingBIRTDesigner.doc
install.sh
sample_response.txt
tklmtipia
uninstall_response.txt
[root@localhost tklm]# ls
bin             doc          META-INF             tklminstall_linux.tar.gz
buildinfo.txt   EIC          PostFailureTask.xml  tklmtipia
COI             foo.tar.bz2  sample_response.txt  uninstall_response.txt
db2             HAScripts    screens.tar.bz2
DE              install.sh   tklmbackup.tar.bz2
[root@localhost tklm]# ./install.sh
```

Figure 10-1 Starting the install script

3. Select the language you want to install, as in Figure 10-2, and click **OK**.

Figure 10-2 Language selection

4. Figure 10-3 shows the TKLM manager starting.

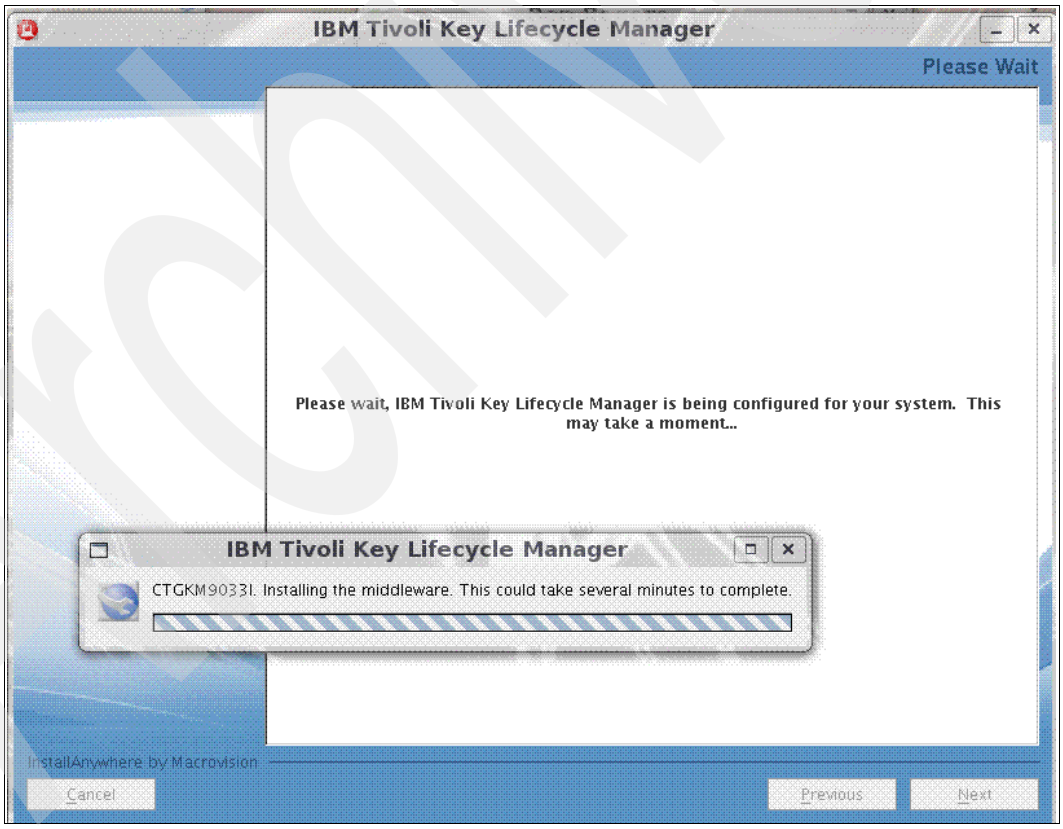


Figure 10-3 Installation starting

The Installation GUI opens, as shown in Figure 10-4 on page 328.

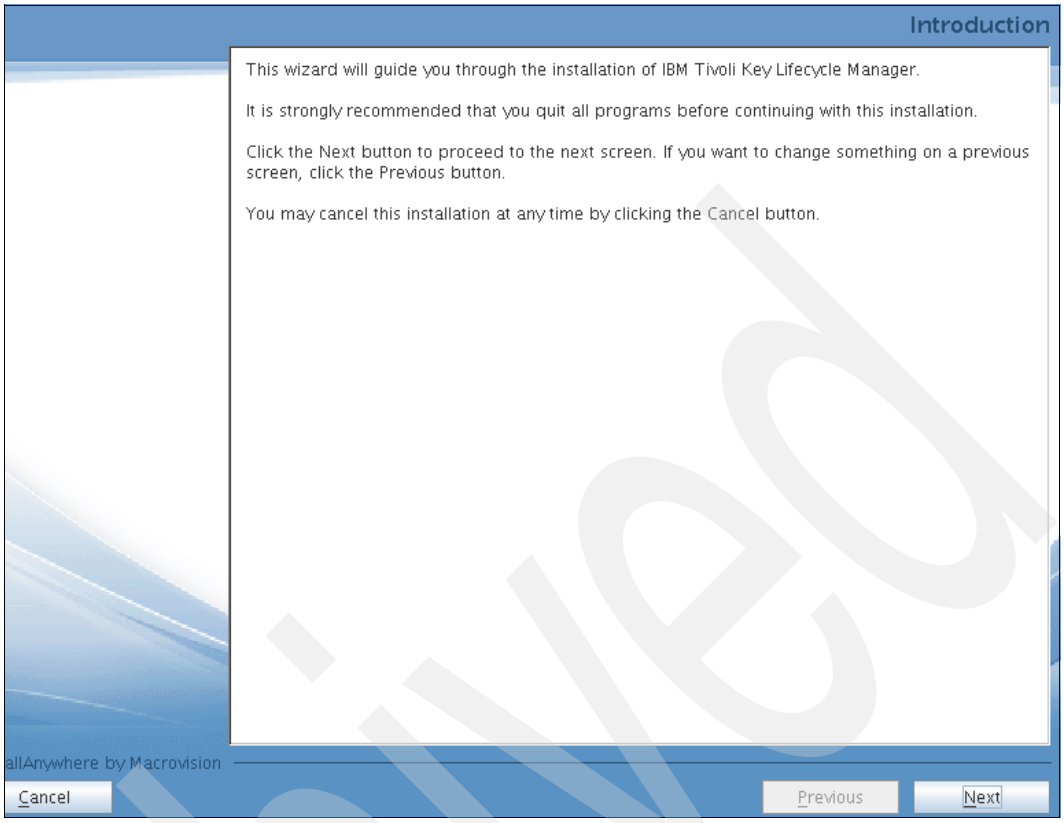


Figure 10-4 Installation GUI

5. Click **Next**.
6. If you agree with the terms of the license, accept them and click **Next**. See Figure 10-5 on page 329.

Figure 10-5 License terms

7. The system on which we are installing TKLM is a clean RHEL AS 2 installation with nothing else on it. In our lab, we accepted the defaults. Choose the DB2 destination, or take the default, and click **Next**.

8. In the next panel, shown in Figure 10-6, accept all of the defaults; enter **password** for all of the passwords and click **Next**.

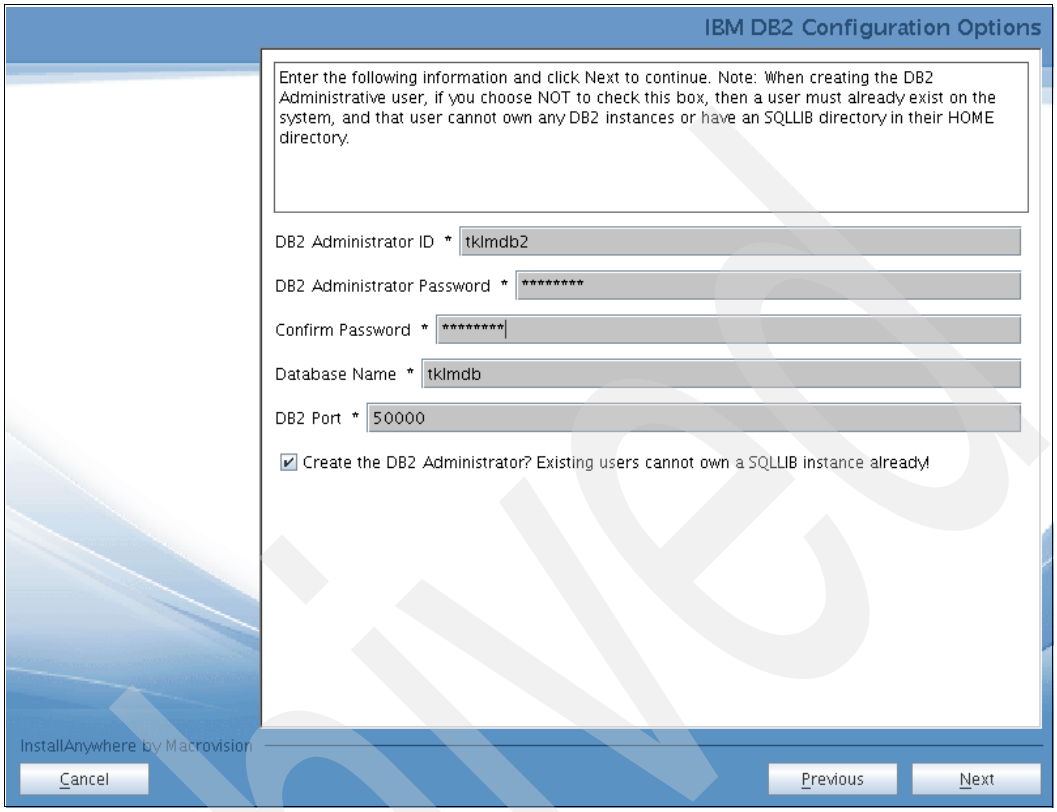


Figure 10-6 Creating the DB2 Administrative user (part 1)

9. In the next panel, shown in Figure 10-7, enter the root group, and click **Next**.

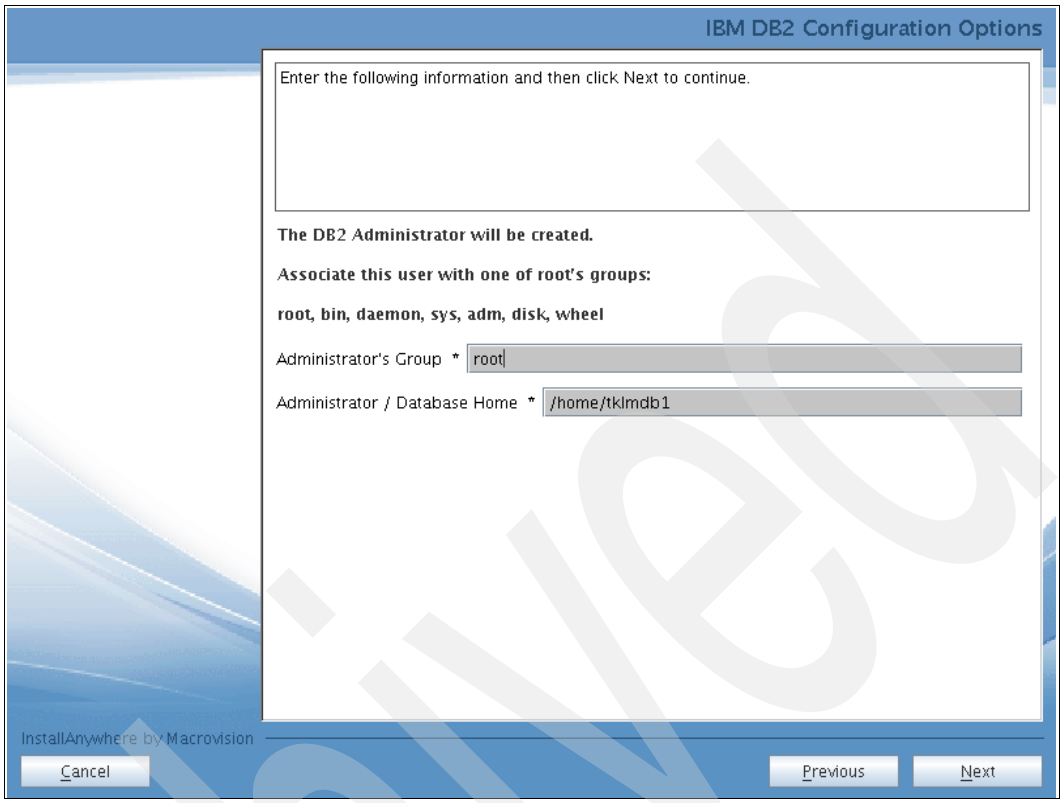


Figure 10-7 Creating the DB2 Administrative user (part 2)

10. In the summary window (shown in Figure 10-8), review the prerequisites.

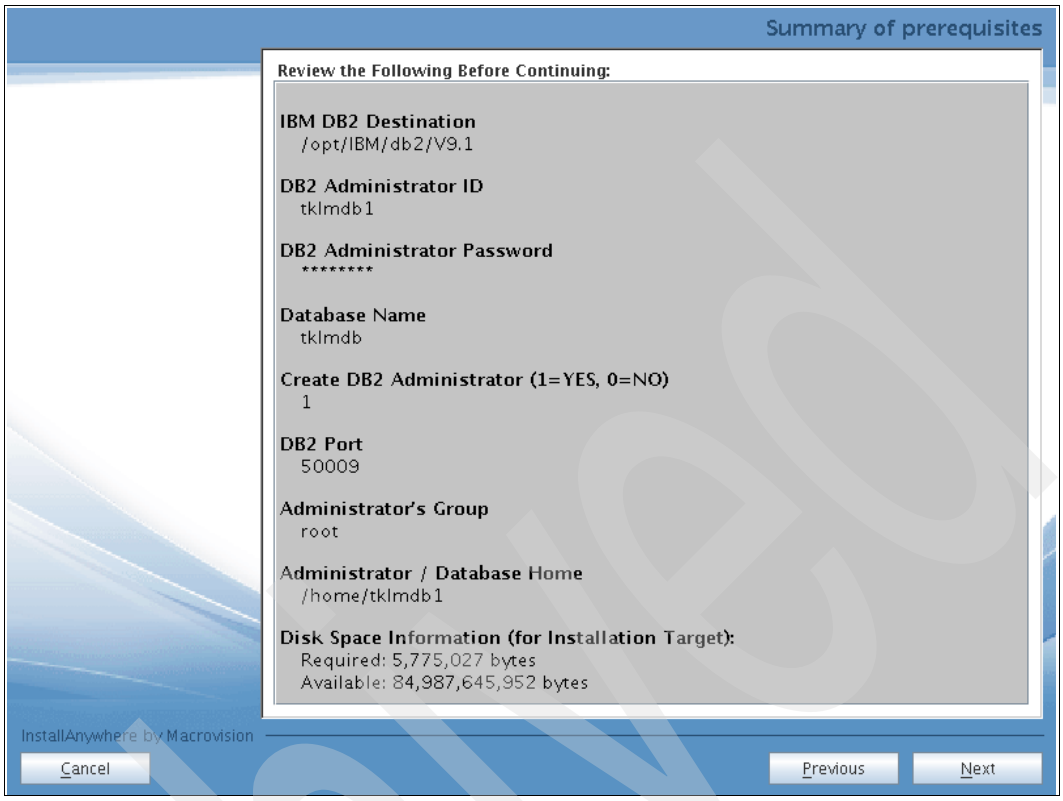


Figure 10-8 Creating the DB2 Administrative user, Summary of prerequisites

11. Click **Next** to accept them and start the DB2 installation.

As shown in Figure 10-9, the DB2 portion of the installation is completed. The TKLM and Tivoli Itegrated Portal (TIP) installation will begin.

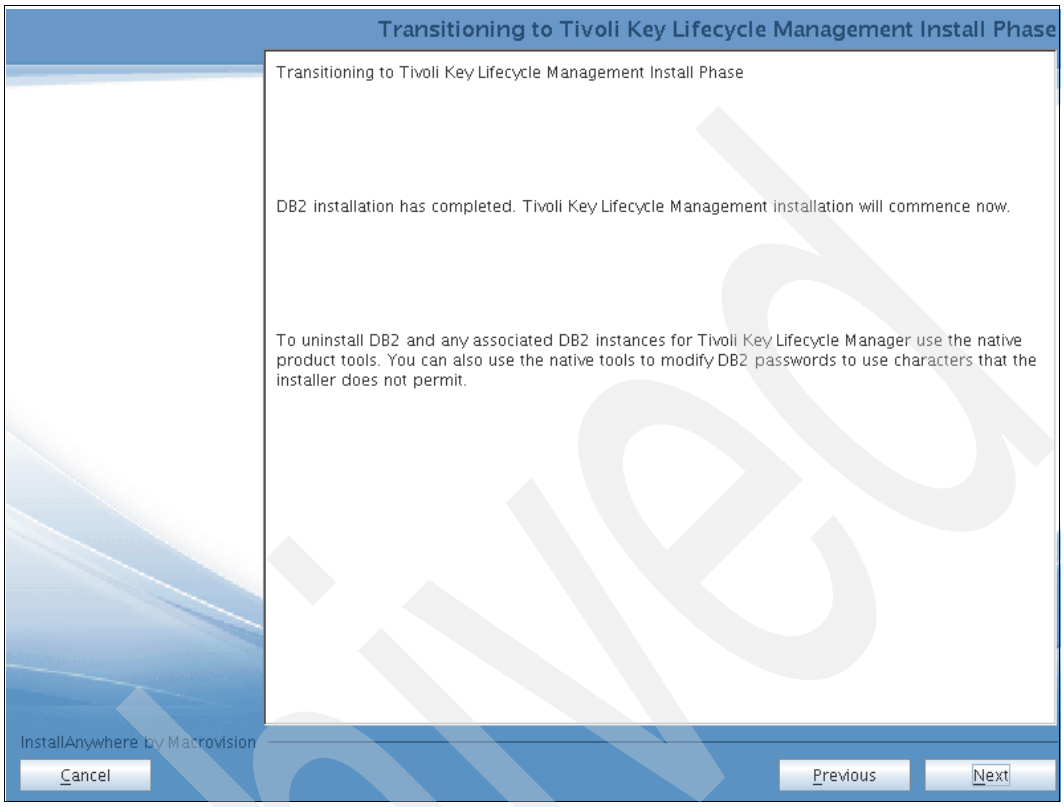


Figure 10-9 DB2 Installation complete

12. As shown in Figure 10-10, choose a directory in which to install the TIP, and click **Next**.

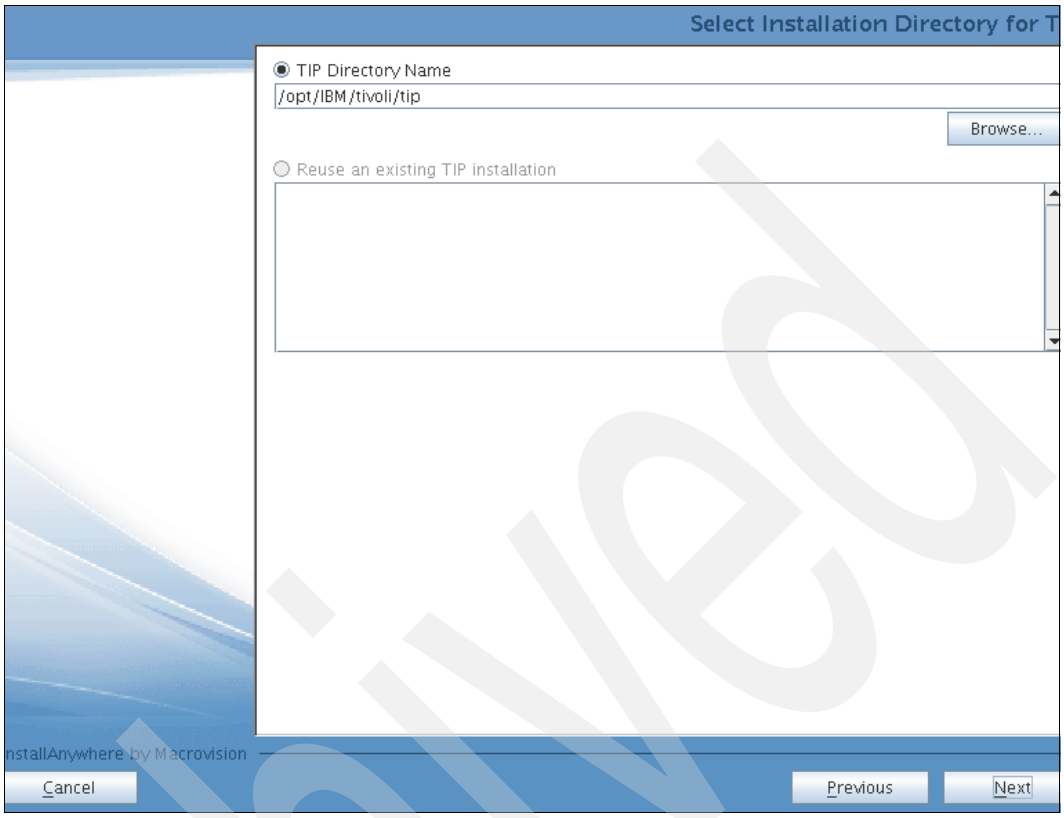


Figure 10-10 Select TIP installation directory

13. Enter a password, as shown in Figure 10-11, and click **Next**.

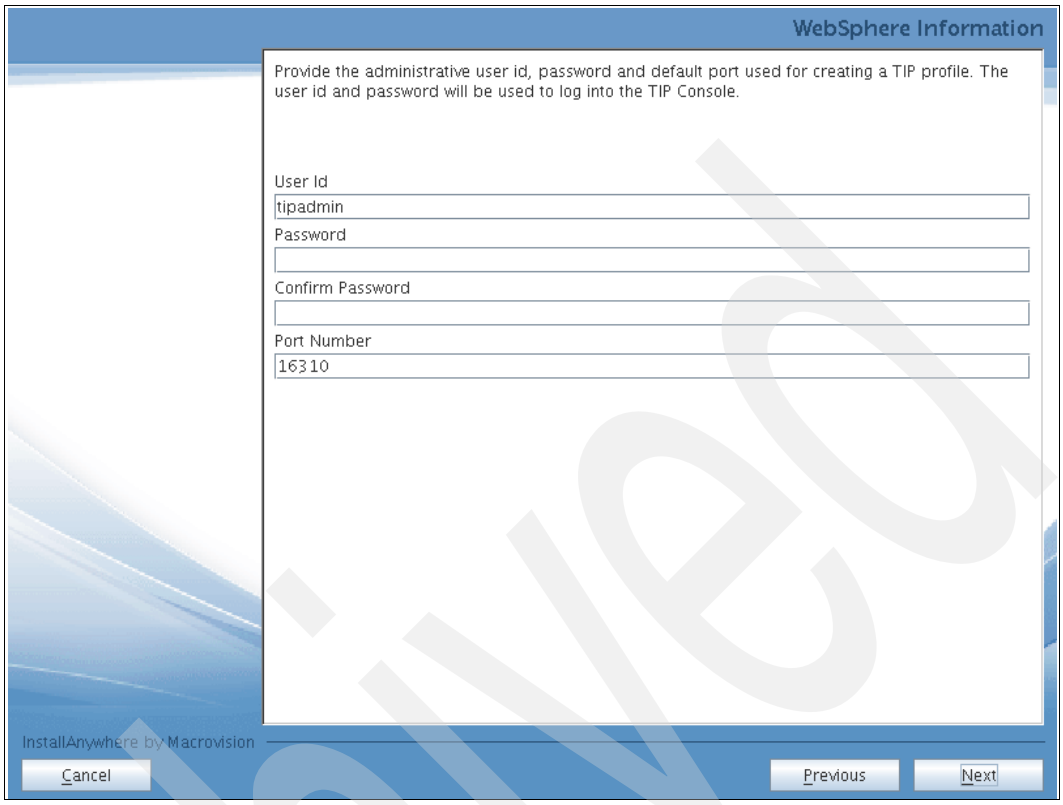


Figure 10-11 Define User ID and Password

14. In the next panel, shown in Figure 10-12, you can begin to migrate an existing EKM implementation. We do not have an EKM on this server, so we will click **Next**.

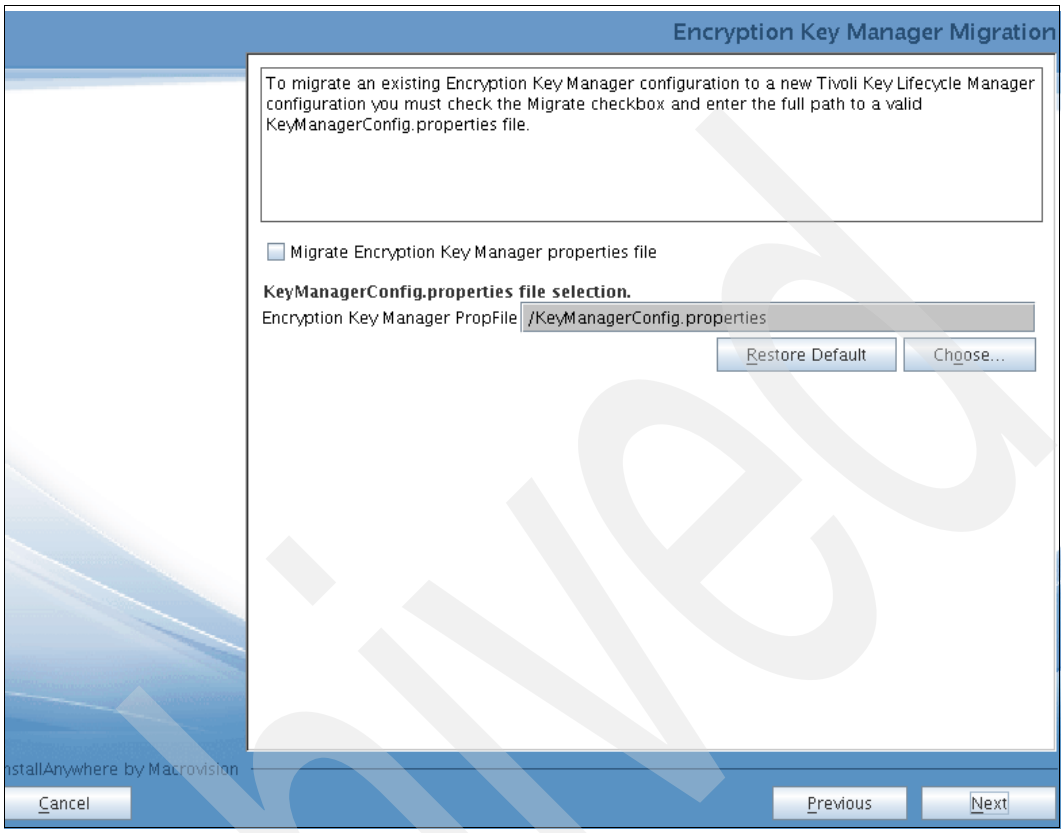


Figure 10-12 EKM Migration

15. The next panel, shown in Figure 10-13, confirms all selections we have made, similar to the DB2 installation. Review the summary.

    If you want to make changes, go back now and make them, otherwise, click **Install**.

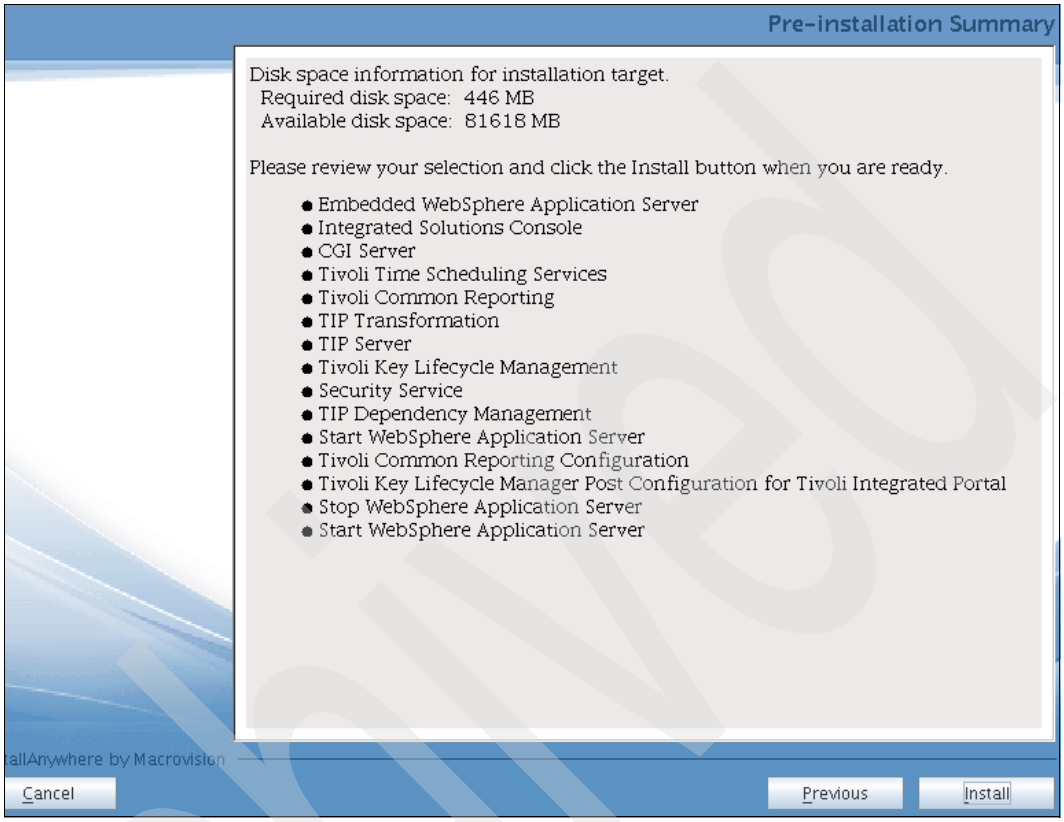


Figure 10-13 Summary of selections

Now that the TKLM is installed, we can configure it.

10.3 Configuring TKLM

Now that TKLM is up and running, we can create a keystore.

To configure TKLM:

1. Open a browser and point it at TKLM. Typically TKLM is at the localhost and localdomain:

   `https://localhost.localdoman:16316/ibm/console`

   A production TKLM most likely is at a different IP or a different URL. The default TKLM installation secures the HTTPS transport with a self-signed certificate. depending on the browser you use, you might get an exception. If that is the case, accept the certificate as a trusted certificate.

   The TIP login window opens as shown in Figure 10-14 on page 338.

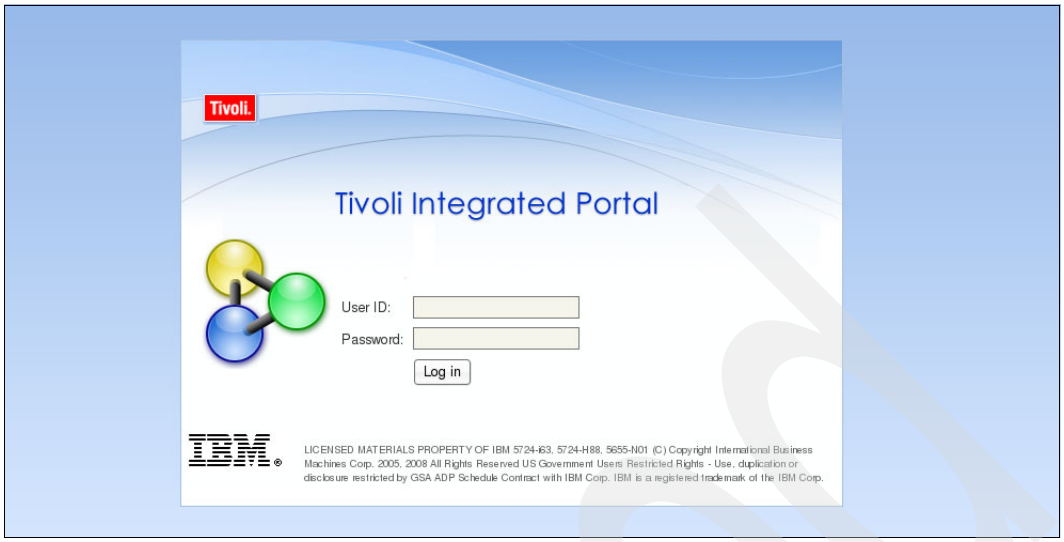


Figure 10-14 Tivoli Integrated Portal

2. Log in to the Tivoli Integrated Portal as user *TKLMAdmin* and the password that was configured during installation. The Welcome window opens, as shown in Figure 10-15.

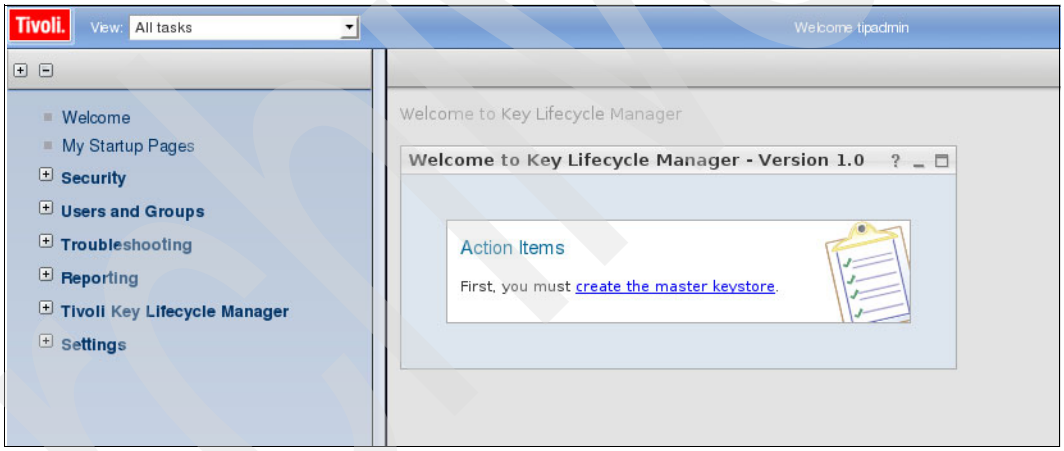


Figure 10-15 The Welcome window

3. Click **create the master keystore**.

   The window shown in Figure 10-16 on page 339 opens.

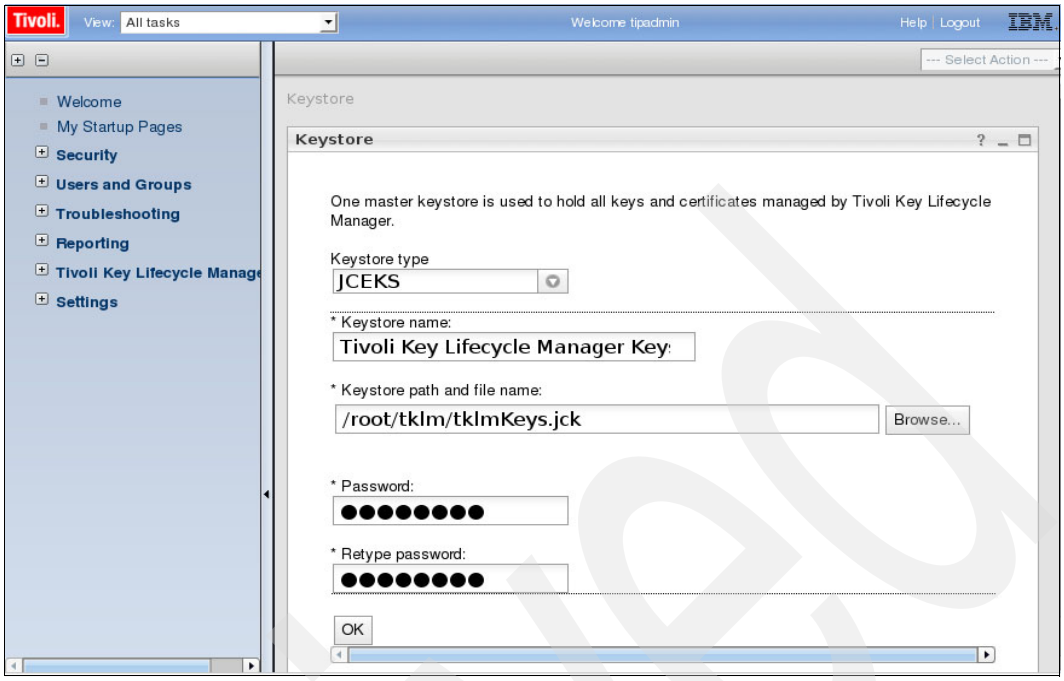


Figure 10-16 Enter keystore information

4. Keystore type is JCEKS. Enter (or browse to) **`tklmKeys.jck`** as the keystore path and file name. JCEKS is the software keystore that supports both asymmetric and symmetric keys. Click **OK.**
5. In the next window, as shown in Figure 10-17, the wizard indicates the Next Steps. Click **Configure the product to use specific communication protocol(s)**.

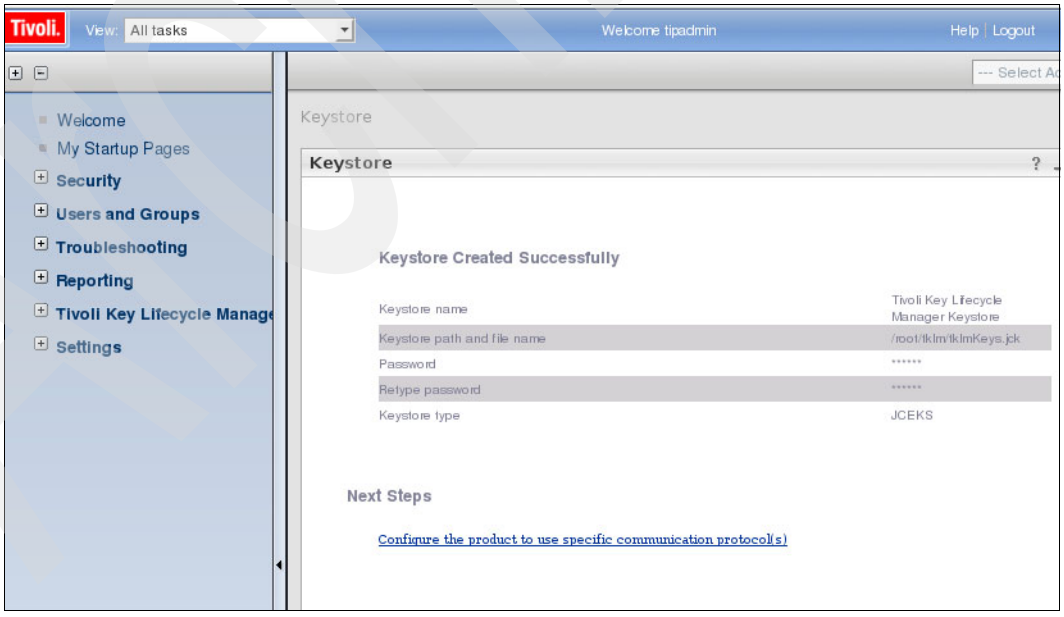


Figure 10-17 Select the next steps

6. In the next window, as shown in Figure 10-18 on page 340, select **Create self-signed certificate**.

If using a third-party, a signed certificate is a requirement that is also supported. We could use an existing certificate from the keystore, but using a certificate for encrypting tape data to also protect the communication protocol is not good practice. Choose a descriptive label and a certificate expiration that follows security guidelines. Optional certificate parameters can also be entered.

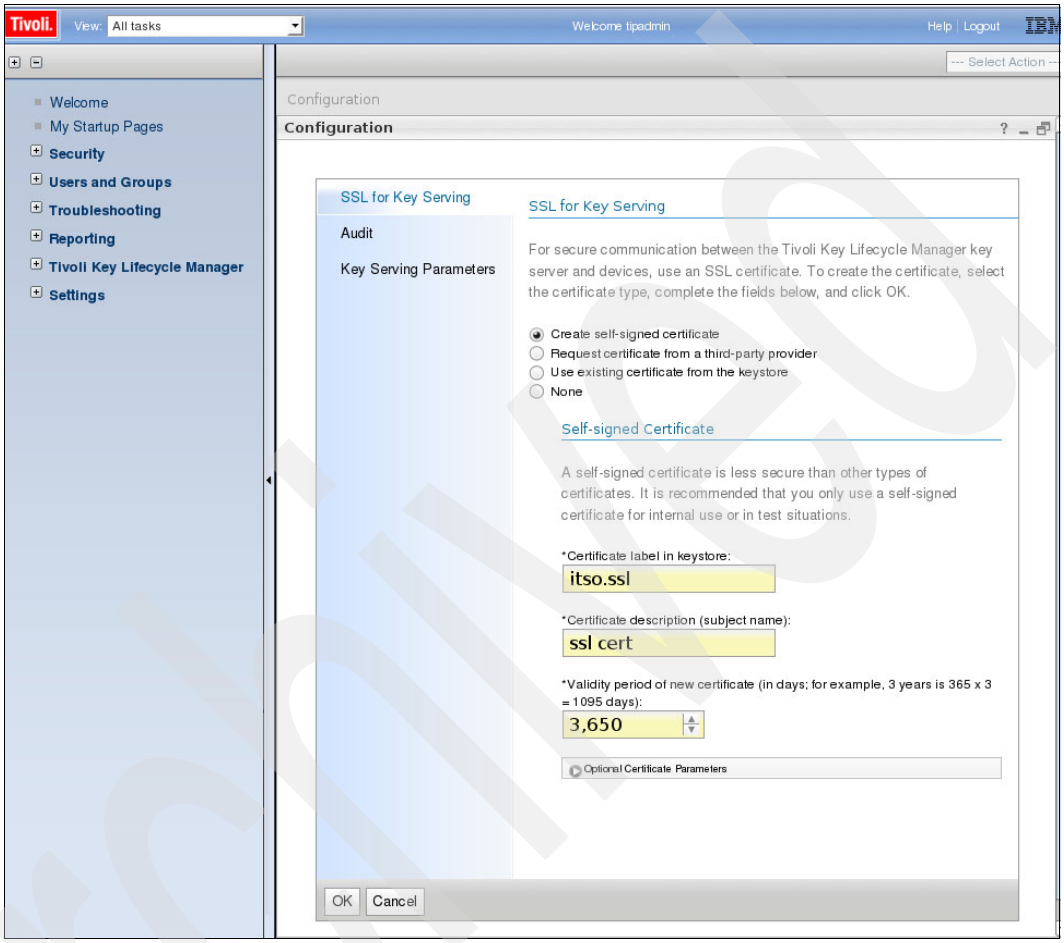


Figure 10-18 Create self-signed certificate

7. Click **OK**. The next window opens, as shown in Figure 10-19 on page 341.

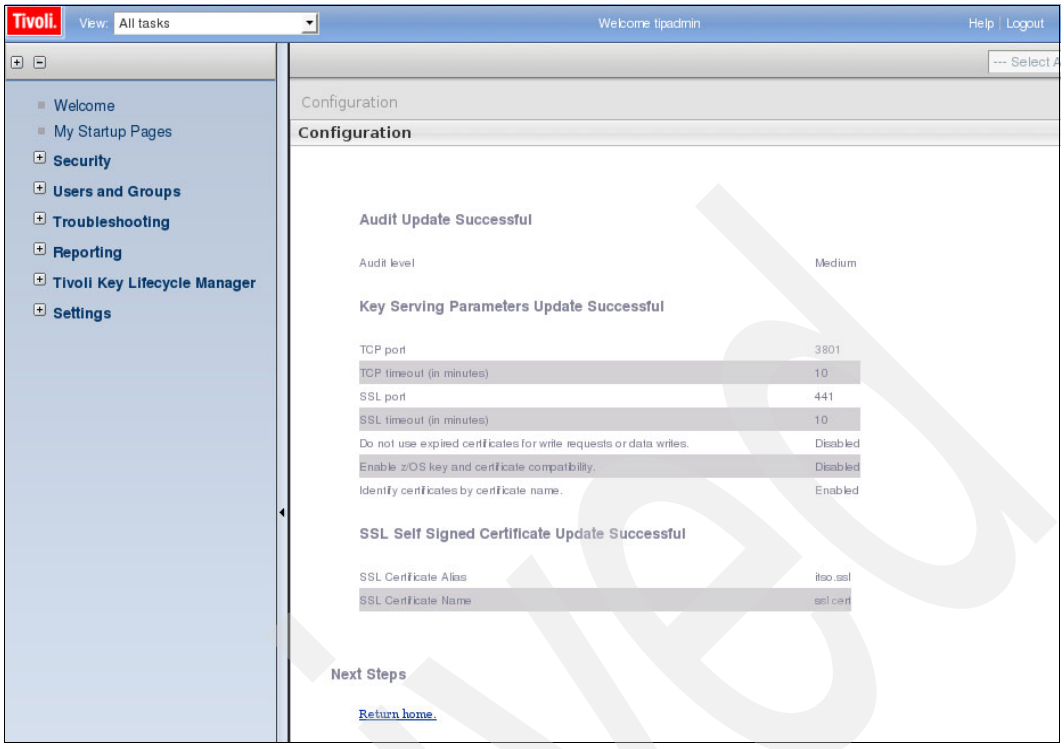


Figure 10-19 Update successful

Our SSL certificate creation was successful.

8. Click **Return home.**
9. In this implementation example we are creating both LTO4 keys and TS1100 keys. Select **LTO** from the menu, as shown in Figure 10-20, and click **Go**.

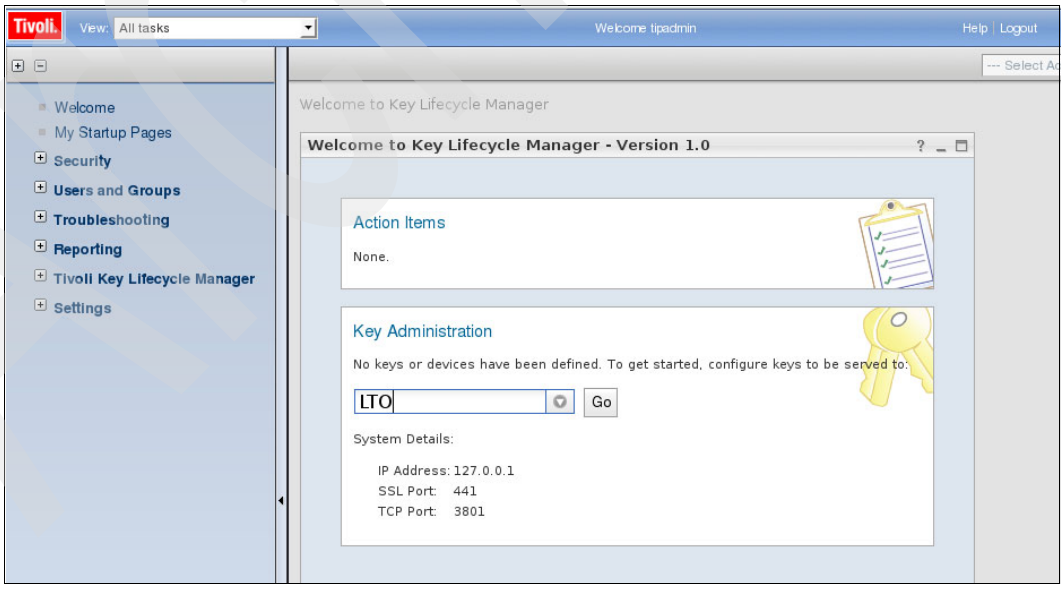


Figure 10-20 Creating keys

10. In the next window, click **Create** (as indicated in Figure 10-21 on page 342) to create a key group.

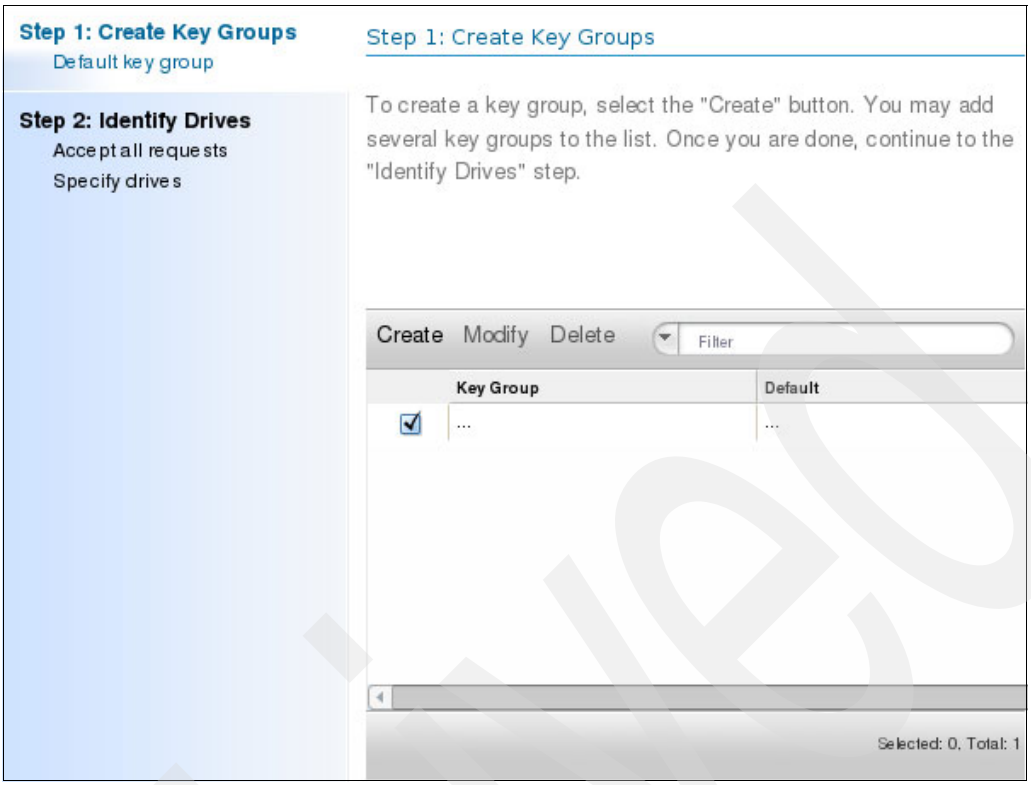


Figure 10-21 Key groups

11. In the next window, Figure 10-22, enter the key group name, the number of keys and a three-letter prefix. Then, click **Create Key Group**.

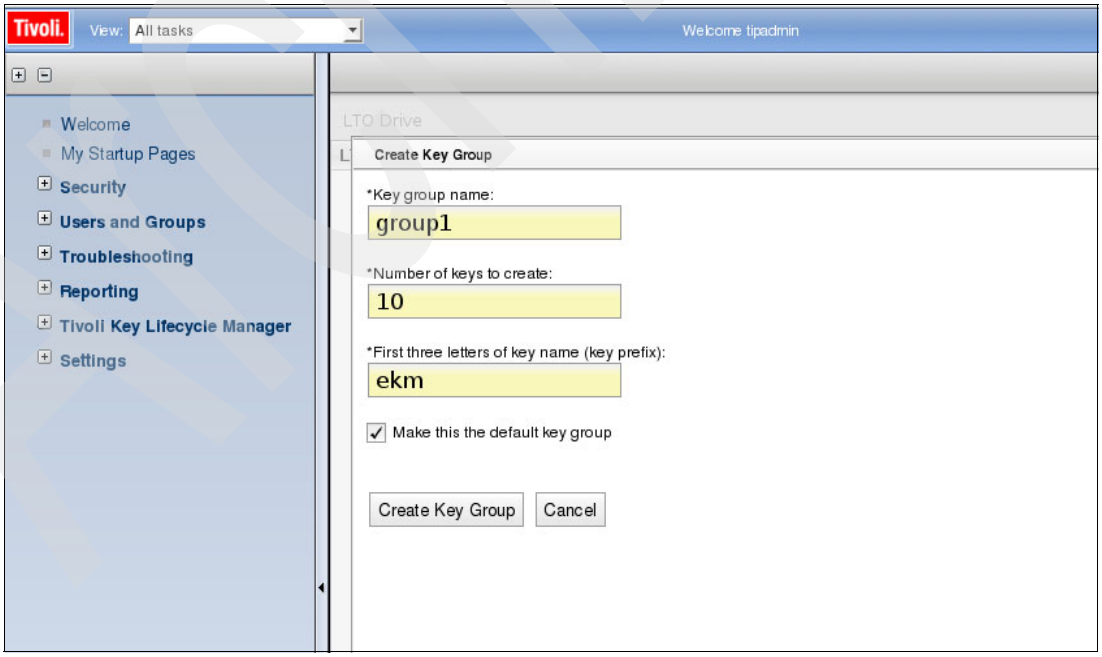


Figure 10-22 Create key groups

12. At the warning message, shown in Figure 10-23 on page 343, click **OK**.

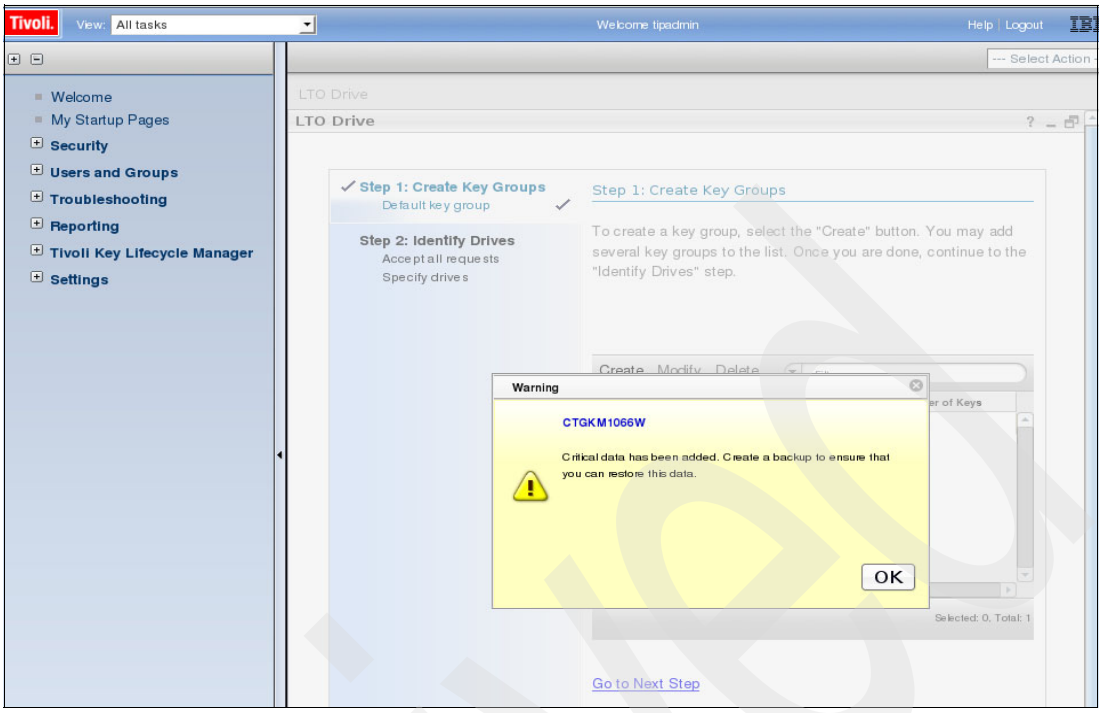


Figure 10-23 Warning message

13. In the next window, Figure 10-24, you can specify the drives from which TKLM will accept requests. Or, to populate TKLM, you can allow TKLM to accept requests from all drives. The best practice is to allow TKLM to accept requests from all drives until TKLM is aware of all the drives and then turn on security to prevent any new requests from unknown drives. At a business continuity site, a best practice is to accept all drives to facilitate a speedy recovery. Click **Return home**.

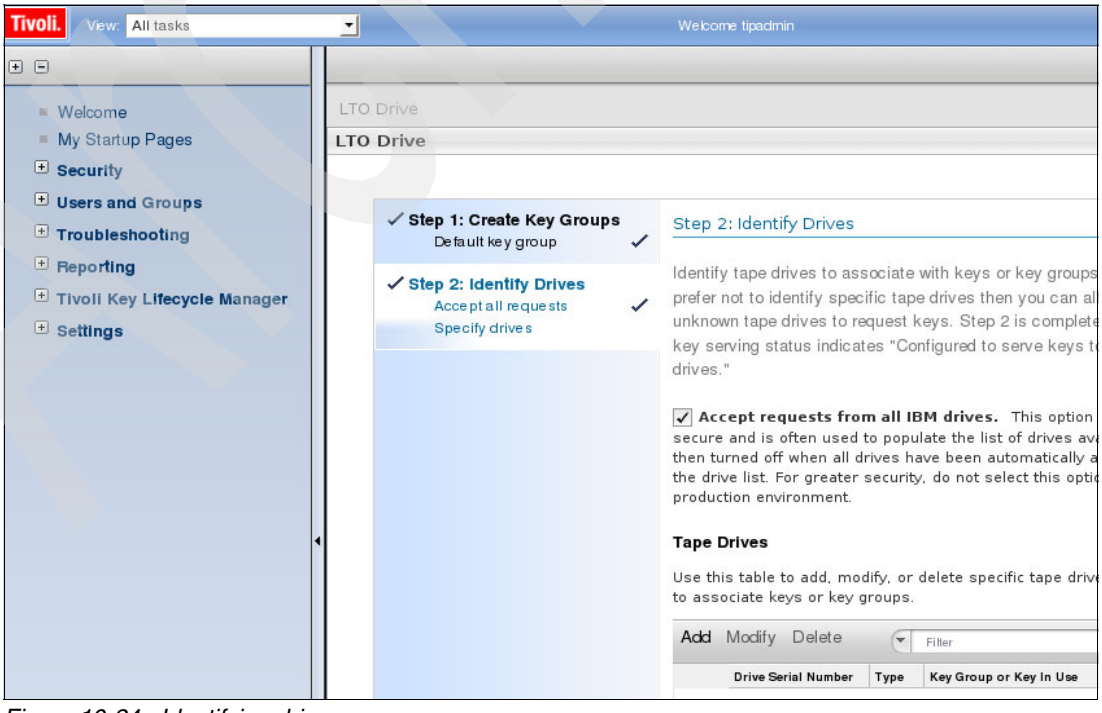


Figure 10-24 Identifying drives

14. Add keys to serve the 3592 drives by selecting 3592, as shown in Figure 10-25. Click **Go**.

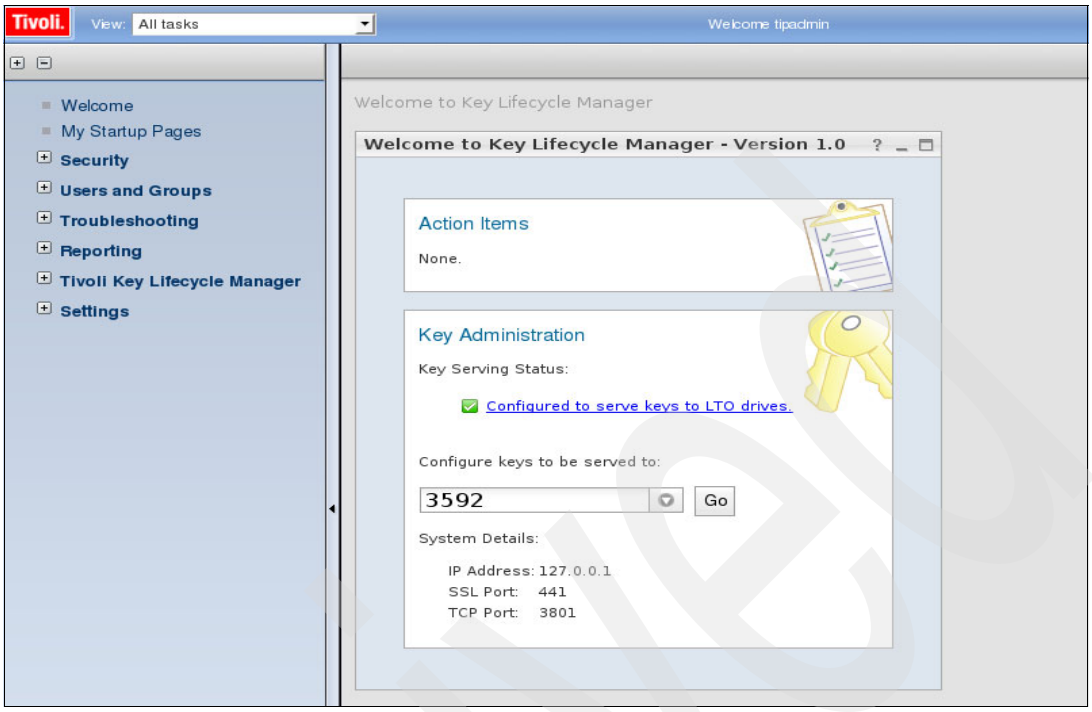


Figure 10-25 Key administration

15. In the next window, shown in Figure 10-26, click **Create**.

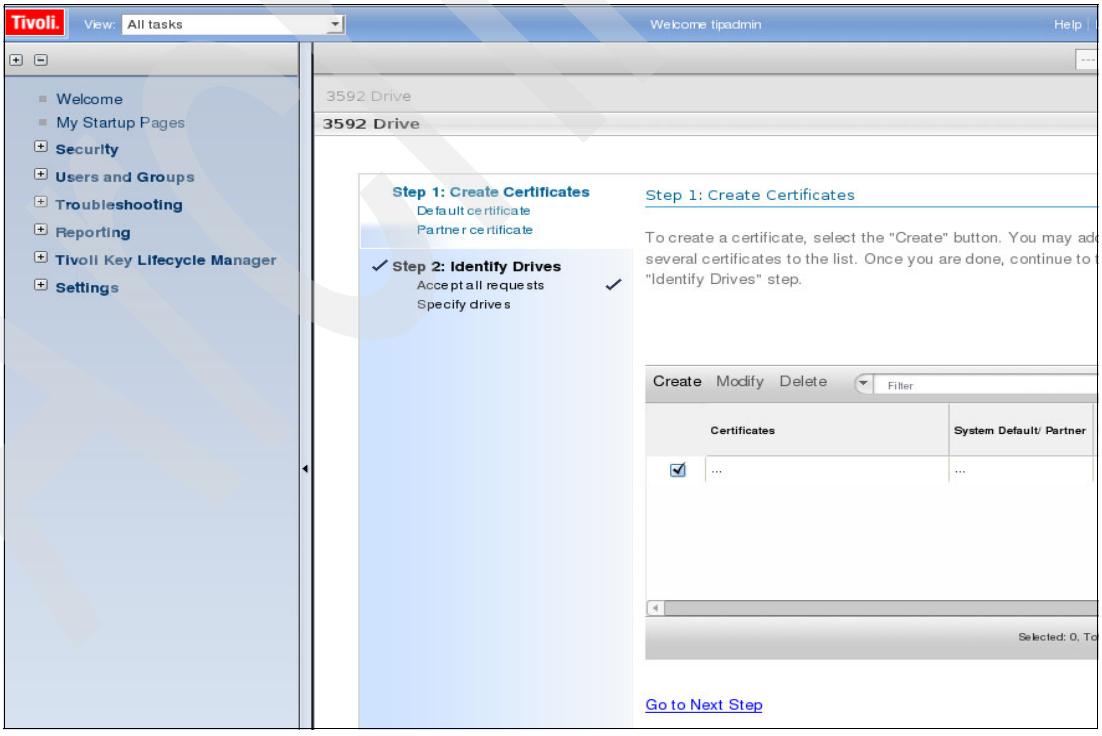


Figure 10-26 identifying drives

16. In Figure 10-27, we create a self-signed certificate. The alias name for our certificate is **tekm.11252009** and it has an expiration of **365** days. The alias name is chosen to be useful and descriptive.A common key naming convention is to use the expiration of the certificate in the label, to show immediately when a certificate will expire, and allow the reuse of the descriptive portion of the alias name. A good practice is have certificates expire at least once a year, hence the 365-day expiration on our certificates.

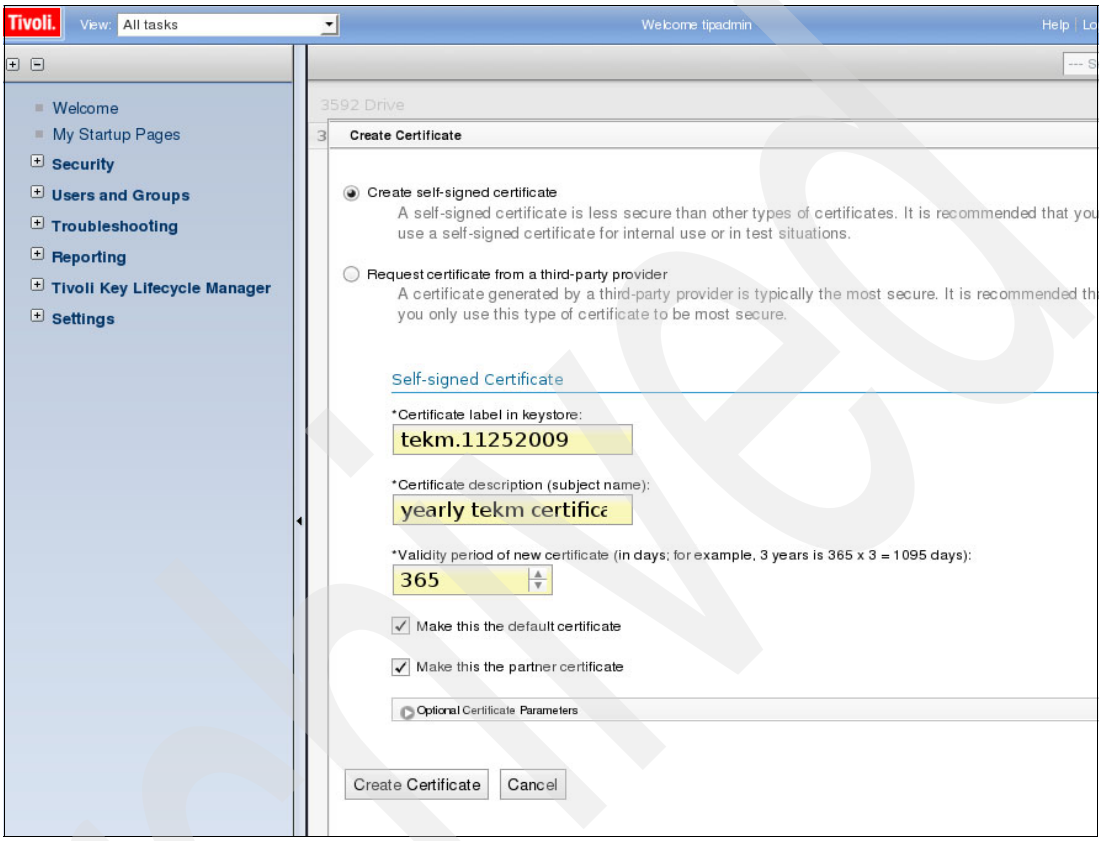


Figure 10-27 Certificate creation

17. Click **Create Certificate**.

18. At the next warning message (see Figure 10-28), click **OK**.

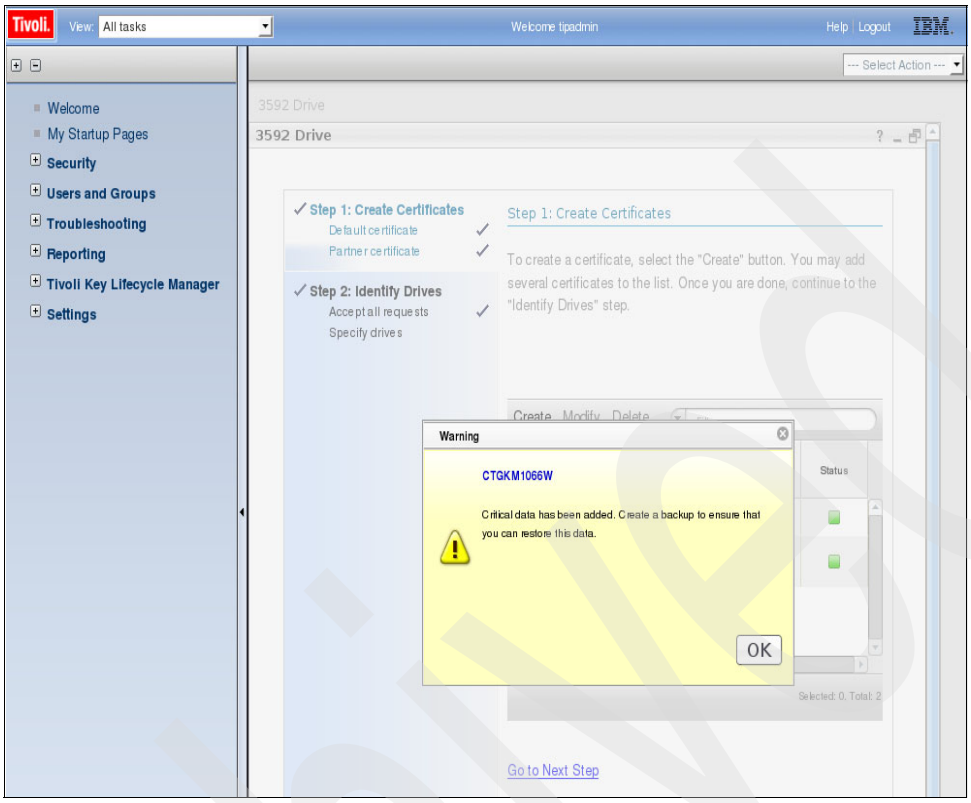


Figure 10-28 Warning message

In the next window, Figure 10-29, the TKLM can now serve keys to both LTO and TS1100 drives.

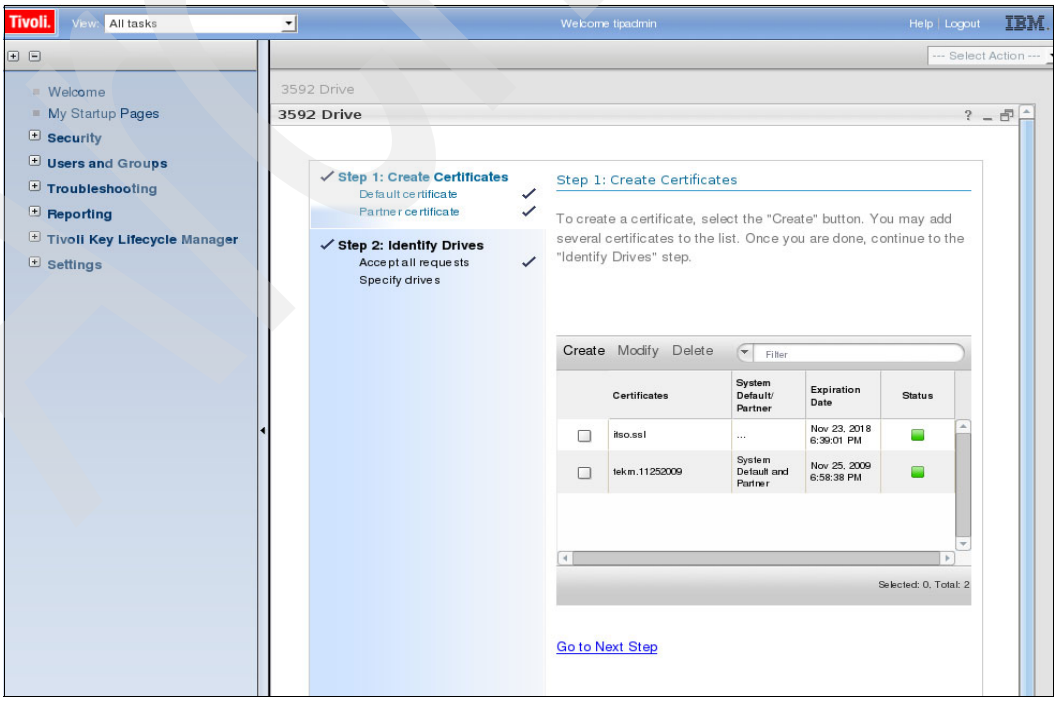


Figure 10-29 serving keys

10.4 Conclusions

The TKLM installation and implementation are significantly easier than setting up an EKM. The TIP GUI gives the user a uniform interface for managing keys, certificates, and tape drives across platforms. From here, a production architecture would have to take into account how to keep keys synchronized between TKLMs, in addition to aggregating log information to keep an audit trail of what tapes were encrypted with what keys and when that happened. The following chapters cover the TKLM functions that enable a user to do this.

11

TKLM operational considerations

This chapter discusses TKLM operational considerations, including the following topics:

- Scripting
- Synchronizing primary TKLM configuration data.
- Maintenance, including adding and removing drives
- Backup procedures
- Removing Web browser certificate warnings

We also include mixed-mode data-sharing examples in which we describe the steps required to share encrypted tapes with a business partner running TKLM or EKM.

11.1 Scripting with TKLM

As with any complex piece of software, routine tasks must be accomplished. By using a GUI, the tasks can be automated and you gain more reliability and predictability. In this section we provide an overview of the scripting interface. This is not intended to replace the TKLM command line reference but to provide several supplemental examples. The scripting interface to TKLM is Jython, a full implementation of Python integrated with the Java platform. For more information about Jython, visit:

http://www.jython.org/Project/

The reference section of the TKLM Information Center contains a complete command line reference, located at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tklm.doc/ref/ref_ic_cli.html

We found that the most useful way of interacting with the command line was to write a Python script and then invoke it from the command line or a launcher.

11.1.1 Simple Linux backup script example

Note that this script is an example and requires some enhancement to be used in production. For instance, including the password in the script is not a good idea.

Example 11-1 invokes the EKLM script on Linux. Example 11-2 shows the contents of `takeBackup.py`, which takes a backup of the currently running TKLM. Example 11-3 shows the output.

Example 11-1 TKLM Script invocation on linux

```
/opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin -password password -lang
jython -f takeBackup.py
```

Example 11-2 The takeBackup.py contents

```
print "Take a Backup of the currently Running TKLM"
print AdminTask.tklmBackupRun('[-backupDirectory /root/TKLMBackup -password
myBackupPwd]')
print "A backup was placed in /root/TKLMBackup with the password myBackupPwd"
```

Example 11-3 Output of Example 11-2

```
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
Take a Backup of the currently Running TKLM
(0) Backup operation succeeded.
==============================


A backup was placed in /root/TKLMBackup with the password myBackupPwd
[root@dyn9011169152 ~]# cd TKLMBackup/
[root@dyn9011169152 TKLMBackup]# dir
tklm_v1.0_20081114153628MST_backup.jar
[root@dyn9011169152 TKLMBackup]#
```

You can see from Example 11-3 on page 350 that the script in Example 11-2 on page 350 first prints out sample text (nothing complicated here). Then, the script invokes **tklmBackupRun** to place the backup in /root/TKLMBackup with the password myBackupPwd, and then prints more descriptive text. The backup operation can easily be added to any script that takes action against the TKLM to capture a before and after snapshot. Enhancing this script could also be used to automate synchronizing TKLM servers by setting up a cron job or windows task scheduler to take a backup, copy it to the secondary TKLM and restore the backup.

11.2 Synchronizing primary TKLM configuration data.

For each keystore, you should define one TKLM as the primary. This primary keystore is the one on which to make changes. Changes are then replicated on the secondary TKLM servers. When selecting the TKLM servers, at a minimum they must be running the same OS and the secondary TKLM must have available the same amount or more free disk space. Matching the two servers as closely as possible is desirable. You should then install TKLM using the same DB2 and TIP settings so that a backup from the primary TKLM can be restored on the secondary TKLM server.

11.2.1 Setting up primary and secondary TKLM servers

For this example we are running two Windows 2003 hosts running under VMware® ESX. This allowed us to easily match the environment seen by TKLM and its middleware. For the purposes of this example we chose not to clone the VMware image, but instead we performed default installations using the same configuration and passwords for DB2, TKLM, and the keystores. To ensure the installations were the same, we recorded the primary TKLM installation to a response file; because, the response file it created was incomplete and did not allow the installer on the secondary machine to run, we instead used the graphical installer and kept defaults the same as with the first machine.

Note: TKLM does not tolerate spaces in the installation directory name. You cannot install to c:\Program Files or to any other directory that contains a space. The graphical installer defaults to c:\ibm which works fine.

A good practice is to fill out the installation worksheets found in the *Tivoli Key Lifecycle Manager Installation and Configuration Guide,* SC23-9977.

Note: The TKLM installer does not always set up the required services to start automatically so be sure to test your TKLM installation after you have enabled the services and restarted the server. For more information, refer to 9.3.2, “EKM to TKLM migration” on page 322.

You may also obtain the product documentation from the IBM Tivoli Key Lifecycle Manager Information Center available at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tklm.doc/welcome.htm

Administration information is presented in HTML. The following guides are also provided in PDF files:

- IBM Tivoli Key Lifecycle Manager Quick Start Guide
- IBM Tivoli Key Lifecycle Manager Installation and Configuration Guide

11.2.2 Synchronizing the primary and secondary TKLM servers

TKLM 1.0 does not automatically synchronize between servers but it does provide a convenient backup and restore operation that can be performed using the command line or Web user interface. Synchronization involves backing up a TKLM and then restoring to a different server with the same configuration parameters. Consider the following notes:

- Select one machine to be the primary, and originate all backups from the primary. All changes should be made on the primary and then deployed through a backup or restore to the secondary TKLM.
- Primary and secondary TKLMs must be running the same OS with the same user accounts for TKLM, TIP, and DB2.
- Restores are disruptive to the secondary TKLM, so ensure that the primary is active and serving key's before performing the restore.

See section 11.4, "TKLM backup and restore procedures" on page 361

11.3 TKLM maintenance

TKLM is intended to help automate some of the drudgery around key management. This does not mean that you can setup TKLM once and then forget about it. As drives enter and leave the environment, TKLM has to be updated. Although you can schedule key rollover and pregenerate the keys, do not do this too far in advance otherwise you risk compromised future keys in addition to current keys.

This section provides information about:

- TKLM drive management using the GUI and CLI
- LTO key group rollover
- 3592 certificate rollover
- User management

11.3.1 Adding and removing drives

TKLM tracks which drives it knows about and allows you to group them depending on your requirements. You may configure TKLM to add any drive that requests a key or to only allow known drives to obtain keys. As with EKM, allowing any drive that requests a key to be automatically added, initially to populate the list, and then disabling the adding of new drives manually is a good compromise between security and simplicity. To allow any drive to be added, check the **Accept requests from all IBM drives** check box. See Figure 11-1.

Figure 11-1 Accept requests form all IBM drives

When adding drives to TKLM, you have to know three key pieces of information: drive serial number, World Wide Node Name (WWNN), and the key group or certificates you would like associated with it. You can obtain this information in several ways, depending on the tape library or host software attached to the drive. One convenient way, with the TS3500, is to download the drive CSV log. You obtain the log by using the TS3500 Web interface. Select **Drives** → **Drive Summary**, and then select name of the *Drive Statistics(.csv)*. You may also

download the file `http://<library ip>/FS/LIBLG_01_DS.csv` (Note that the drive serial number and drive WWNN have a leading underscore character (_) character that has to be removed. Because TKLM supports saving only 4 bytes of the WWNN, select the last 4 bytes.

Using the GUI

Note: Depending on the system, the GUI can take awhile to initially load. Wait for it. If you start getting unusual-looking pages, you probably started clicking links before the GUI finished loading. If that happens, log out, log in again, and then wait for the page to finish loading.

The GUI is simple but slower. Simply log in and go to **Tivoli Key Lifecycle Manager** → **Key Administration** → **LTO**, or **Tivoli Key Lifecycle Manager** → **Key Administration** → **3592**. From the Add menu, select **Drive**. Refer to Figure 11-2. A new window opens.

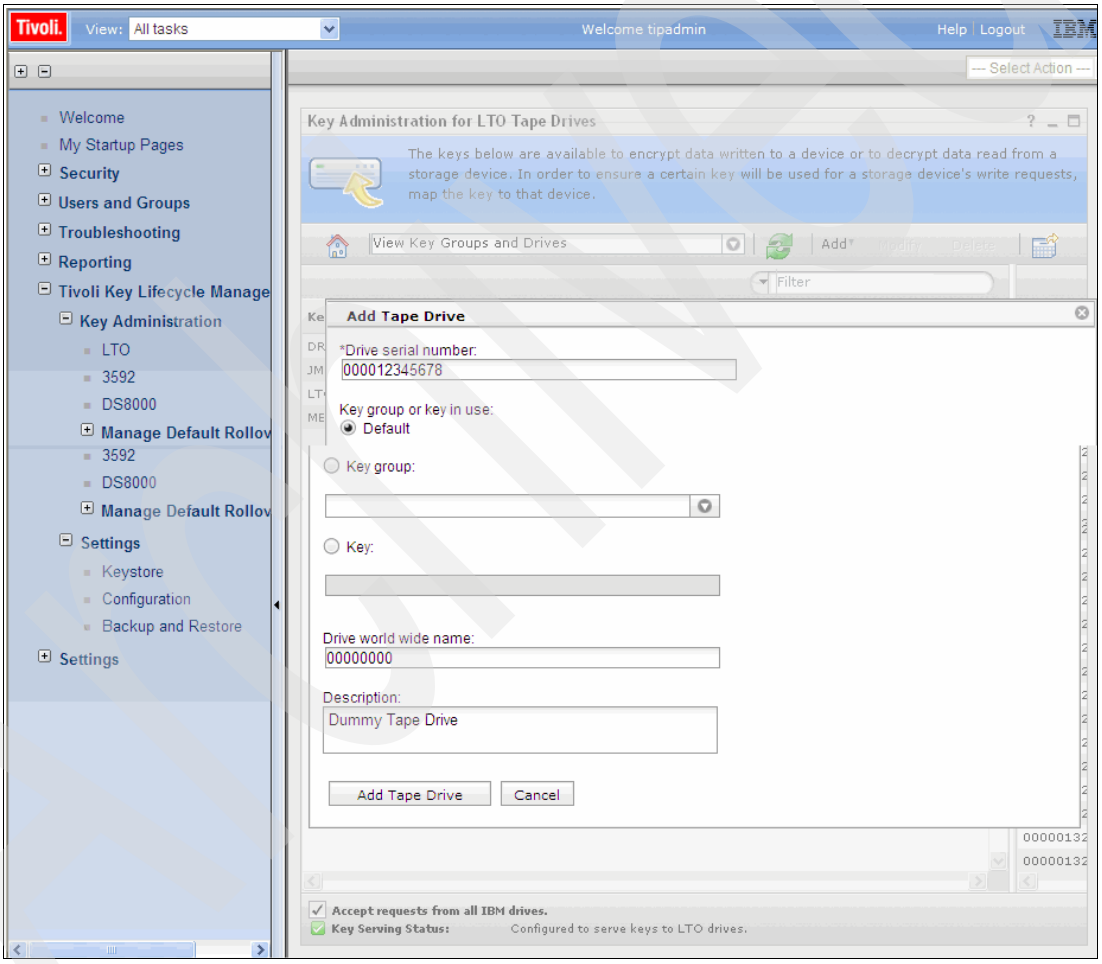


Figure 11-2 Add a tape drive using the GUI

Using the command line to add drives

For more information about the TKLM command line interface see the information center:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tklm.doc/ref/ref_ic_cli.html`

Starting the Windows command line

From the TIP_HOME directory (usually `c:\IBM\tivoli\tip\bin`), run the command:

```
wsadmin -username TKLMAdmin -password password -lang jython
```

Starting the AIX, Solaris, RHEL, or SLES command line

From the TIP_HOME directory (usually `/opt/IBM/tivoli/tip/bin`), run the command:

```
./wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Adding drives using the Jython command line

To add a drive, you have to know its serial number and type LTO or 3592. You can then add a drive or more likely a set of drives. Example 11-4 starts the command line and adds a set of drives to the default key group. In the example, the `addDrive.sh` script starts the command line and tells it to run the `addDrive.py` script. Which then adds two LTO and one 3529 drives.

Example 11-4 Start command line and run Jython script to add 3 drives to TKLM

```
[root@dyn9011169152 ~]# cat addDrive.sh
/opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin -password password -lang
jython -f addDrive.py
[root@dyn9011169152 ~]# cat addDrive.py
print "Add a LTO Drive with the serial number 000012345670"
print AdminTask.tklmDeviceAdd('[-type LTO -serialNumber 000012345670 -attributes
"{worldwideName abcd0100} {description TS3500Frame2Drive1}"]')
print "Add a LTO Drive with the serial number 000012345671"
print AdminTask.tklmDeviceAdd('[-type LTO -serialNumber 000012345671 -attributes
"{worldwideName abcd0101} {description TS3500Frame2Drive2}"]')
print "Add a 3592 Drive with the serial number 000012345672"
print AdminTask.tklmDeviceAdd('[-type 3592 -serialNumber 000012345672 -attributes
"{worldwideName abcd0200} {description TS3500Frame3Drive1}"]')

print "List out all the drives in the TKLM"
print AdminTask.tklmDeviceList()

[root@dyn9011169152 ~]#
```

The output is shown in Example 11-5 on page 355.

You may also automate adding the drives to a specific key group using the `aliasOne` and `aliasTwo` attributes for 3592 drives or the `symAlias` attribute for LTO drives. Note that because the attributes are already quoted, using descriptions or aliases with spaces might be difficult.

Example 11-5 Sample run of script in Example 11-4 on page 354

```
[root@dyn9011169152 ~]# ./addDrive.sh
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
Add a LTO Drive with the serial number 000012345670
CTGKM0001I Command succeeded.
Add a LTO Drive with the serial number 000012345671
CTGKM0001I Command succeeded.
Add a 3592 Drive with the serial number 000012345672
CTGKM0001I Command succeeded.
List out all the drives in the TKLM
CTGKM0001I Command succeeded.

Description = salesDivisionDrive
Serial Number = 000012345678
Device uuid = DEVICE-0012d3cf-89d5-412b-8a19-1897e7ce1146
Device type = LTO
World wide name = abcd1234

Description = TS3500
Serial Number = 100012345678
Device uuid = DEVICE-0a6aa209-b58a-482f-8e0d-dc128b129988
Device type = LTO
World wide name = 0bcd1234

Description = TS3500Frame2Drive1
Serial Number = 000012345670
Device uuid = DEVICE-65cae709-3132-4aaf-831e-6a32b15d0477
Device type = LTO
World wide name = abcd0100

Description = TS3500Frame2Drive2
Serial Number = 000012345671
Device uuid = DEVICE-11719c55-37ba-4a1d-997e-dba59387c17a
Device type = LTO
World wide name = abcd0101

Description = TS3500Frame3Drive1
Serial Number = 000012345672
Device uuid = DEVICE-ee7a0ce9-aaf3-4e79-a059-395f7f98ecac
Device type = 3592
World wide name = abcd0200

[root@dyn9011169152 ~]#
```

11.3.2 Scheduling key group rollover

One of the advantages of TKLM is that you can schedule it to change key groups at a predetermined time without human intervention. Because keys and key groups do not expire like certificates, you do not have to worry about keys expiring. However, you should still adhere to the key usage time guidelines set by your organization.

Note: With TKLM 1.0, you can only schedule key group rollover for the default key group.

To create a key group, go to **Tivoli Key Lifecycle Manager** → **LTO,** and select **Add**. The panel in Figure 11-3 opens. Fill out the key data according to your requirements.

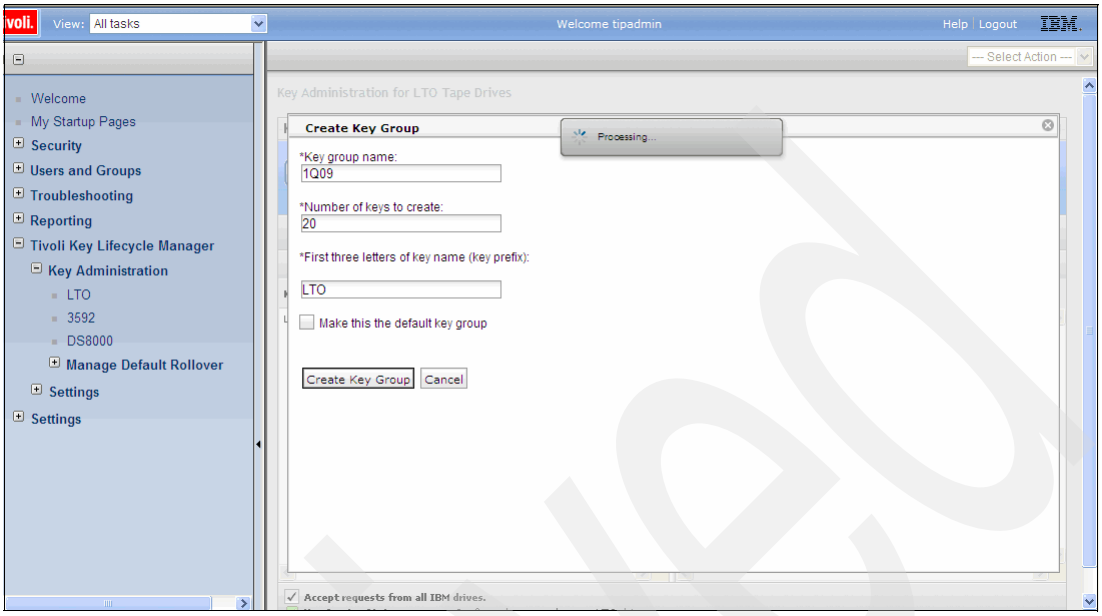


Figure 11-3 Creating a key group

After creating the new key group, you may schedule when it should become the new default key by selecting the calendar icon as shown in Figure 11-4.

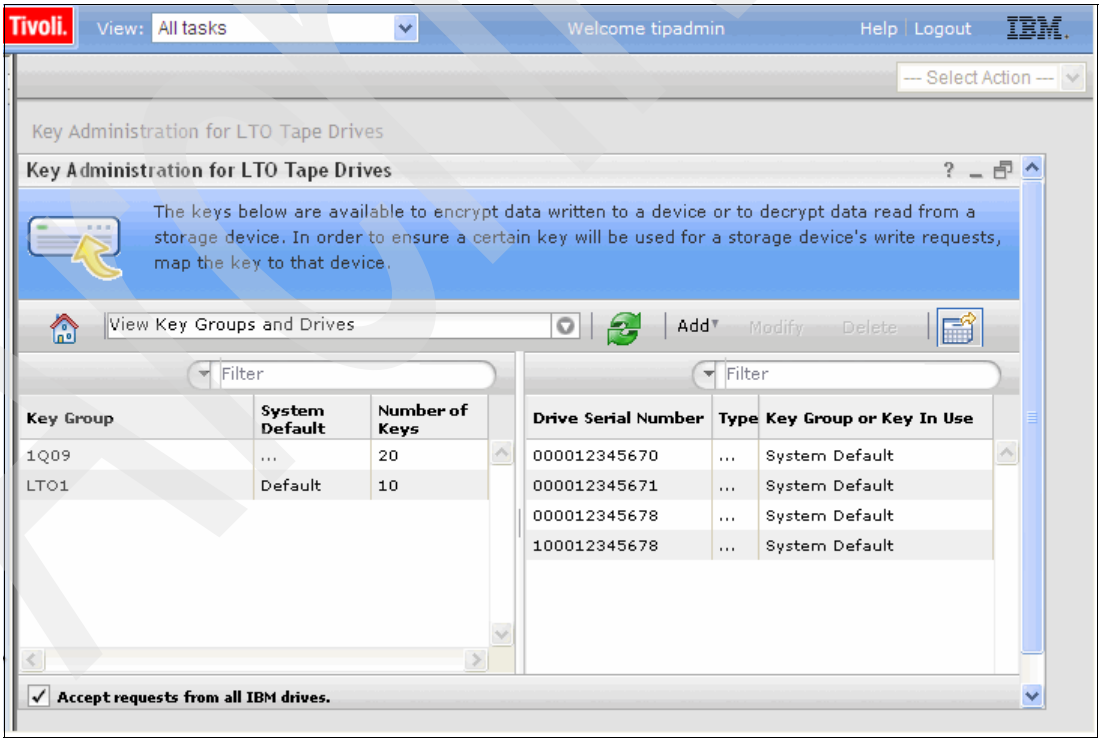


Figure 11-4 Scheduling a key group rollover

A future default panel opens, as shown in Figure 11-5.

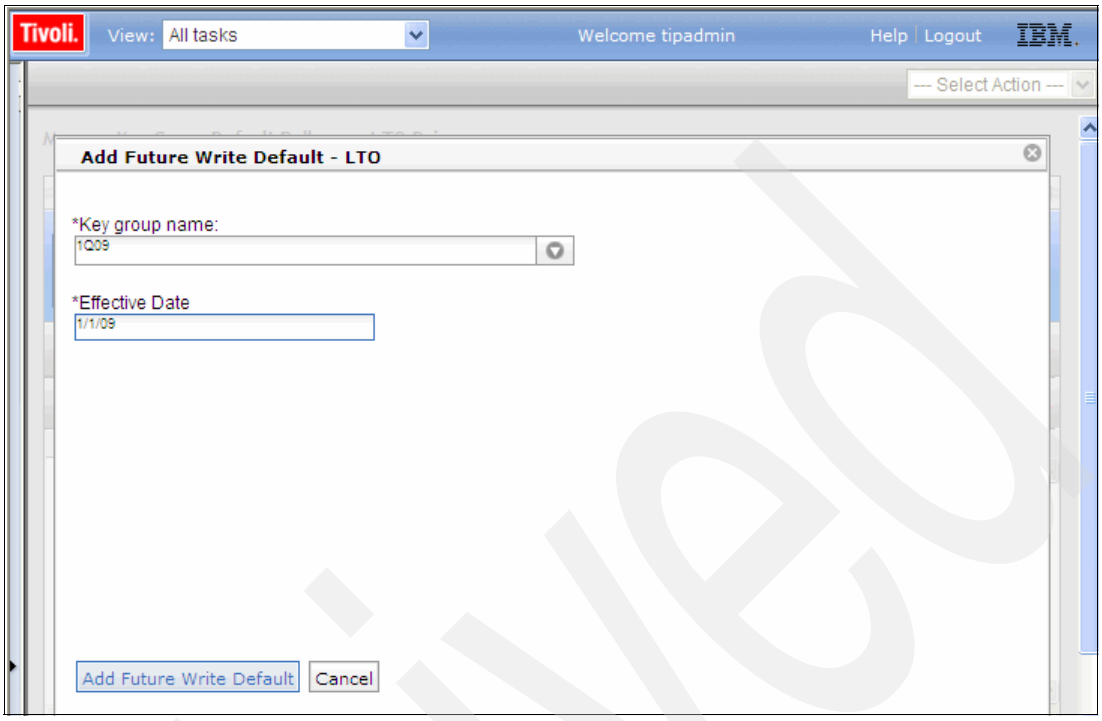


Figure 11-5 Setting the key group rollover

After you set the key group you want to change to, and the date on which you want TKLM to make the change, the key change schedule opens, as shown in Figure 11-6 on page 358.

Note: Key rollovers occur based on the operating system time of the TKLM server. To ensure that rollovers occur when you expect, having the TKLM server use NTP is a good practice. Another point to consider is time zones. If your disaster recovery site is in another time zone, there might be a time window where the two key servers are serving different keys.

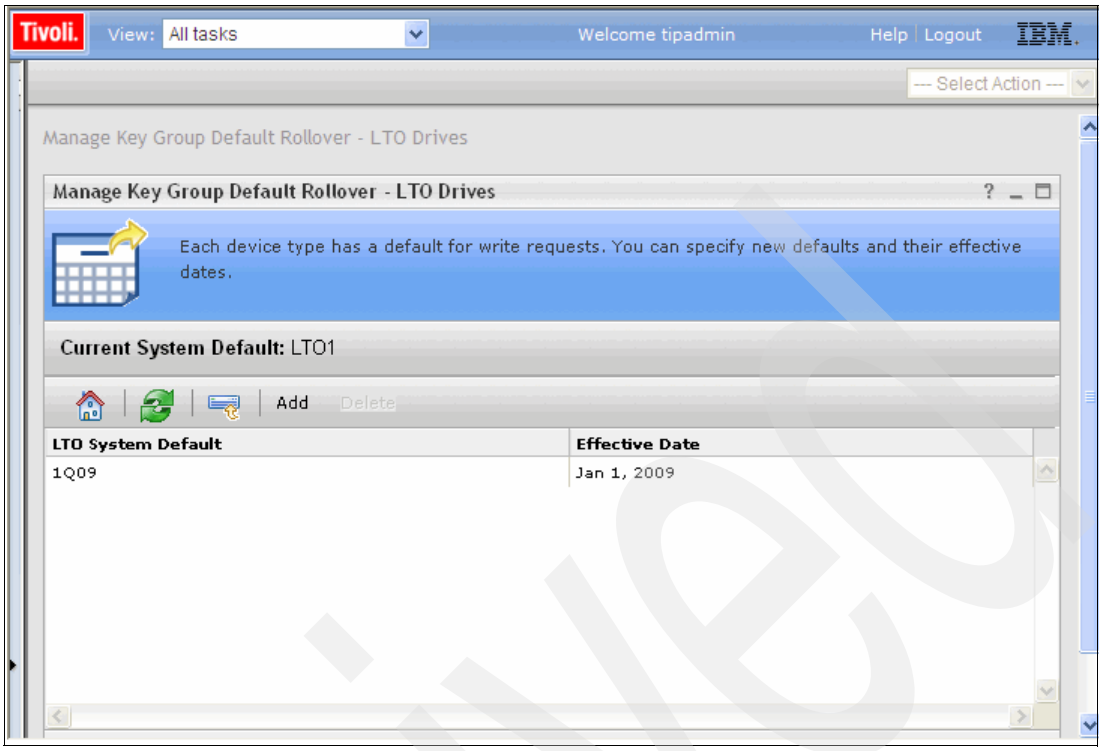


Figure 11-6 Checking the key group schedule

11.3.3 Scheduling certificate rollover

One of the advantages of TKLM is that you can schedule it to change certificates at a predetermined time with out human intervention. However care must be taken when doing this as the certificates are generated at creation time not certificate rollover time. This means, when you create a certificate that will take effect at a later date you must not expire the certificate until after the next scheduled certificate change. For example, if you have a default certificate that expires in one week but your policy is to change certificates on a three-month cycle, when you create the certificate, you must set them to expire in no sooner than three months and one week.

To create a new certificate select **Tivoli Key Lifecycle Manager** → **Key Administration** → **3592**, and then select **Add**, as in Figure 11-7.

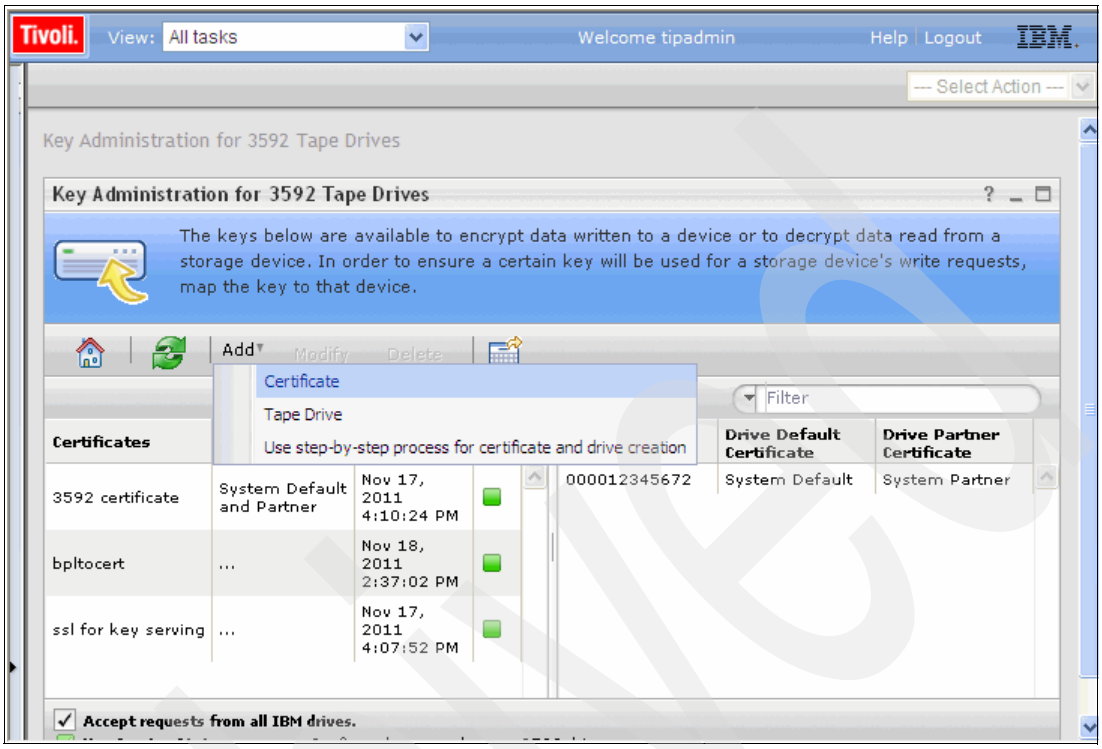


Figure 11-7 Adding a 3592 certificate

Select **3592 Certificate Rollover** as in Figure 11-8.

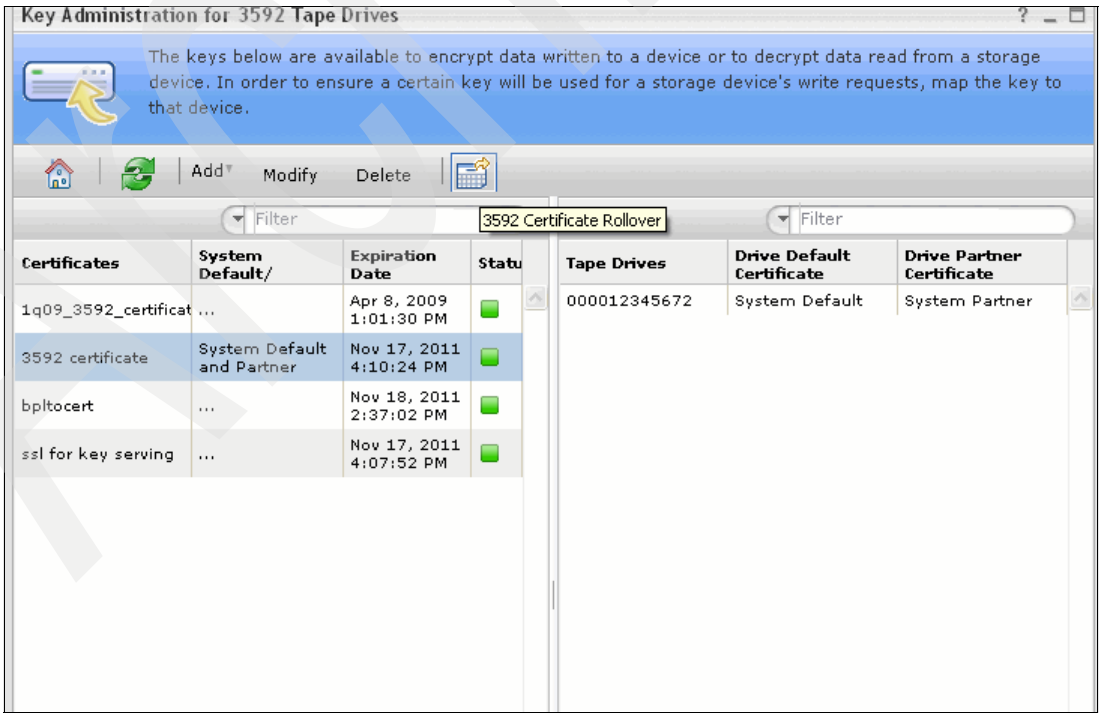


Figure 11-8 Select the Calendar icon to schedule a certificate change

Select **Add**. Refer to Figure 11-9.

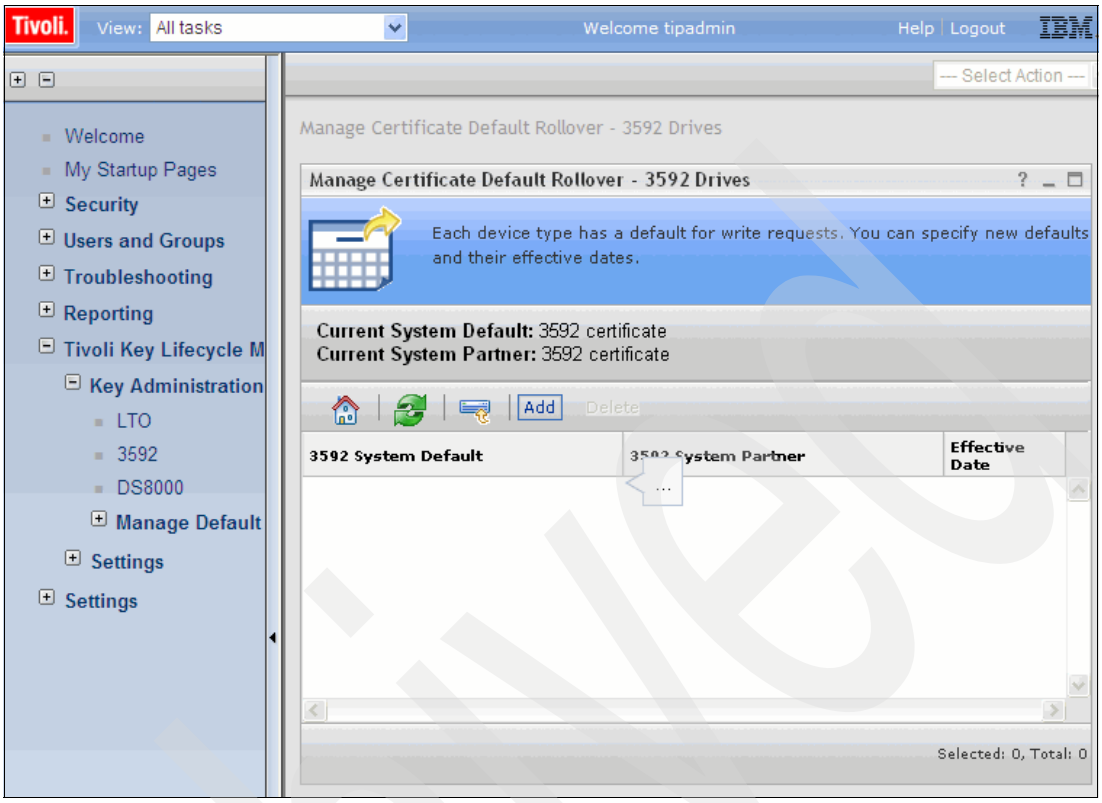


Figure 11-9 Certificate rollover schedule

Select the new certificate you want to use and when it should take affect. Refer to Figure 11-10 on page 361. Note its expiration date to ensure it will be usable for encrypting cartridges as long as you need it.

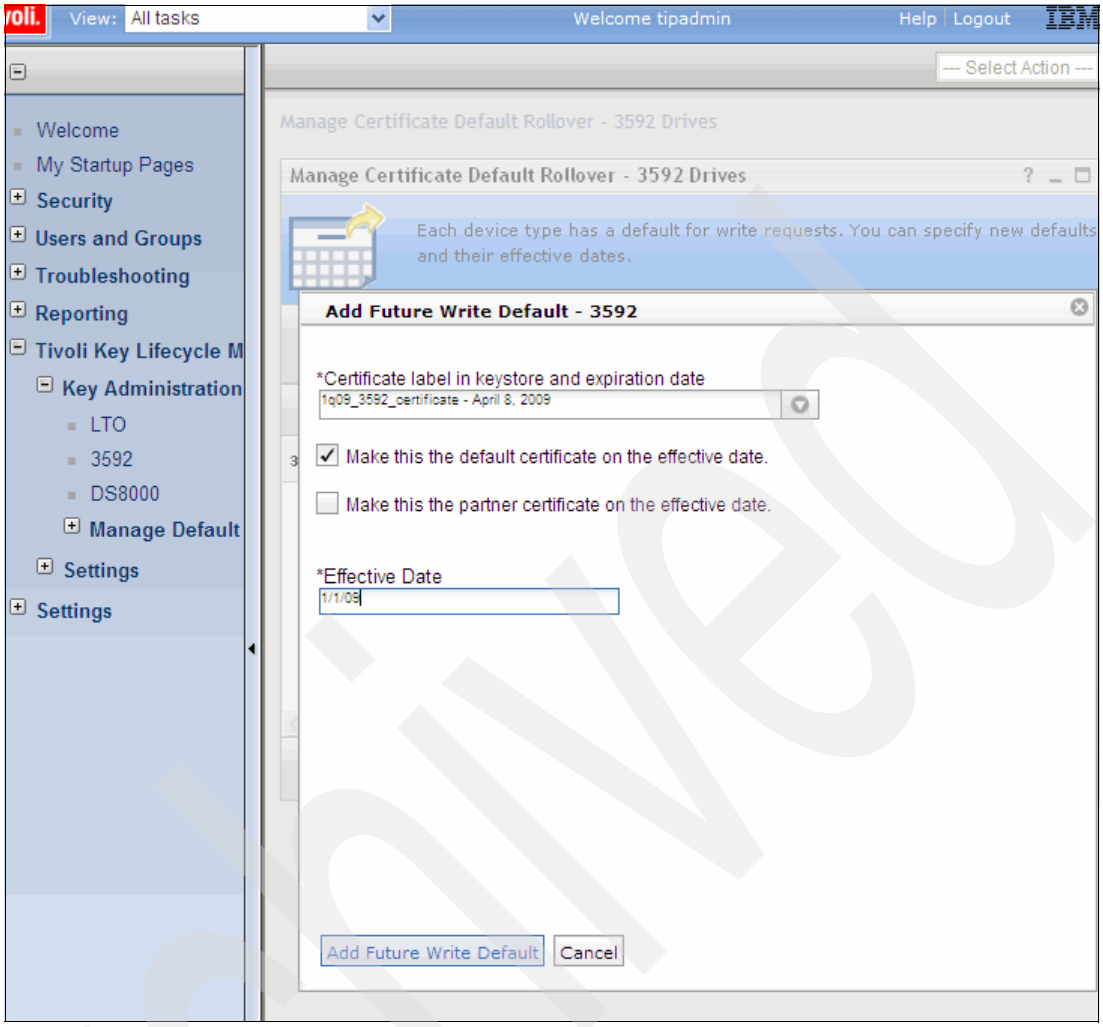


Figure 11-10 Select the certificate and its effective date

Make sure you backup and replicate this change to any secondary TKLM servers.

11.4 TKLM backup and restore procedures

The TKLM backup saves a password-protected copy of the TKLM server settings including the keystore and DB2 tables. However, when restoring, the function assumes that the environment is similar. TKLM restore-operations should be on the same platform with the same system, user account information and TKLM, and middleware file layout.

Because the keystore is backed up with the TKLM instance, you should treat the backups with the same logical and physical security controls that you do for the TKLM keystore.

Note: Backup and restore are disruptive to the TKLM server as of TKLM 1.0.

11.4.1 Backup by using the GUI

Using the GUI to backup the TKLM configuration is fairly simple but does require discipline by the administrator to perform a backup after significant changes and on a periodic interval. Before starting the backup, the administrator should plan for either a small service outage and confirm that one or more secondary key servers are running.

To backup using the GUI:

1. Select **Tivoli Key Lifecycle Manager** → **Settings** → **Backup and Restore**.
2. Click **Create Backup**. Refer to Figure 11-11.

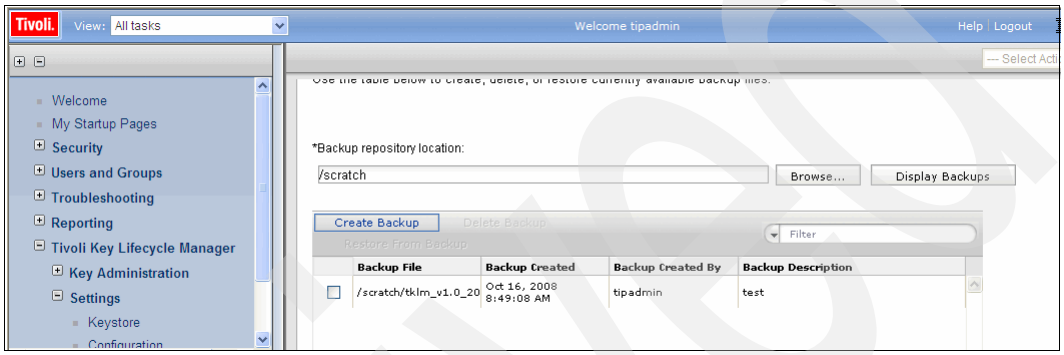


Figure 11-11 TKLM backup/restore

The panel in Figure 11-12 is displayed. You are prompted to select a location for the backup, a password for the backup and a short description.

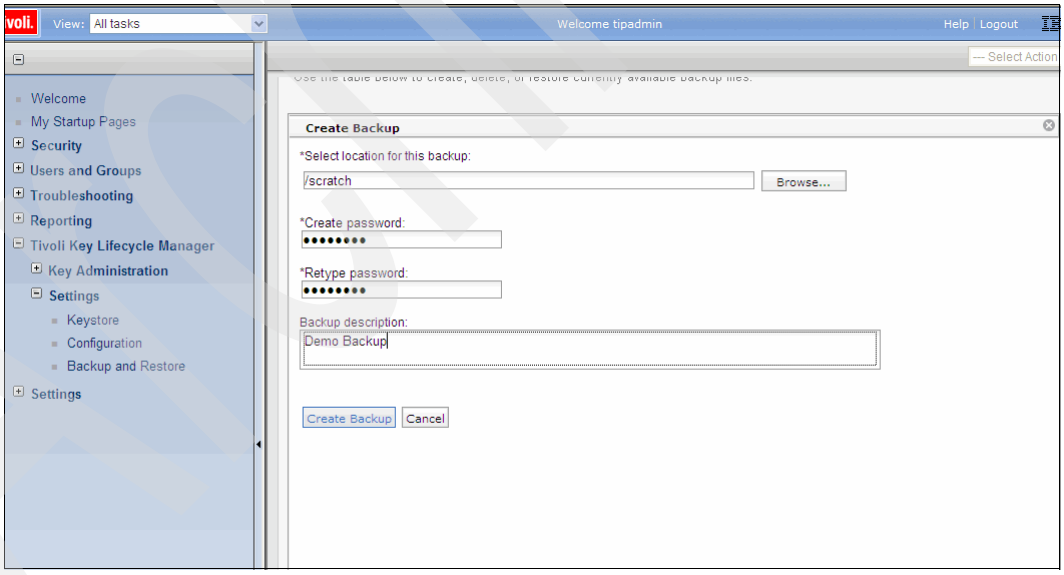


Figure 11-12 Backup creation

3. Click **Create Backup**. A warning message indicates that the backup will be stopping the server. In our environment, this was a short outage of a couple of minutes.

11.4.2 Restore by using the GUI

Using the GUI to restore the TKLM configuration is fairly simple but does require discipline by the administrator to perform a backup after significant changes and on a periodic interval.

Note: Restoring a backup requires command-line access to restart the server after the file has been restored.

To restore by using the GUI:

1. Select **Tivoli Key Lifecycle Manager** → **Settings** → **Backup and Restore**. Refer to Figure 11-13.

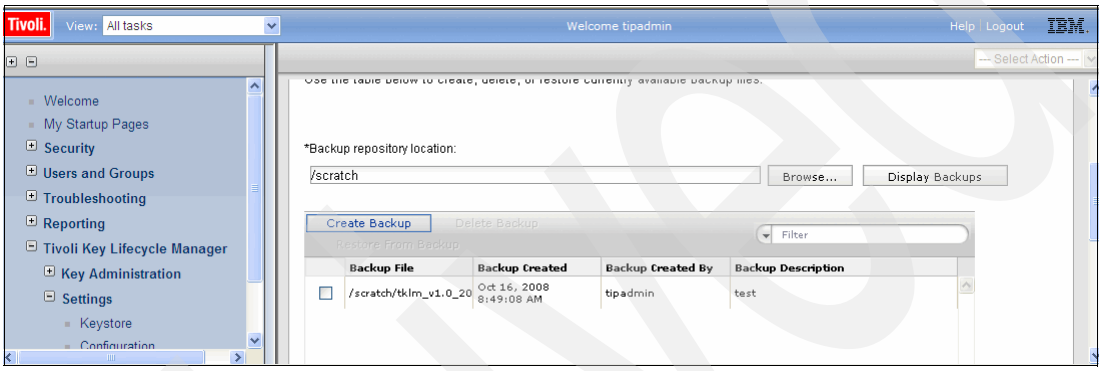


Figure 11-13 TKLM Backup/Restore

2. From the list of Backup Files, select the file that you want to restore and click **Restore From Backup** as shown in Figure 11-14.

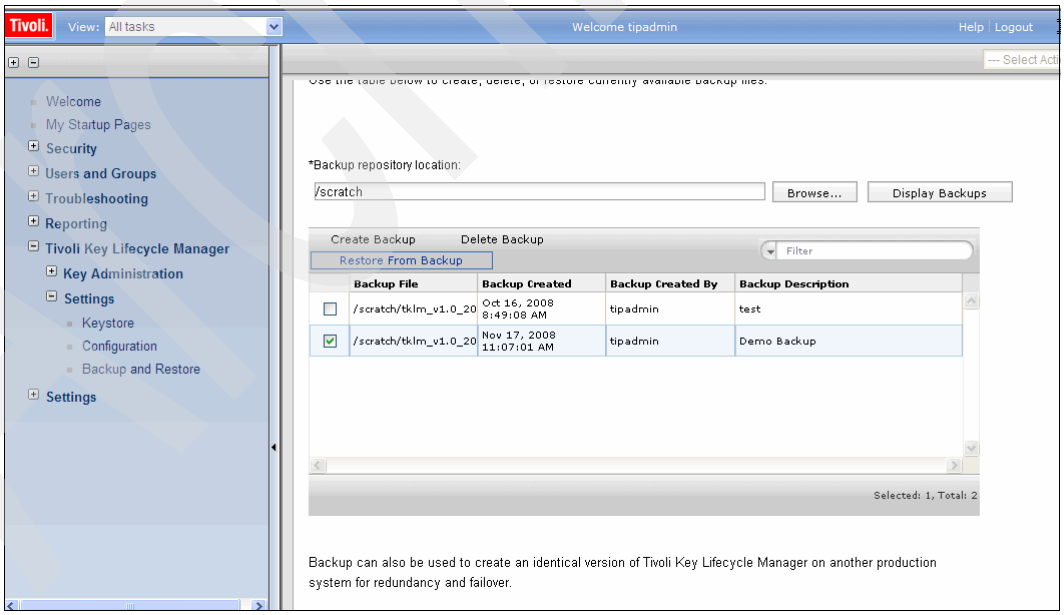


Figure 11-14 Select the backup file to restore

3. Enter the password for the backup file, as shown in Figure 11-15.

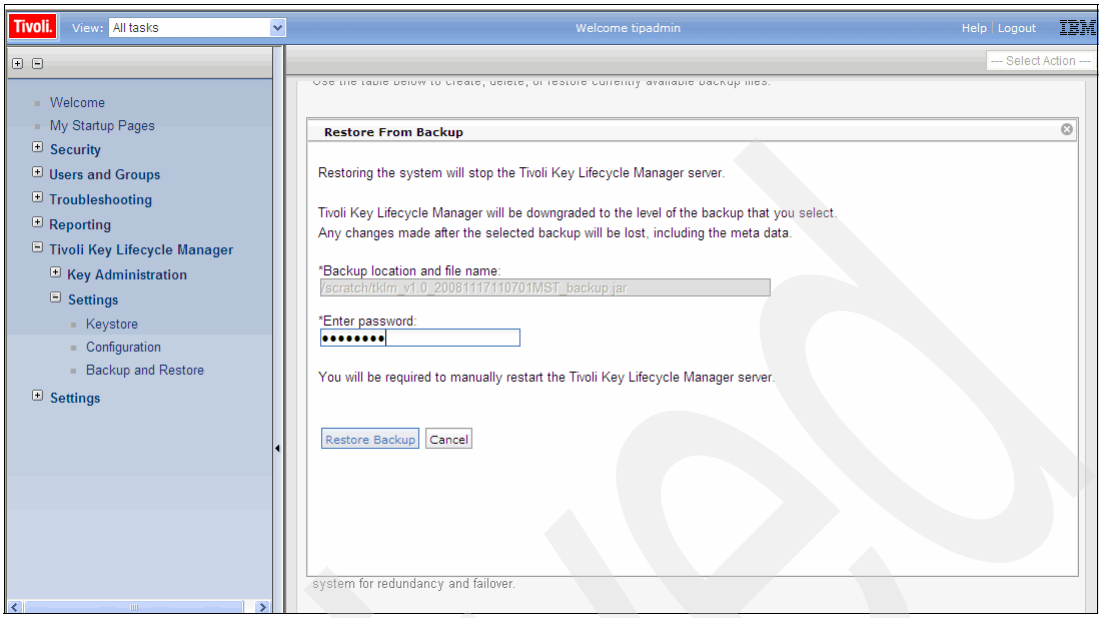


Figure 11-15 Enter the password

4. Confirm that it is OK to stop the TKLM server and overwrite the current configuration.
5. Restart TKLM, by opening a command line and then:
   - If on Windows, you must be an administrator or equivalent user. Perform these steps:
     i. From the `c:\IBM\tivoli\tip\bin` directory run **`stopServer.bat server1`**.
     ii. Enter the tipadmin user name and password.
     iii. Run **`startServer.bat`**.
     iv. Enter the tipadmin user name and password.
   - If on Linux, AIX, or Solaris, you must be root or equivalent privileged user that can start and stop the TKLM services. Perform these steps (seeExample 11-6 on page 365):
     i. From the `/opt/IBM/tivoli/tip/bin` directory, run **`stopServer.sh server1`**.
     i. Enter the tipadmin user name and password.
     ii. Run **`startServer.sh`**.
     iii. Enter the tipadmin surname and password.

Example 11-6 Starting and stopping the TKLM server on Linux

```
[root@dyn9011169152 bin]# ./stopServer.sh server1
ADMU0116I: Tool information is being logged in file

/opt/IBM/tivoli/tip/profiles/TIPProfile/logs/server1/stopServer.log
ADMU0128I: Starting tool with the TIPProfile profile
ADMU3100I: Reading configuration for server: server1
Realm/Cell Name: <default>
Username: tipadmin
Password:
 ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

[root@dyn9011169152 bin]# ./startServer.sh server1
ADMU0116I: Tool information is being logged in file

/opt/IBM/tivoli/tip/profiles/TIPProfile/logs/server1/startServer.log
ADMU0128I: Starting tool with the TIPProfile profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 1955
[root@dyn9011169152 bin]#
```

11.4.3 Backup by using the command line

Backing up by using the CLI is similar to using the GUI only less forgiving. Because of the startup time for the Jython interface, backing up by using the GUI is probably as fast and easier, unless you have integrated it into other scripts. Refer to 11.1, “Scripting with TKLM” on page 350 for an example of a quick backup script.

Note: TKLM Backup is a disruptive operation so ensure that you only perform backups during maintenance windows or when you have a secondary TKLM up and serving keys.

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default `c:\IBM\tivoli\tip\bin` directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default `/opt/IBM/tivoli/tip/bin/` directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Running the backup from the CLI

When you are at the `wsadmin` prompt, you can take a backup by using the command:

`print AdminTask.tklmBackupRun('[backupDirectory` *<the directory to save the backup in>* `-password` *<password to use on the backup file>*`]')`

After this completes you should have something similar to Example 11-7 on page 366, from a linux TKLM server. This example creates a backup with no description and a password of

myBackupPws in the /root/TKLMBackup directory. For a Windows backup, simply enter a valid Windows path name such as c:\TKLMBackup.

Example 11-7 Taking a backup from the CLI

```
[root@dyn9011169152 ~]# /opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin
-password password -lang jython
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>print AdminTask.tklmBackupRun('[-backupDirectory /root/TKLMBackup
-password myBackupPws]')
(0) Backup operation succeeded.
================================


wsadmin>exit
[root@dyn9011169152 ~]#
```

11.4.4 Restore by using the command line

Restore using the Command Line is similar to using the command line from the GUI with the advantage you are already at a command prompt so it is more natural to restart the server. You could even script up the restore operation so it includes starting and stopping the server.

Note: TKLM restore is a disruptive operation ensure you only perform restore's during maintenance windows or when you have a primary TKLM up and serving keys.

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default c:\IBM\tivoli\tip\bin directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default /opt/IBM/tivoli/tip/bin/ directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Running the restore from the CLI

When you are at the wsadmin prompt, you can restore a backup by using the command:

print AdminTask.tklmBackupRunRestore('[-backupFilePath *<backup file inc file name>* -password *<backup file password>*]')

After this completes you should have something similar to Example 11-8 on page 367, from a linux TKLM server. This example restores a backup file with a password of myBackupPws with the full file name. For a Windows backup, simply enter a valid windows path name like c:\TKLMBackup\<backup filename>.jar. The example then restarts the TKLM server, which is a required step after performing a restore operation.

Example 11-8 Restoring TKLM from a backup file

```
[root@dyn9011169152 ~]# /opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin
-password password -lang jython
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>print AdminTask.tklmBackupRunRestore('[-backupFilePath
/root/TKLMBackup/tklm_v1.0_20081117141514MST_backup.jar -password myBackupPws]')
(0) Restore operation succeeded. Restart the server.
================================================


wsadmin>exit
[root@dyn9011169152 ~]# /opt/IBM/tivoli/tip/bin/stopServer.sh server1ADMU0116I:
Tool information is being logged in file
           /opt/IBM/tivoli/tip/profiles/TIPProfile/logs/server1/stopServer.log
ADMU0128I: Starting tool with the TIPProfile profile
ADMU3100I: Reading configuration for server: server1
Realm/Cell Name: <default>
Username: tipadmin
Password:
 ADMU3201I: Server stop request issued. Waiting for stop status.
ADMU4000I: Server server1 stop completed.

[root@dyn9011169152 ~]# /opt/IBM/tivoli/tip/bin/startServer.sh server1ADMU0116I:
Tool information is being logged in file
           /opt/IBM/tivoli/tip/profiles/TIPProfile/logs/server1/startServer.log
ADMU0128I: Starting tool with the TIPProfile profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 5978
[root@dyn9011169152 ~]#
```

11.5 Data sharing with business partners

At some time, you will probably want to send an encrypted tape to a business partner. As with EKM, you will have to define keys or sets of keys that can be read by you and your business partner. At this time, you may only import and export keys from TKLM using the TKLM command line.

11.5.1 Sharing TS1100 certificate data with a business partner

TKLM can export or import certificates in two formats, which are base64 and Distinguished Encoding Rules (DER). The default used in our example is base64.

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default `c:\IBM\tivoli\tip\bin` directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default /opt/IBM/tivoli/tip/bin/ directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Listing the current certificates in the keystore

Assuming you have started the CLI you can then list the keys in the keystore by using the **tklmCertList** command as shown in Example 11-9. To export a certificate you must first determine its UUID. Find the key alias by using the GUI and then list out the key by name to find its UUID.

Example 11-9 Sample key store list usage

```
/opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin -password password -lang
jython
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>print AdminTask.tklmCertList()
CTGKM0001I Command succeeded.

uuid = CERTIFICATE-c68fb4ab-55c2-4599-8362-824d5fc45858
alias(es) = ssl for key serving
key store name(s) = Tivoli Key Lifecycle Manager Keystore
key state = active
issuer name = CN=SSL for Key Serving, OU=, O=, L=, ST=, C=
subject name = CN=SSL for Key Serving, OU=, O=, L=, ST=, C=
creation date = Nov 17, 2008
expiration date = Nov 17, 2011
serial number = 1226963273423369000

uuid = CERTIFICATE-71ee0c59-676d-4d71-ab22-3f7dfeee5af0
alias(es) = 3592 certificate
key store name(s) = Tivoli Key Lifecycle Manager Keystore
key state = active
issuer name = CN=Certificate for 3592, OU=, O=, L=, ST=, C=
subject name = CN=Certificate for 3592, OU=, O=, L=, ST=, C=
creation date = Nov 17, 2008
expiration date = Nov 17, 2011
serial number = 1226963425384785000

wsadmin>
```

If you only want a particular key, you have to use the `alias(es)` and `key store name(es)` parameters as shown in Example 11-10.

Example 11-10 keystore list using -alias

```
wsadmin>print AdminTask.tklmCertList('[-alias "3592 certificate" -keyStoreName
"Tivoli Key Lifecycle Manager Keystore"]')
CTGKM0001I Command succeeded.

uuid = CERTIFICATE-71ee0c59-676d-4d71-ab22-3f7dfeee5af0
alias(es) = 3592 certificate
key store name(s) = Tivoli Key Lifecycle Manager Keystore
key state = active
issuer name = CN=Certificate for 3592, OU=, O=, L=, ST=, C=
subject name = CN=Certificate for 3592, OU=, O=, L=, ST=, C=
creation date = Nov 17, 2008
expiration date = Nov 17, 2011
serial number = 1226963425384785000

wsadmin>
```

TS1100 encrypted media certificate export

Now that we know the certificate UUID as shown in Example 11-10, we can export the key by using the **tklmCertExport** command, as shown in Example 11-11.

Example 11-11 Using tklmCertExport to export certificates from TKLM

```
wsadmin>print AdminTask.tklmCertExport('[-uuid
CERTIFICATE-71ee0c59-676d-4d71-ab22-3f7dfeee5af0 -fileName /root/3592certificate]')
CTGKM0001I Command succeeded.
/root/3592certificate
wsadmin>exit
[root@dyn9011169152 ~]# cat /root/3592certificate
-----BEGIN CERTIFICATE-----
MIICSjCCAbOgAwIBAgIIEQcMvBJO5GgwDQYJKoZIhvcNAQEFBQAwVjEJMAcGA1UEBhMAMQkwBwYD
VQQIEwAxCTAHBgNVBAcTADEJMAcGA1UEChMAMQkwBwYDVQQLEwAxHTAbBgNVBAMTFENlcnRpZmlj
YXRlIGZvciAzNTkyMB4XDTA4MTExNzIzMTAyNFoXDTExMTExNzIzMTAyNFowVjEJMAcGA1UEBhMA
MQkwBwYDVQQIEwAxCTAHBgNVBAcTADEJMAcGA1UEChMAMQkwBwYDVQQLEwAxHTAbBgNVBAMTFENl
cnRpZmljYXRlIGZvciAzNTkyMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCXnKuPspW7ErCJ
heYLEgU/VGI1qfOMN22NXgkgsO3DIROBzqvAWgnYhBFoQraRhdZYK4Vu55XkIkzad7jVJ3yL771W
CXzRYHvLUmISIEpTGD4QBDSMFxF3JcRQrBUYQHuWkzqWn2sLbViEF+3NQvtqrP/8PTAuGS+rhhbt
n5EgyQIDAQABoyEwHzAdBgNVHQ4EFgQUYH89NQCw/zxWUVsqylfO6skOKOMwDQYJKoZIhvcNAQEF
BQADgYEAWrj8YX5zOAbfVAi1CmhVfxEcb3eeyXfh5b/7AGZyOH+xtxLJdt8Ro3H66k52uI7kRQf8
jCixfpbal8ITZHey6WH43WHl+gnSJIhq8CLugbRGaWcwuj9xSORdYHvloxhKBiwPVMLsrTGCsJCh
Yv7GiSHs9UehS58N7savmSQqaXo=
-----END CERTIFICATE-----
[root@dyn9011169152 ~]#
```

You can then send this certificate containing your public key to a business partner so they can write encrypted tapes that you can read using the private key stored in TKLM.

TS1100 encrypted media certificate import

Your business partner must send you a certificate containing the public key that the business partner would like to use to encrypt the TS1100 cartridges. After you receive the key, you may import the certificate as shown in Example 11-13 on page 370. The parameters usage is as follows:

```
print AdminTask.tklmCertImport('[-fileName <full path to certificate> -alias <the name of
the certificate in tklm> -format <base64 or DER> -keyStoreName "Tivoli Key Lifecycle
Manager Keystore" -usage <3592>]')
```

Example 11-12 Importing a certificate from a business partner

```
wsadmin>print AdminTask.tklmCertImport('[-fileName /root/bpCert.cer -alias
bp3592Cert -format base64 -keyStoreName "Tivoli Key Lifecycle Manager Keystore"
-usage 3592]')
CTGKM0001I Command succeeded.
wsadmin>
```

11.5.2 Sharing LTO key data with a business partner

TKLM can share keys with business partners by using TKLM by exporting the symeteric or secret keys. These keys are then imported into the business partner's key store and may be used to read or write tapes written with the same key or keys.

Note: With TKLM 1.0 you can only import LTO key data from TKLM

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default `c:\IBM\tivoli\tip\bin` directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default `/opt/IBM/tivoli/tip/bin/` directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

LTO encrypted media key group export

Before you can export keys to a business partner you must first have (import), from the business partner, a public key to use for encrypting the LTO keys, which will allow secure transmission of the LTO key (export).

Importing a business partner's public key

Your business partner must send you a certificate containing the public key that the business partner would like to use to encrypt the LTO keys. After you receive the key, you may import the certificate as shown in Example 11-13. The parameters usage is:

```
print AdminTask.tklmCertImport('[-fileName <full path to certificate> -alias <the name of
the certificate in tklm> -format <base64 or DER> -keyStoreName "Tivoli Key Lifecycle
Manager Keystore" -usage <3592>]')
```

Example 11-13 Importing a certificate from a business partner

```
wsadmin>print AdminTask.tklmCertImport('[-fileName /root/bpCert.cer -alias
bpLTOCert -format base64 -keyStoreName "Tivoli Key Lifecycle Manager Keystore"
-usage 3592]')
CTGKM0001I Command succeeded.
wsadmin>
```

Viewing key aliases with the GUI

You may view a range of LTO keys when using the TKLM command line. LTO keys are referred to as secret or symmetric keys. For this example, we have created a key group named LTO1; all keys in this key group start with the three letters LTO. Using the GUI, you may change the view in the TKLM to show the key aliases in a key group, as shown in Figure 11-16.

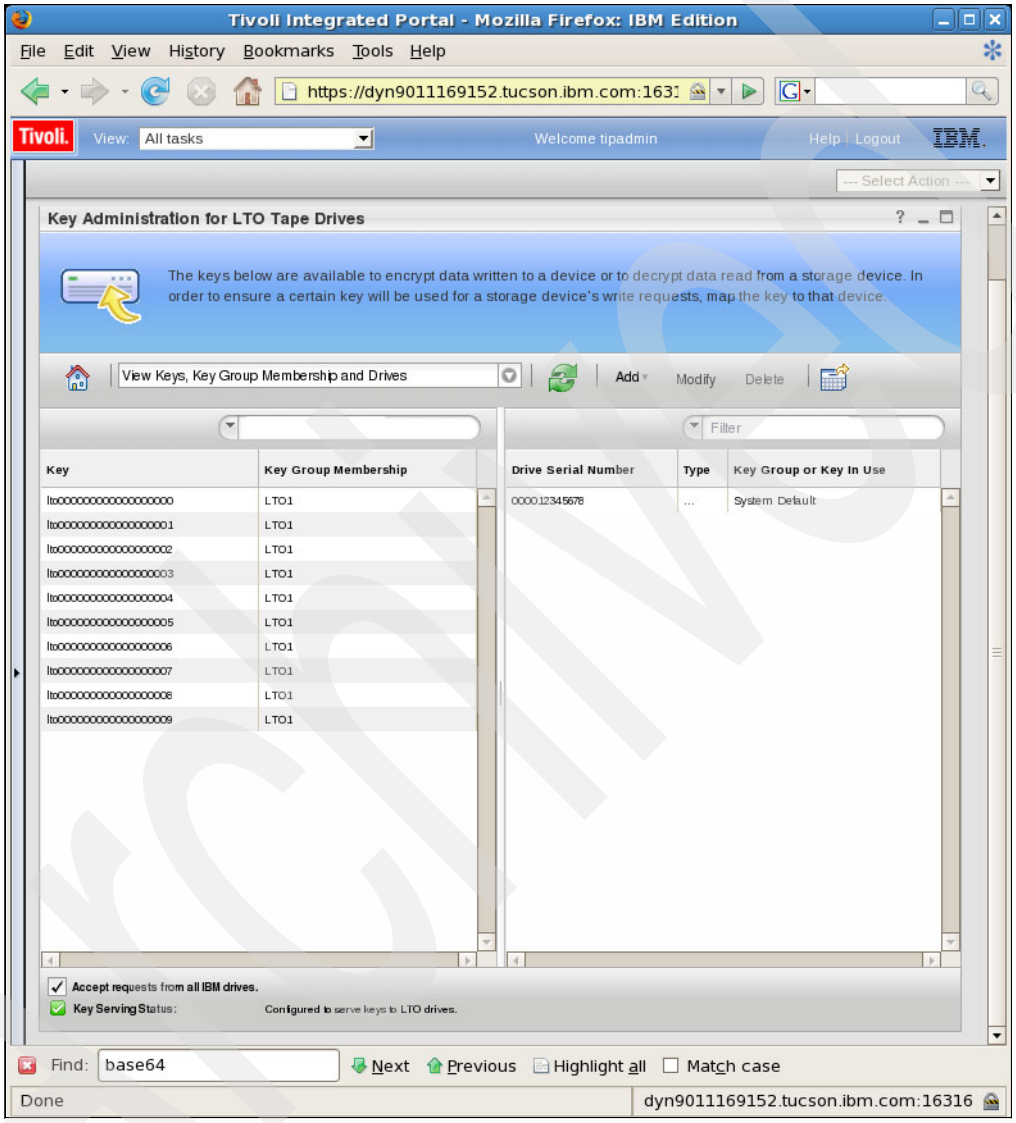


Figure 11-16 The key alias in a key group

Exporting LTO keys from the CLI

After you know the alias of the key groups you want to export, you may use the CLI to export them.

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default `c:\IBM\tivoli\tip\bin` directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default /opt/IBM/tivoli/tip/bin/ directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Exporting the LTO keys

In Example 11-14, we are exporting the keys LTO0000...0 to LTO0000...9 note that the command line allows you to ignore the leading 0s (zeros) so the range passed to the command line using the -aliasRange parameter is LTO0-9. The next option -fileName is the absolute path of the file we are exporting. You may use a relative path but then it is relative to the TKLM install directory. The keyStoreName parameter must be set to "Tivoli Key Lifecycle Manager Keystore". The -type option must be set to secretkey as this tells the export command we are exporting LTO keys the **tklmKeyExport** command line may also be used to export public-private key pairs from TKLM if necessary. The -keyAlias paramater specifies which certificate of the public key to use to encrypt the key file.

Example 11-14 Exporting LTO keys

```
[root@dyn9011169152 ~]# /opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin
-password password -lang jython
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
WASX7031I: For help, enter: "print Help.help()"
wsadmin>
wsadmin>print AdminTask.tklmKeyExport ('[-aliasRange LTO0-9 -fileName
/root/bpLTOKeys -keyStoreName "Tivoli Key Lifecycle Manager Keystore" -type
secretkey -keyAlias "3592 certificate"]')
CTGKM0001I Command succeeded.
```

LTO encrypted media key or keys import

To import a TKLM key file from another source you must first have the certificate containing the private key of the public-private key-pair used to encrypt the key file loaded into TKLM as described in 11.5.1, “Sharing TS1100 certificate data with a business partner” on page 367. To load the key file you have to use the CLI.

Starting the CLI on Windows

To start the Jython interactive command line as administrator or equivalent user, run the following command (assuming TKLM is installed in the default c:\IBM\tivoli\tip\bin directory):

```
wsadmin.bat -username TKLMAdmin -password password -lang jython
```

Starting the CLI on Linux, AIX, Solaris

To start the Jython interactive command line as root or equivalent user, run the following command (assuming TKLM is installed in the default /opt/IBM/tivoli/tip/bin/ directory):

```
wsadmin.sh -username TKLMAdmin -password password -lang jython
```

Loading the key file into TKLM

Example 11-15 on page 373 shows a script that loads a key file into TKLM. The importKeyFile.sh script calls the importKeyFile.py script, which prints descriptive text and then imports a file named /root/ltoKeyFile using the private key or certificate ltocert. The type, secretkey, is an LTO asymmetric key.

The usage parameter is a required option specifying that this is an LTO key file. For additional options you can use the CLI reference at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.tklm.doc/ref/ref_ic_cli_key_import.html

Example 11-15 Sample Jython script to add a key file

```
[root@dyn9011169152 ~]# cat importKeyFile.sh
/opt/IBM/tivoli/tip/bin/wsadmin.sh -username TKLMAdmin -password password -lang
jython -f importKeyFile.py
[root@dyn9011169152 ~]# cat importKeyFile.py
print "Importing key file /root/ltoKeyFile to TKLM with the same alias as used on
other TKLM"
print AdminTask.tklmKeyImport('[-fileName /root/ltoKeyFile -keyAlias ltocert
-keyStoreName "Tivoli Key Lifecycle Manager Keystore" -type secretkey -usage
LTO]')

[root@dyn9011169152 ~]#
```

Example 11-16 imports an LTO secret or asymmetric key group into TKLM.

Example 11-16 Key Import into TKLM

```
[root@dyn9011169152 ~]# ./importKeyFile.sh
WASX7209I: Connected to process "server1" on node TIPNode using SOAP connector;
The type of process is: UnManagedProcess
Importing key file /root/ltoKeyFile to TKLM with the same alias as used on other
TKLM
CTGKM0001I Command succeeded.
[root@dyn9011169152 ~]#
```

11.6 Removing TKLM

You might have to remove TKLM from a system. The current TKLM 1.0 uninstall process is not as straightforward as running the installer. The steps in this section clean the TKLM off a Windows system. Make sure you have another TKLM or EKM backup system configured and working, or that you are performing the TKLM cleanup during a maintenance cycle, because after the TKLM service is stopped you can no longer use this TKLM server to read or write tape cartridges.

11.6.1 Backing up the keystore

Ensure that you have a good backup copy of the keystore before starting this process because the process removes the currently running keystore. For the procedures to back up your TKLM configuration and keystore, see 11.4, “TKLM backup and restore procedures” on page 361.

11.6.2 Removing TKLM from a Windows system

The general steps include stopping and removing the TIP service, uninstalling DB2, and delete TKLM from the system.

To remove TKLM from a Windows system:

1. Stop the TIP service:

   a. Open the services window and find “Tivoli Integrated Portal - TipProfile_Port_16310. See Figure 11-17.

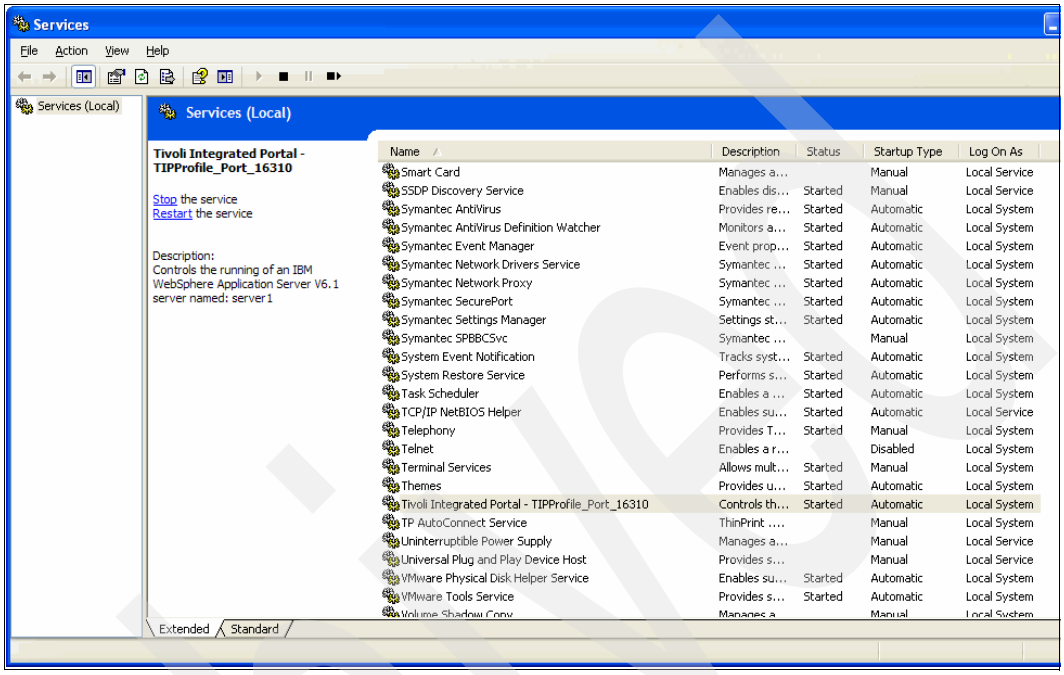


Figure 11-17 Stopping the TIP service

   b. If the service does not stop you can stop it from the command line. In Example 11-17, the TIP service had been installed in `c:\ibm\tivoli\tip\` directory.

Example 11-17 Stopping the TIP service

```
C:\ibm\tivoli\tip\bin>WASService -stop TIPProfile_Port_16310
Could not stop service: The service has not been started.

C:\ibm\tivoli\tip\bin>
```

2. Remove the TKLM DB2 instance owner. Refer to Example 11-18.

Example 11-18 Uninstall TKLM DB2 instance owner

```
C:\ibm\tivoli\tip\products\tklm\_uninst>removeDB2Inst.bat
You will lose the database instance and all data in it when you run this batch
file, are you sure you want to continue? (yes/no)     yes
DB2 instance tklmdb2
DB2 name tklmdb
DB2 install directory  C:\Program Files\IBM\SQLLIB\
Removing database instance...
Uninstall DB2 instance is complete. You can look at
\tklmtemp\removeDB2Inst.output.txt for more information
C:\ibm\tivoli\tip\products\tklm\_uninst>
```

3. Uninstall TIP using. Refer to Example 11-19 and Figure 11-18.

Example 11-19 Launch the graphical uninstaller

```
C:\ibm\tivoli\tip\_uninst\TIPInstall>uninstall.exe
```

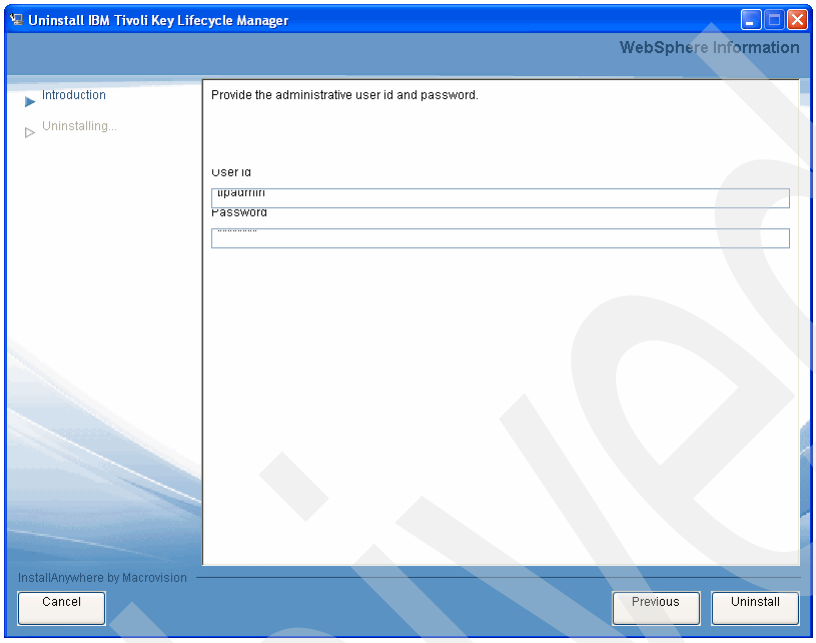


Figure 11-18 Enter your TIP admin user name and password

Uninstalling can take several minutes.

- If uninstalling fails:
  i. Run the `removeDEInfo.bat` file, in the `products\tklm\_uninst` directory.
  ii. From the bin directory, run `WASService -remove TIPProfile_Port_<port_number>`.

4. Remove the TIP home directory. Refer to Example 11-20.

Example 11-20 Cleanup other directories

```
C:\ibm\tivoli>rmdir /s tip
tip, Are you sure (Y/N)? y

C:\ibm\tivoli>

Directory of C:\Documents and Settings\VRes01

11/13/2008  12:05 PM    <DIR>          .
11/13/2008  12:05 PM    <DIR>          ..
11/13/2008  10:06 AM    <DIR>          Desktop
08/08/2007  06:36 AM    <DIR>          Favorites
11/12/2008  12:57 PM           197,634 IA-TIPInstall-00.log
11/13/2008  12:05 PM            97,336 IA-TIPUninstall-00.log
10/29/2008  08:49 AM    <DIR>          My Documents
10/29/2008  08:56 AM    <DIR>          Start Menu
               2 File(s)        294,970 bytes
               6 Dir(s)   9,682,993,152 bytes free
```

```
C:\Documents and Settings\VResO1>del IA*.log

C:\Documents and Settings\VResO1> rmdir /S c:\tklmtemp
c:\tklmtemp, Are you sure (Y/N)? y

C:\Documents and Settings\VResO1>del c:\tklm_install.std*

C:\Program Files\IBM>"\Program Files\IBM\Common\acsi\setenv.cmd"

C:\Program Files\IBM>"\Program Files\IBM\Common\acsi\bin\si_inst.bat" -r -f
ACUINIO004I UnInstallation completed successfully!
The batch file cannot be found.

C:\Program Files\IBM>rmdir /s "c:\Program Files\IBM\Common\acsi"
c:\Program Files\IBM\Common\acsi, Are you sure (Y/N)? y
```

5. Verify that the acsi directory has actually been removed.

- Uninstall DB2. Refer to Figure 11-19.

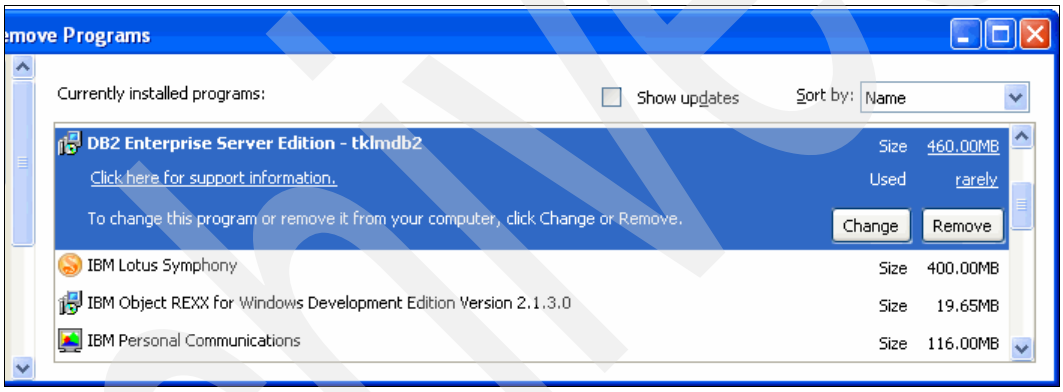


Figure 11-19 Remove DB2

6. Stop and disable the DB2 services (refer to Figure 11-20).
   - DB2(tklmx)
   - DB2 Governor (tklmx)
   - DB2 License Server (tkmx)
   - DB2 Management Service (tklmx)
   - DB2 Remote Command Server (tklmx)
   - DB2 Security Server (tklmx)
   - DB2DAS

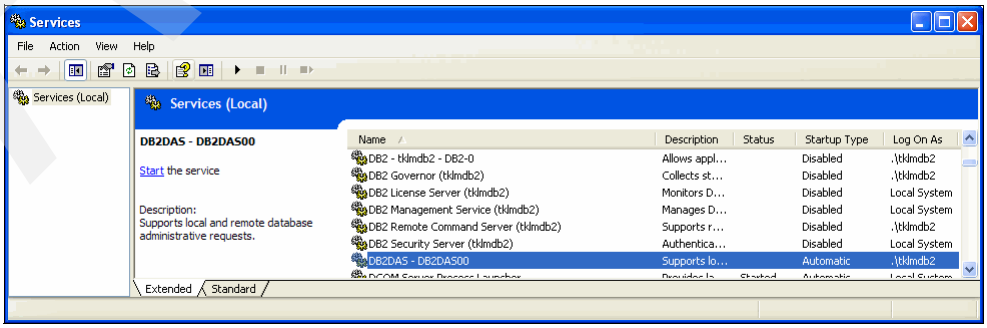


Figure 11-20 TKLM DB2 services to stop and disable

7. Delete the TKLM Windows user. Refer to Figure 11-21.

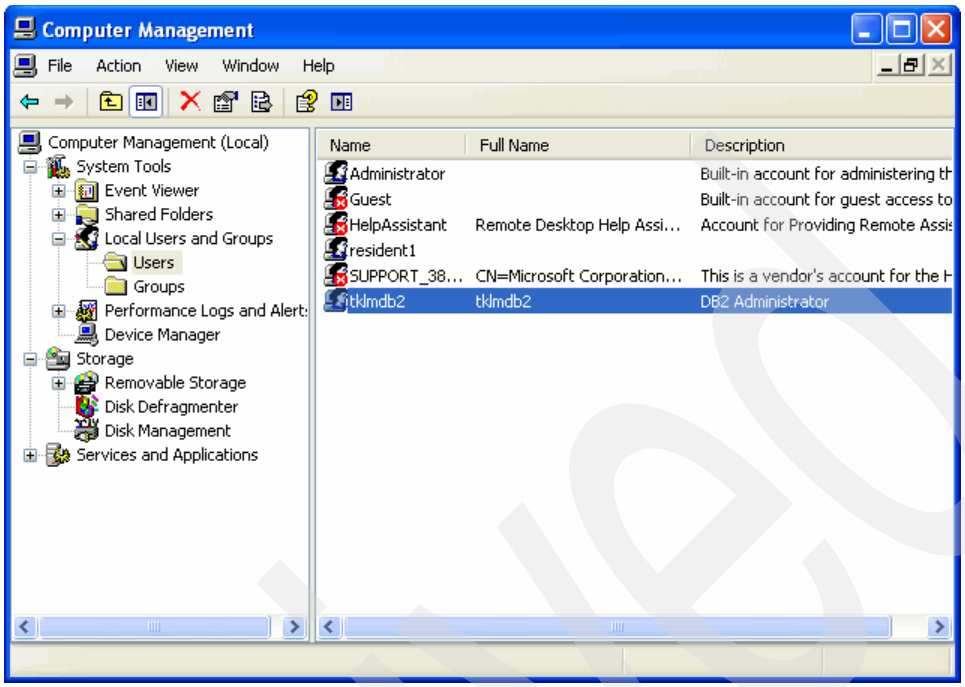


Figure 11-21 Delete the TKLM windows user

8. Restart the computer

11.7 Fixing the security warnings in your Web browser

Internet Explorer and Firefox both raise security warnings about the TKLM certificate. This action is normal because the certificate that is installed is an unsigned certificate. If you want to stop the warnings, following the steps in this section.

11.7.1 Fixing the security warning in Internet Explorer browser

If you receive the error shown in Figure 11-22, select **Continue** if you are sure that you have the correct IP for your TKLM server and you have not previously installed the certificate for this server.

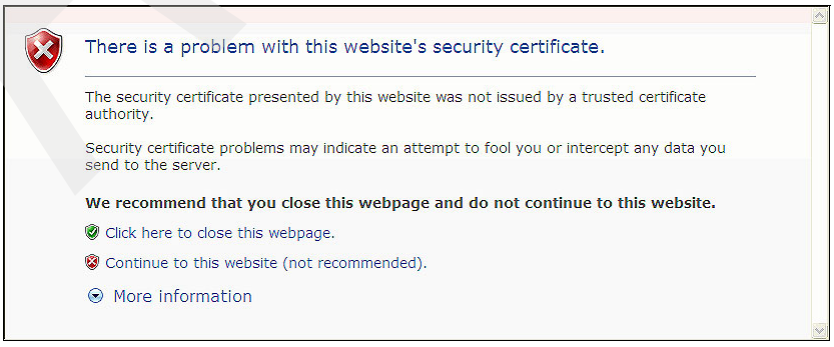


Figure 11-22 Warning message in Internet Explorer browser window

Click the **Certificate Error** and select view certificates. Refer to Figure 11-22 on page 377.

Figure 11-23 IE Error

Then, select **Install certificate** and follow the prompts to install the TKLM server certificate. After you have added the certificate, restart Internet Explorer.

Note: You must connect to TKLM using the host name in the certificate, not IP or other host names, to avoid certificate mismatch warning.

11.7.2 Fixing the security warning in Firefox 2

Firefox 2 is more forgiving. It presents an initial warning, which, if you accept, does not appear again. As of this writing, Firefox 3 is not supported and certain pages do not render correctly.

Part 4

Implementing tape data encryption

In this part, we explain the steps required to implement tape data encryption on the various host platforms, tape drives, and tape libraries.

12

Implementing TS1100 series Encryption in System z

In this chapter, we describe the required implementation steps for using the TS1120 and TS1130 Tape Encryption capability in z/OS or z/VM. For these platforms, the TS1120 and TS1130 drives must be attached to an 3592-J70 tape controller or an IBM TS1120 and TS1130 Model C06 Tape Controller. The TS1120 or TS1130 drives can reside in a TS3500 tape library, a 3494 tape library, or an IBM TS3400 Tape Library. The drives also can reside in a C20 Silo-compatible frame or a rack, although we do not discuss these devices here.

Note: This chapter does not discuss encryption on TS1120 drives when they are attached to a TS7700 Virtualization Engine. For this information, refer to Chapter 13, “Implementing TS7700 Tape Encryption” on page 421.

We discuss the following implementation topics:

- Implementation overview
- Tape library implementation
- z/OS implementation steps
- z/VM implementation steps
- Miscellaneous implementation considerations
- TS1120 Model E05 rekeying support in z/OS

12.1 Implementation overview

The implementation steps can include:

- The installation of the tape library, TS1120 and TS1130 tape drives, and tape controllers
- A firmware upgrade of the tape and library components and installation of the required software support

We will assume here that the drives, libraries, and controllers have already been installed and are operating without encryption. We describe what is required to implement encryption on the TS1120 tape drives.

All prerequisites necessary for hardware (microcode levels for the tape drives, libraries, and so forth) and software for the various platforms are discussed in Chapter 4, “Planning for software and hardware” on page 91.

Although we do not discuss TS1120 or TS1130 encryption implementation in detail for z/VSE and z/TPF operating systems, both of these operating systems use out-of-band encryption in a manner similar to z/VM. Use the procedures for z/VM to enable out-of-band encryption on the tape control units after the installation of feature code (FC) 5595 or FC9595 on the tape control units. For information about how to control and set encryption for z/VSE or z/TPF, we refer you to the following publications:

- For z/VSE, refer to *z/VSE V4R1 Administration*, SC33-8304.
- For z/TPF, refer to *z/TPF Enterprise Edition Operations*, GTPO-1MS5, at the IBM TPF Product Information Center:

  `http://publib.boulder.ibm.com/infocenter/tpfhelp/current/index.jsp?topic=/com.ibm.tpf.doc/pichHome.html`

12.2 Implementation prerequisites

First, you must have tape drives, tape controllers, and a tape library operating without encryption, which we review in 12.2.1, “Initial tape library hardware implementation” on page 383 and in 12.2.2, “Initial z/OS software definitions” on page 384.

To enable and use tape data encryption in an already installed tape library, additional setup steps are required. These implementation steps are described in detail in this chapter.

The prerequisites for encryption in System z are:

- Encryption Key Manager (EKM) can be implemented on any supported platform.

  The EKM must be connected to and communicating with the operating system (z/OS in-band) or the EKM must be connected to and communicating with the tape control units (z/VM, z/VSE, or z/TPF out-of-band). At least one certificate with a key label must be in the EKM keystore.

  We provide an overview in 12.3, “EKM implementation overview” on page 385. The complete setup steps for implementing EKM are described in Chapter 7, “Planning and managing your keys” on page 227.
- Hardware setup for tape data encryption prerequisites include:
  - An encryption-capable TS1120 tape drive attached to a TS1120 Model C06 Tape Controller or a 3592-J70 tape controller.

- Enable the encryption method for the TS1120 drives installed in the library. See 12.4, “Tape library implementation” on page 386. However, defer this action until the necessary operating system maintenance has been installed.
- Enablement of encryption on the tape controllers. However, defer this action until the necessary operating systems maintenance has been installed. See 12.5, “Tape control unit implementation” on page 394.

- z/OS tape data encryption setup prerequisites include:
  - z/OS software maintenance installed that supports an encryption-enabled TS1120 and TS1130. See 12.6.1, “z/OS software maintenance” on page 394.
  - Update the IECIOS*xx* PARMLIB member. See 12.6.2, “Update PARMLIB member IECIOSxx” on page 395.
  - z/OS basic DFSMS implementation that includes a DFSMS Data Class that specifies a Recording Technology of EE2 and specifies a key label (that exists in the EKM keystore) to use for encrypting. See 12.6.3, “Define or update Data Class definitions” on page 396.
  - Update the JES3 definitions. See 12.6.4, “Considerations for JES3” on page 400.
  - Verify and update your tape management system. See 12.6.5, “Tape management system” on page 401.
- z/VM tape data encryption setup is covered in section 12.7, “z/VM implementation steps” on page 407.

For detailed z/OS implementation checklists, refer to Appendix A, “z/OS planning and implementation checklists” on page 557.

Finally, we discuss other implementation items in 12.8, “Miscellaneous implementation considerations” on page 415 and TS1120 rekeying in 12.9, “TS1120 and TS1130 tape cartridge rekeying in z/OS” on page 416.

Now, you are ready to start creating encrypted cartridges.

12.2.1 Initial tape library hardware implementation

If you do not have an IBM system-managed tape library installed, refer to the following publications for a detailed discussion of all implementation and migration steps:

- For TS3500, refer to *IBM TS3500 for System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789.
- For IBM 3494, refer to *IBM TotalStorage 3494 Tape Library: A Practical Guide to Enterprise-Class Tape Drives and Tape Automation*, SG24-4632.
- For the TS3400, refer to *IBM System Storage TS3400 Tape Library Planning and Operator Guide*, GC27-2107.

An overview of the tape library implementation steps follows:

- Hardware installation

  The service support representative (SSR) installs the hardware, such as the tape library frames or components, the Library Manager, the tape drives, and the controllers. In parallel, you may start defining the new hardware to your environment.

  If you are installing encryption-capable TS1120 and TS1130 drives, the only additional installation step is to enable encryption on the TS1120 and TS1130 tape drives, but *not*

until all of the software maintenance to support encryption has been installed. We describe enabling encryption on the TS1120 tape drives in detail later.

- Library Manager setup

  z/OS or z/VM requires a Library Manager for the TS3500 and 3494 tape libraries. For the TS3500 library, the Library Manager or Managers are located in the 3953-F05 frame. For the 3494 library, the Library Manager is in the 3494-L*xx* frame and optionally, the 3494-HA1 frame.

  After the tape library and drives are installed, you have to create definitions on the hardware level. These definitions depend on which devices are installed inside the TS3500 or 3494 tape library, whether you are sharing the tape library between different platforms, whether an IBM Virtualization solution (TS7700 or IBM 3494 VTS) is included, and other environment specifics. In a z/OS environment, you perform the setup at the TS3500 and at the IBM 3953 Library Manager level.

  Setup steps for the *TS3500 tape library* include:

  – Enable Advanced Library Management System (ALMS). ALMS is required for attachment of the TS3500 with IBM 3953 to System z servers.
  – Enable the Insert Notification feature.
  – Define a logical library.
  – Add tape drives to the logical library.
  – Define cartridge slots and control path drives for the logical library.
  – Enable eight-character VOLSER support.
  – Define cartridge assignment policies (CAPs).

  At the *IBM 3953 Library Manager* or the 3494 Library Manager, setup steps include:

  – Define physical VOLSER ranges for native drives, TS7700 Virtualization Engine, and VTS, if installed.
  – Define TS7700 and VTS management policies and parameters.

  For *TS7700 Virtualization Engine* setup steps, refer to *IBM System Storage Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312.

- Hardware Configuration Definition

  The Hardware Configuration Definition (HCD) dialog is mainly related to hardware. You use the HCD panels to define the new hardware. All 3592 models, J1A and E05, emulate 3590 Model B tape drives from a software point of view, so the TS1120 is defined as a 3590 tape drive as well.

  Because you do not define whether an IBM TS1120 or TS1130 tape drive is encryption-enabled in HCD, both encryption-enabled and non-encryption-enabled TS1120 or TS1130 drives are defined exactly the same. You do not have to change existing HCD definitions when implementing tape data encryption on an already defined TS1120.

12.2.2 Initial z/OS software definitions

IBM tape libraries in a z/OS environment are managed through Data Facility Storage Management Subsystem (DFSMS). Therefore, the major part of implementing a tape library is defining the tape library to SMS and defining the SMS constructs and Automatic Class Selection (ACS) routines that direct tape allocation to the requested tape library.

Depending on your existing software environment, all or some of the following implementation steps are required for z/OS host software setup:

- Verify or update these SYS1.PARMLIB members:
  - DEVSUP*xx*
  - SCHED*xx*
  - IGDSMS*xx*
  - IEFSSN*xx*
  - CONSOL*xx*
  - COMMND*xx*
  - GRSCNF*xx*
  - LOAD*xx*
  - COFVLD*xx*
  - ALLOC*xx*
  - IECIOS*xx*
- Define security profiles.
- Allocate the Tape Configuration Database (TCDB).
- Prepare, start, and customize OAM, including Installation Exits.
- Update and customize your tape management system.
- Using the Interactive System Management Facility (ISMF) panels, define:
  - The TS3500, 3494, or TS3400 tape library

    Note: The TS3400 does not appear to the host as a library; the drives in the library appear as stand-alone drives. You only have to define the TS3400 to the host if you will be using the drives in the TS3400 with the Manual Tape Library (MTL) support.

  - One or more Data Class constructs
  - One or more Storage Class constructs
  - One or more Management Class constructs
  - One or more Storage Group constructs
- Update the ACS routines to assign the new constructs as required.
- Translate, validate, and test the ACS routines.
- Activate the SMS configuration.
- If you are using JES3, update the JES3 INISH deck.

12.3 EKM implementation overview

In a z/OS or z/VM environment, you must use System-Managed Encryption, which requires the Encryption Key Manager (EKM) as a prerequisite. In preparation for encryption on your TS1120 tape drives, you must implement EKM and have at least one key in its keystore. Refer to section 6.1, “Implementing the EKM in z/OS” on page 160, which describes the setup steps for implementing EKM.

The basic EKM implementation steps are:

1. Note that UNIX System Services as a prerequisite for Java is already included with the z/OS operating system. Verify whether the installed version of Java is at the appropriate level for the Encryption Key Manager component.
2. Install the EKM after downloading the newest versions available.

3. Obtain the required security permission for the UNIX System Services segment.
4. Create a keystore for EKM.
5. Set up the EKM environment on JAVA.
6. Start the EKM.
7. Specify the TCP/IP address of the EKM or EKMs through the IECIOS*xx* PARMLIB member or the SETIOS command.
8. Generate or import certificates and private keys into EKM's keystore.

12.4 Tape library implementation

TS1120 encryption on the z/OS platform can be implemented in the TS3500, the 3494, or the TS3400 tape library. You must specify the System-Managed Encryption method to enable encryption in the TS1120 or TS1130 drives. Next, we describe how to specify the System-Managed Encryption method for each of the libraries.

Note: Do *not* perform these implementation steps until you have installed all required operating system support for TS1120 and TS1130 tape drives with encryption.

If you are using out-of-band encryption (which is usually for z/VM, z/VSE, or z/TPF operating systems), you also have to update the EKM IP addresses to be used by the tape controllers. We discuss updating the EKM IP addresses in the z/VM section, because z/VM uses out-of-band encryption.

12.4.1 Implementation steps for the IBM TS3500 Tape Library

To update the tape drive definitions, use the Web browser interface of the TS3500, the TS3500 Tape Library Specialist. Figure 12-1 on page 387 shows the Welcome panel of the TS3500 Specialist.

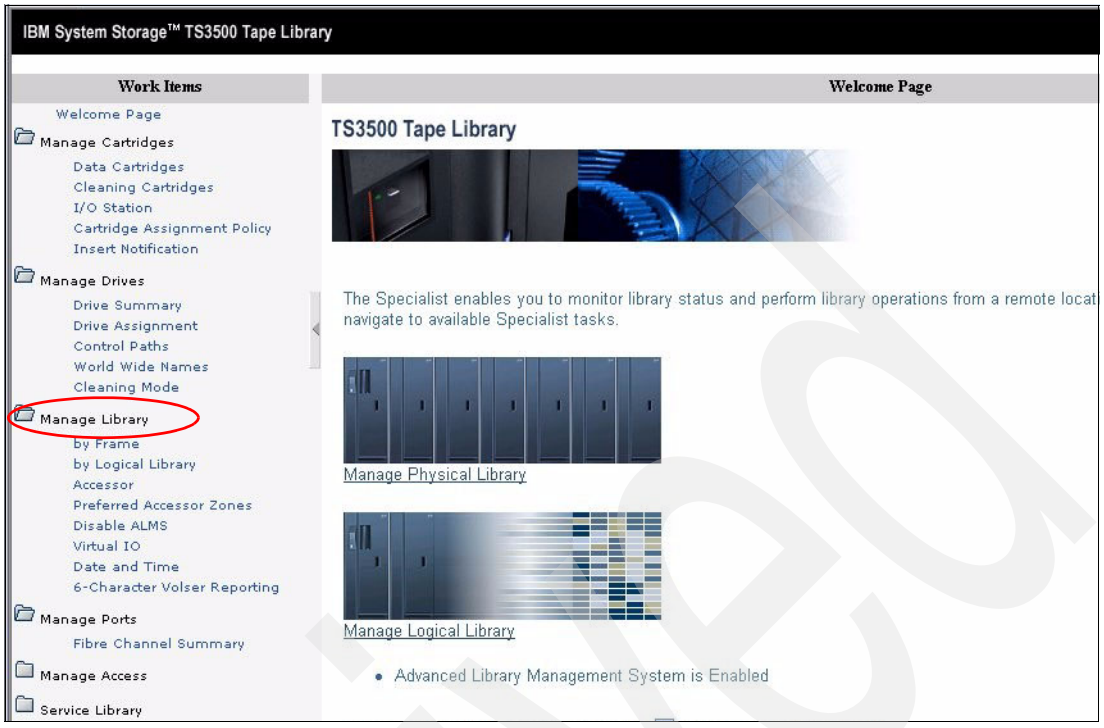


Figure 12-1 Main entry or welcome panel of TS3500 Tape System Library

To perform the implementation:

1. On the left side of the panel, select **Manage Library**. The Manage Logical Libraries panel opens. See Figure 12-2 on page 388.

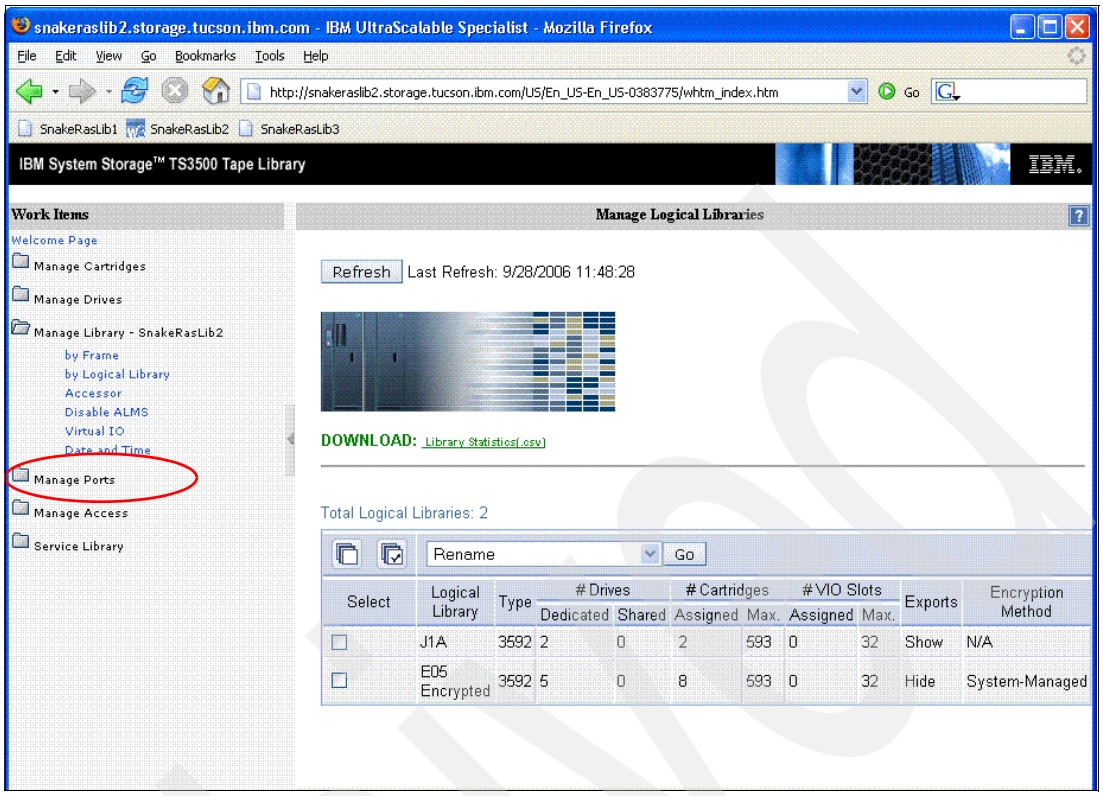


Figure 12-2 Manage Logical Libraries panel

2. Check the logical library that you want to update and then click **Manage Drives** on the left side of the panel. The panel shown in Figure 12-3 displays.

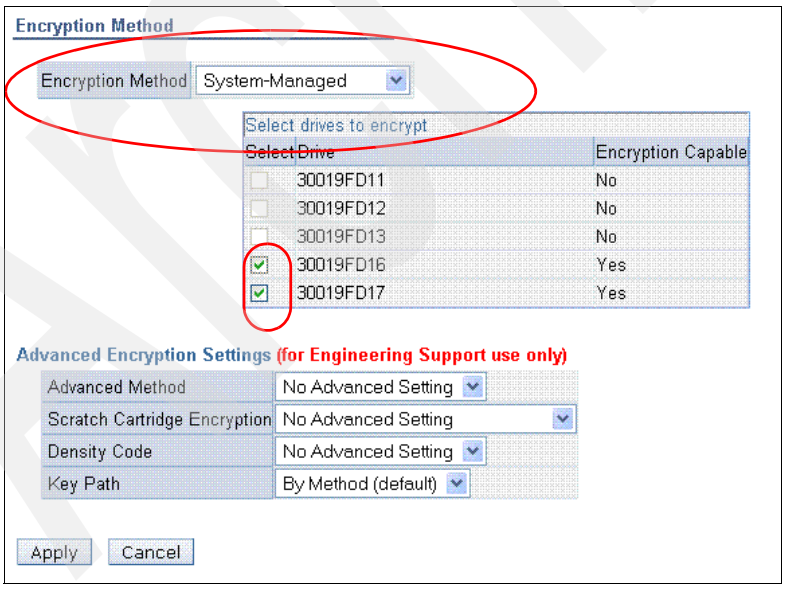


Figure 12-3 Encryption Method panel

3. In this panel, select the encryption method **System-Managed,** check the encryption-capable drives to be attached to z/OS, and click **Apply**.

Warning: All drives attached to the same control unit **must** be selected. As Figure 12-3 on page 388 shows that you can select the last two drives only if those drives are not connected to the same control unit.

12.4.2 Implementation steps for the IBM 3494 Tape Library

Note: If your 3494 Library Manager is pre-535 microcode level, you will not be able to perform these procedures. Instead, order FC5595 or FC9595 on the 3592-J70 or TS1120 and TS1130 Model C06 tape controllers. This item ships instructions and procedures for the IBM SSR to use for enabling encryption on the controller and all attached encryption-capable drives.

You can either use the Web browser interface of the 3494 tape library or the 3494 Library Manager user interface. We describe both methods in the following sections.

3494 Enterprise Automated Tape Library Specialist

To update the tape drive definitions, use the Web browser interface of the 3494, the 3494 Enterprise Automated Tape Library Specialist (ETL). Figure 12-4 shows you the Welcome panel of the Enterprise Automated Tape Specialist.

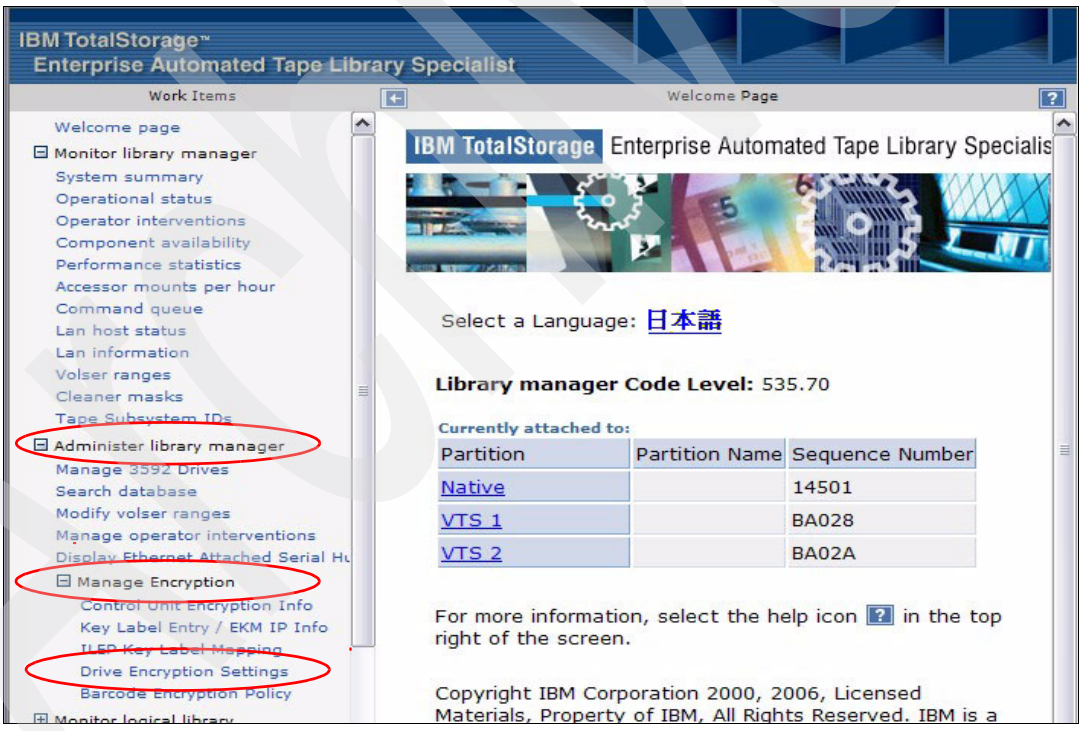


Figure 12-4 Main entry or welcome panel of the 3494 ETL

To perform the implementation:

1. On the left side of the panel, select **Administer library manager** → **Manage Encryption** → **Drive Encryption Settings**, which displays a panel similar to that in Figure 12-5 on page 390. We do not show the left pane of the window in Figure 12-5 on page 390 for better clarity. Figure 12-5 on page 390 shows a list of all encryption-capable drives in the 3494 library and how they are currently configured.

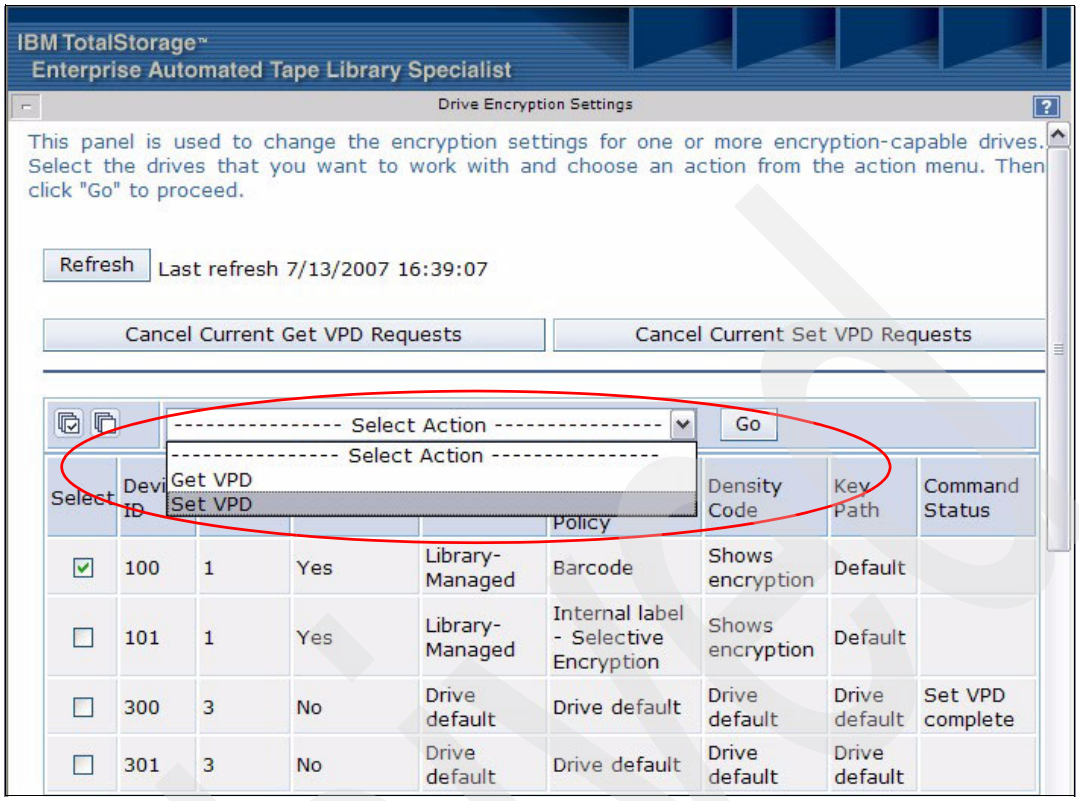


Figure 12-5 Drive Encryption Settings panel

2. In this panel, check the boxes for all of the encryption-capable drives to be attached to z/OS, from the Select Action pull-down menu, select **Set VPD**, and then click **Go**. The panel shown in Figure 12-6 opens.

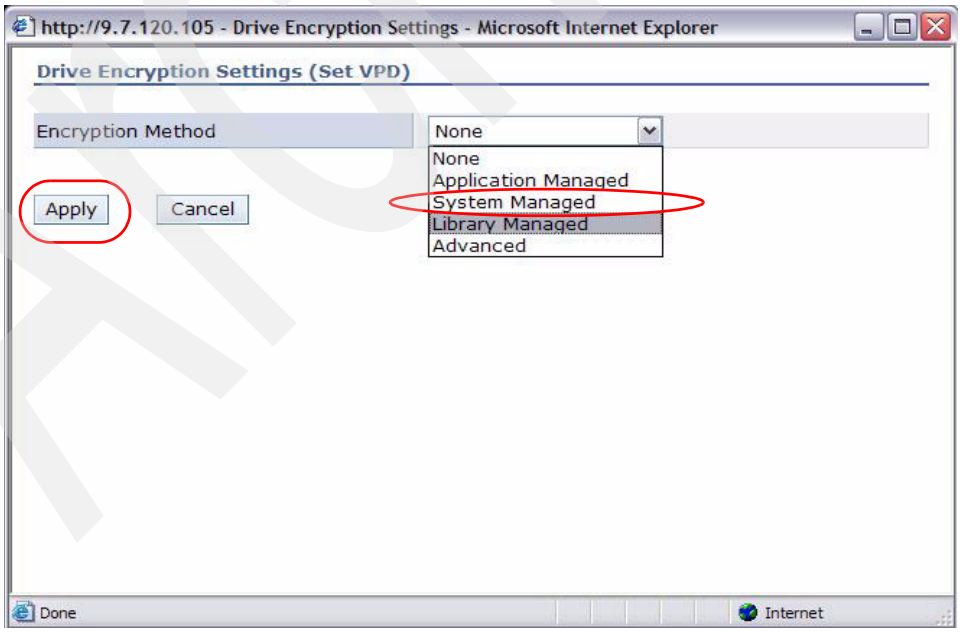


Figure 12-6 Encryption Method panel

3. In this panel, select the Encryption Method **System-Managed** from the pull-down menu, and click **Apply**. Being able to set encryption in this manner was made available with Library Manager microcode level 535.

3494 Library Manager user interface

To update the tape drive definitions for encryption using the 3494 Library Manager user interface, start with the main Library Manager panel as shown in Figure 12-7.

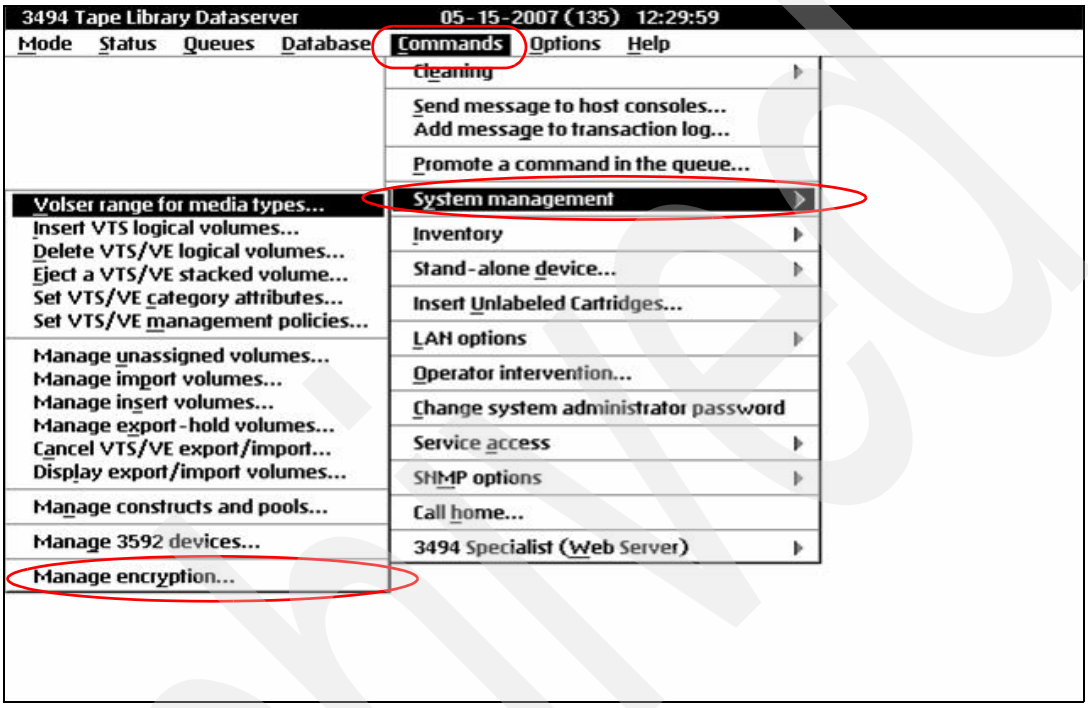


Figure 12-7 Selections for Manage Encryption of the 3494 Library Manager

To perform the implementation:

1. Select **Commands** → **System Management** → **Manage Encryption** to display a **Manage Encryption** panel similar to Figure 12-8 on page 392.

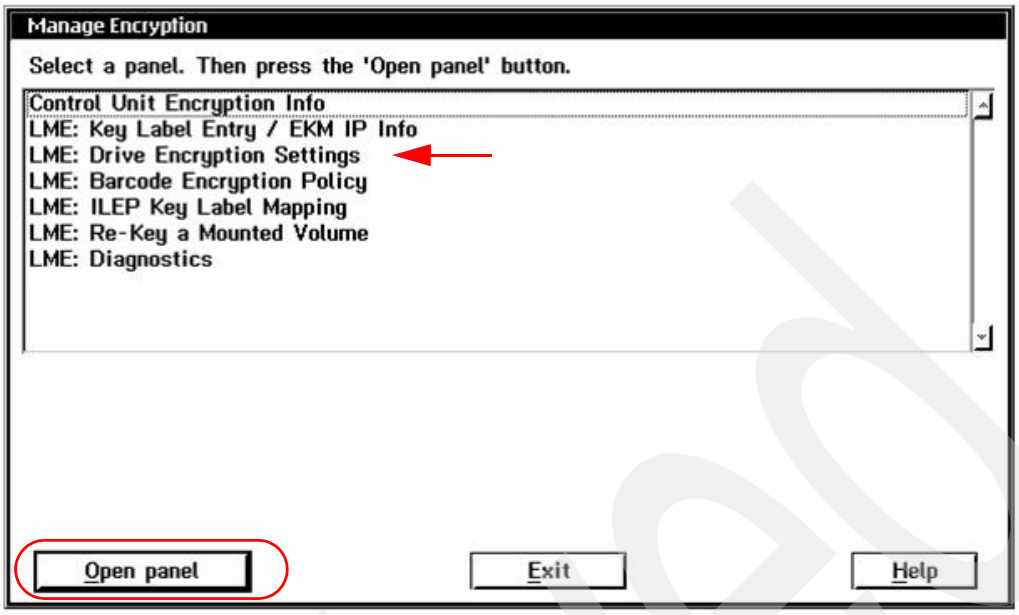


Figure 12-8 Manage Encryption panel

2. With the introduction of 3494 Library Manager microcode level 535, this panel contains the line **LME: Drive Encryption Settings** or **SME: Drive Encryption Settings** or similar line. Highlight that entry and click **Open panel**, which displays a panel similar to Figure 12-9.

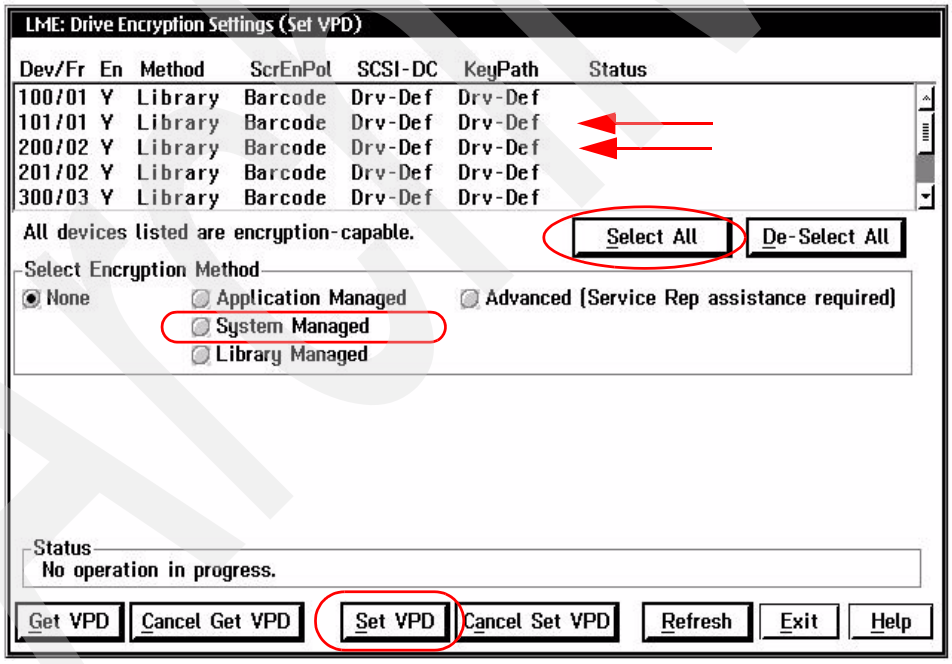


Figure 12-9 Drive Encryption Settings panel

3. This panel lists all of the encryption-capable drives and their current settings. Select the drives that will be attached to z/OS (or use **Select All**), click the **System Managed** radio button, and then click **Set VPD**.

12.4.3 Implementation steps for the IBM TS3400 Tape Library

In this section, we describe how to implement the IBM TS3400 Tape Library.

IBM TS3400 Tape Library Specialist

To update the tape drive definitions, use the Web browser interface of the TS3400. You must log in as an administrator. At the Java Security Warning message, simply click **Run** to start the specialist. Figure 12-10 shows the initial **System Summary** panel of the IBM TS3400 Tape Library Specialist.

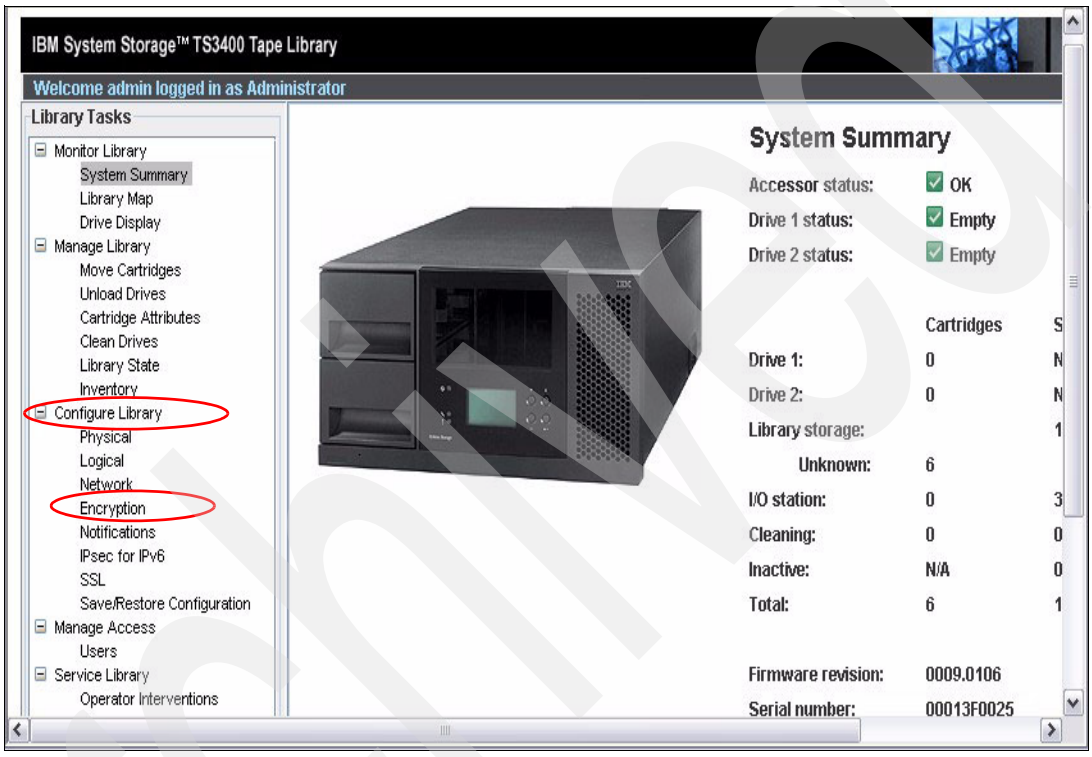


Figure 12-10 Main entry or welcome panel of the TS3400 Tape Library Specialist

To perform the implementation:

1. There are only two TS1120 drives in a TS3400, so normally you only have one logical library and we assume that here. On the left side of the panel, select **Configure Library** → **Encryption**, which displays a panel similar to that in Figure 12-11 on page 394.

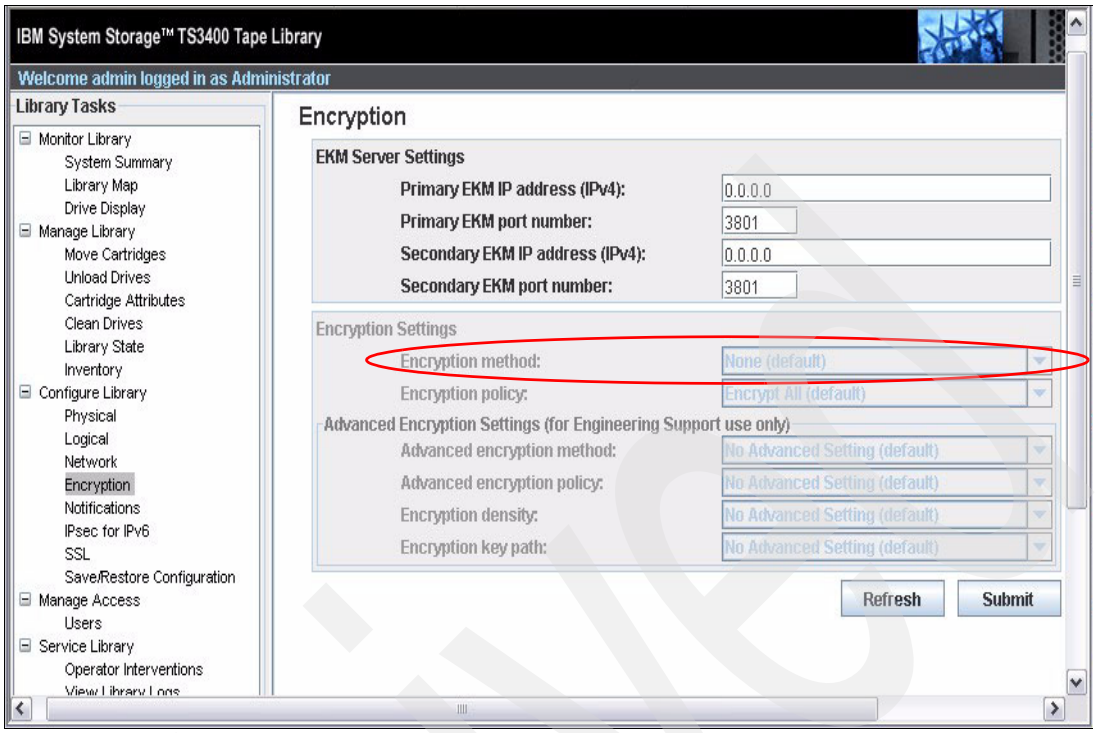


Figure 12-11 Drive Encryption Settings panel

2. In this panel, under the Encryption Settings, select the Encryption method of **System-Managed** from the pull-down. For **System-Managed**, this is the only specification required. Then, click **Submit.** This process enables the one or two TS1120 drives in the TS3400 library for System-Managed Encryption.

Repeat these steps for any other TS3400 libraries that are attached with the tape control unit.

12.5 Tape control unit implementation

In the planning phase, you should have ordered FC9595 (Plant) or FC5595 (Field) for your 3592 Model J70 or TS1120 Model C06 tape control units. Ask your service support representative (SSR) to perform the installation steps for these features *only after* you have installed all of the required operating system support for TS1120 tape drives with encryption.

12.6 z/OS implementation steps

In this section, we describe the host software implementation steps that are required for tape data encryption.

12.6.1 z/OS software maintenance

The following z/OS software components have been enhanced to support tape data encryption on the TS1120 and TS1130 tape drive:

- Access method services (AMS)
- DFSMS data set services (DFSMSdss)

- DFSMS hierarchical storage manager (DFSMShsm)
- DFSMS removable media manager (DFSMSrmm)
- Encryption Key Manager component for the Java platform (EKM)
- Input/output supervisor (IOS)
- MVS job entry subsystem 3 (JES3)
- Object access method (OAM)
- Storage Management Subsystem (SMS)

You have to check the 3592DEVICE preventive service planning (PSP) bucket to determine what z/OS maintenance is required to support an encryption-enabled TS1120 or TS1130 drive on your version of the z/OS operating system. Support was available beginning with z/OS V1.4 and later. This software maintenance must be installed before the TS1120 or TS1130 drives are enabled for encryption. *Otherwise, the encryption-enabled TS1120 drives will not come online.*

12.6.2 Update PARMLIB member IECIOS*xx*

In support of the Encryption Key Manager (EKM), the IECIOS*xx* PARMLIB member has a new EKM parameter. If in-band key management is used, you must modify this PARMLIB member to include the TCP/IP-related information required to direct the I/O Supervisor (IOS) proxy to an appropriate EKM (primary and secondary). This tells z/OS how to communicate with the primary EKM and (optionally) the secondary EKM.

IPv6 support has been provided with APAR OA22271.

Specify the host names for the primary and secondary EKMs. For each EKM, either a host name with an optional port or a decimal IP address with an optional port can be specified as shown in Example 12-1. The default for the port is normally 3801.

Example 12-1 PARMLIB member IECIOSxx

```
EKM PRIMARY=host.name.com:port[,SECONDARY=127.0.0.1:port]
```

The host name can contain up to 60 characters. The DNS search suffixes are automatically appended, which allows searches with shorter names.

Note: When the EKM subcommand is specified through PARMLIB, *omit* the comma after the word EKM and leave a blank space between EKM and the specified parameter.

EKM settings can be changed by selecting another IECIOS*xx* member. Do this by using the SET IOS=*xx* command dynamically.

Refer to *z/OS V1R7.0 MVS Initialization and Tuning Guide*, SA22-7591-03, and *z/OS V1R7.0 MVS Initialization and Tuning Reference*, SA22-7592-11, for reference for the command.

To display the current EKM settings, use either of the following commands:

- `D IOS,EKM`
- `D IOS,EKM,VERIFY=ALL`

If you also specify VERIFY, you can verify the availability of the primary and secondary EKMs.

See Figure 12-12 on page 396 for the results of the command.

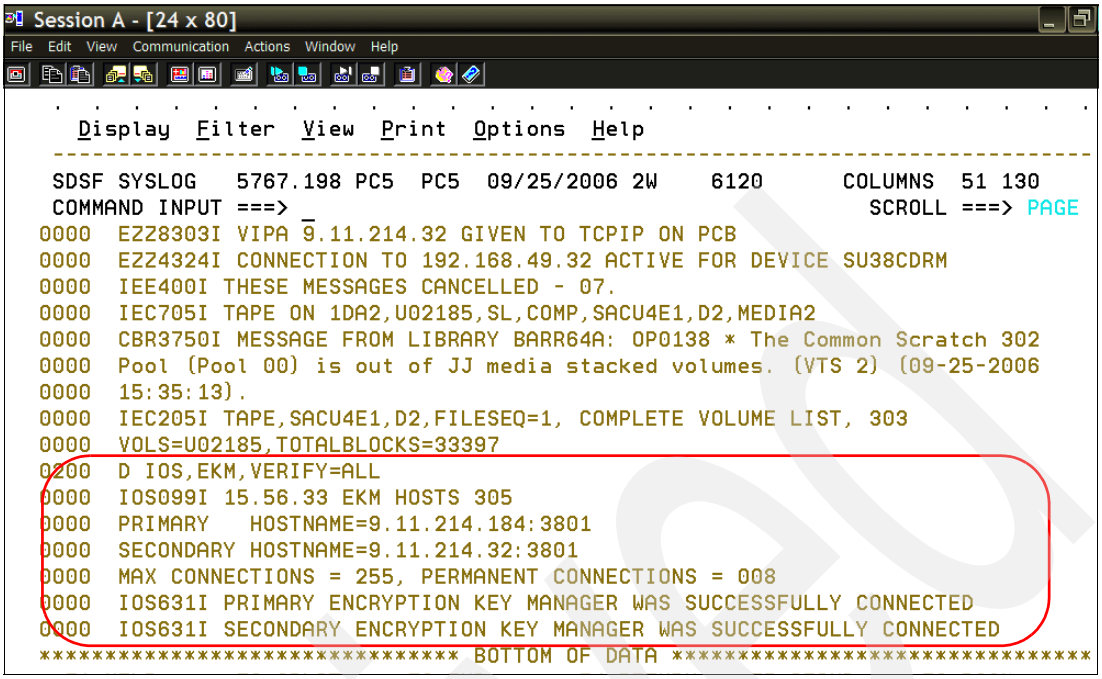


Figure 12-12 The D IOS,EKM,VERIFY=ALL command shows both EKM TCP/IP addresses

The following command can dynamically change the settings of the EKM:

```
SETIOS EKM
```

The command details are shown in Example 12-2.

Example 12-2 SETIOS EKM command details

```
SETIOS EKM[,PRIMARY={dns_name[:port]   } ]
                or   {ip_address[:port]}
                or   {NONE             }
       [,SECONDARY={dns_name[:port]    }]
         or        {ip_address[:port] }
         or        {NONE              }
      [,MAXCONN=dd1 ]
      [,MAXPCONN=dd2 ]
```

For details of the DISPLAY IOS,EKM and SETIOS EKM commands, refer to *z/OS V1R7.0 MVS System Commands,* SA22-7627-12.

12.6.3 Define or update Data Class definitions

Tape data encryption can be requested through the Data Class construct. The Recording Technology specified in the Data Class determines whether a cartridge is written in J1A format (E1), in TS1120-E05 or TS1130-E06 format (E2), or in encrypted E05 format (EE2).

The steps are:

1. You can define new Data Class constructs, or you can update the Recording Technology parameter of existing Data Classes. Figure 12-13 on page 397 shows the ISMF Data Class Alter panel where you change the `Recording Technology`. In our example, encryption is requested by specifying `EE2`.

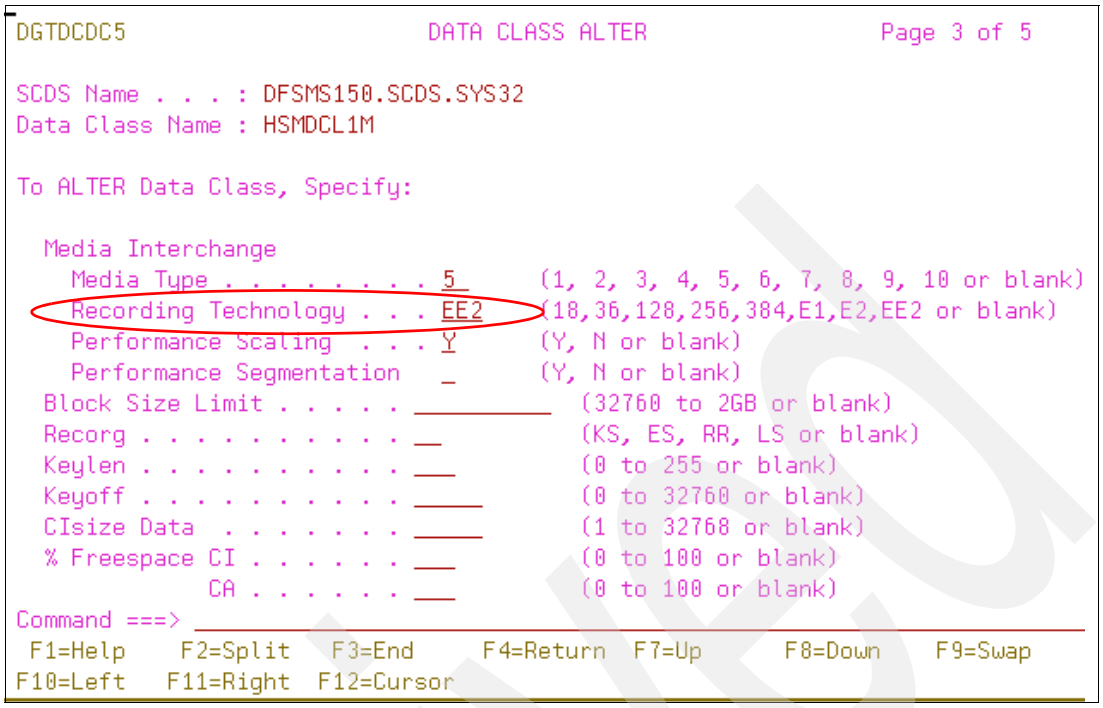


Figure 12-13 Data Class Alter (Page 3 of 5)

2. The Media Type must be 5, 6, 7, 8, 9, or 10. The Recording Technology must be EE2 to encrypt the tape. The Performance Scaling is usually an N and Performance Segmentation is not used. The other parameters do not apply to tape. Select PF8 after you have entered or updated the Recording Technology. On the panel that follows, you also have to enter the Key Labels and the Encoding for Key Labels for both key labels as shown in Figure 12-14 on page 398.

   Key Label has to be the name of a key label that you have loaded into the EKM keystore. Encoding for Key Label is either L for Label or H for Hash. L can be used (if you and the site that will read the tape) will have the same key label names for their corresponding certificates. H has to be used when the key label names might differ at the location where the tape will be read, for example, a business partner's location.

   You can specify one or both key labels. If you specify only Key Label 1, then Key Label 1 will be used for both keys stored on the 3592 cartridge.

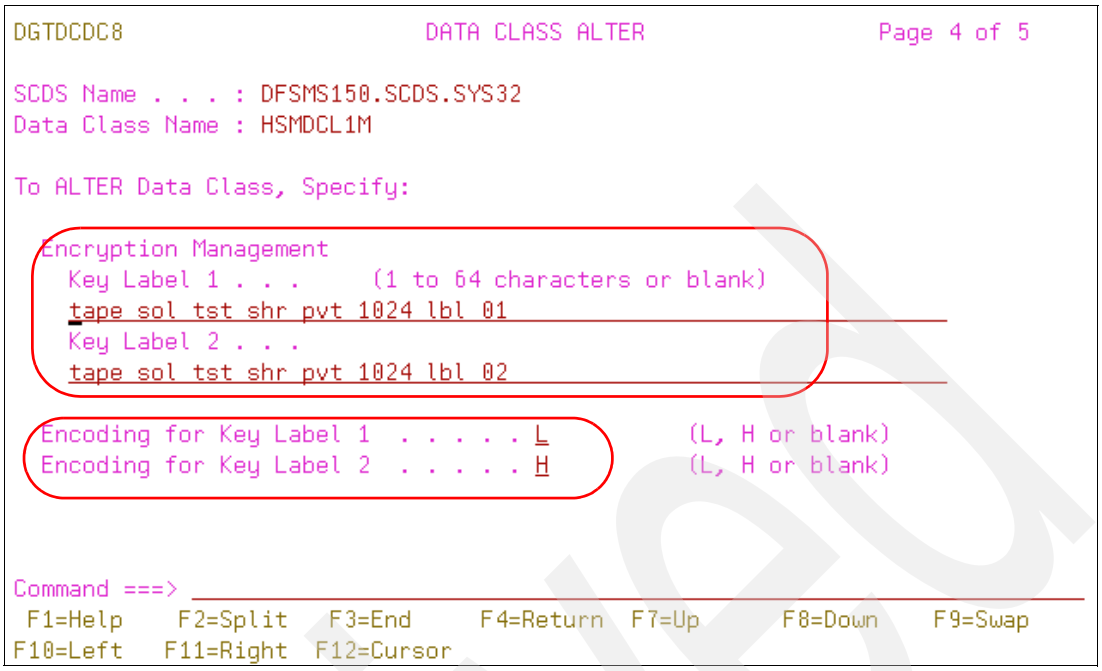


Figure 12-14 Data Class Alter (Page 4 of 5)

If you change existing Data Classes, verify your Data Class Automatic Class Selection (ACS) routine to make sure that you are assigning the correct constructs. If you create new Data Classes, update your Data Class ACS routine to have the new constructs assigned to those tape data sets that you want to have encrypted.

To activate the new SMS definitions:

1. Translate the Data Class ACS routine.
2. Validate the ACS routines.
3. Activate the SMS Control Data Set (SCDS).

Key labels in JCL

In addition to using a Data Class construct to enable encryption and assign the key labels, you can optionally assign the key labels using job control language (JCL) as shown in Example 12-3 on page 399. However, if you do this, encryption must still be invoked through a Data Class specification using a Recording Technology of EE2. The JCL parameters on the DD statement are as follows, where *x* = 1 or 2:

- KEYLABL*x*=
- KEYENCD*x*=

Example 12-3 Sample JCL to write an encrypted cartridge

```
//CO2STRW1   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.CO2M5CX2.PC5.NOPOOL.CO2STRS1.MASTER,
//         KEYLABL1='TAPE_SOL_TST_SHR_PVT_1024_LBL_02',
//         KEYENCD1=L,
//         KEYLABL2='TAPE_SOL_TST_SHR_PVT_2048_LBL_03',
//         KEYENCD2=H,
//         LABEL=(1,SL),UNIT=CO2M5CX2,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
/*
```

z/OS produces messages indicating when a tape has been encrypted and what key labels and key modes were used. The job log for the job in Figure 12-15 lists the keys that were used.

```
 SDSF OUTPUT DISPLAY C02STRW1 JOB05753  DSID     2 LINE 0       COLUMNS 11- 90
 COMMAND INPUT ===> _                                         SCROLL ===> CSR
******************************** TOP OF DATA ********************************
          J E S 2  J O B  L O G  --  S Y S T E M  P C 5    --  N O D E  T U C

JOB05753 ---- THURSDAY,  21 SEP 2006 ----
JOB05753  IRR010I  USERID BARISKA  IS ASSIGNED TO THIS JOB.
JOB05753  $HASP373 C02STRW1 STARTED - INIT AS   - CLASS B - SYS PC5
JOB05753  IEF403I C02STRW1 - STARTED - TIME=14.06.57
JOB05753  IEF233A M 0780,J1G151,,C02STRW1,CREATE1
JOB05753  IEC205I SEQ001,C02STRW1,CREATE1,FILESEQ=2, COMPLETE VOLUME LIST,  150
          VOLS=J1G151,LISTED VOL(S) HAVE BEEN DATA ENCRYPTED,KL1CD:L,KL2CD:H,
          KL1=tape_sol_tst_shr_pvt_1024_lbl_01,
          KL2=tape_sol_tst_shr_pvt_1024_lbl_01,TOTALBLOCKS=9
JOB05753  IEF234E K 0780,J1G151,PVT,C02STRW1,CREATE1
JOB05753  JOB=C02STRW1 STEP=CREATE1  PGM=IEBDG    CC=0000
JOB05753  IEF404I C02STRW1 - ENDED - TIME=14.07.38
JOB05753  $HASP395 C02STRW1 ENDED
S2 JOB STATISTICS ------
2006 JOB EXECUTION DATE
  23 CARDS READ
  71 SYSOUT PRINT RECORDS
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP      F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 12-15 MSGIEC205I indicates key labels and key codes used

Note: The JCL data definition (DD) statements override encryption specifications in the Data Class. If only one KEYLABL1 statement and only one KEYENCD1 are coded in the JCL, a second keylabel and keycode with the same information will be generated on the cartridge automatically. Whenever the first standard label (SL) on a cartridge contains encrypted data, all following SL file data behind that will be encrypted. Knowing this, no further JCL or Data Classes are required to write encrypted data to the same cartridge. You can prepare cartridges for encryption purposes this way by writing a small file with LABEL=(1,SL) to a cartridge.

12.6.4 Considerations for JES3

To allow for JES3 to build a list of Library Device Group (LDG) names, you update the JES3 INISH deck. Library Device Group names are a predefined set of tape subsystems within the JES3plex. LDGs are esoteric groups in HCD and defined to JES3 as SETNAME groups. JES3 selects a device from the LDGs and passes it to allocation.

For tape data encryption, you must define two new Library Device Group names:

LDL*sssss* Includes any 3592 Model E05 encryption-capable device emulating a 3590 Model B within the library indicated by serial number *sssss*.

LDG359L Includes any 3592 Model E05 Encryption-capable device emulating a 3590 Model B in any library in the JES3plex.

The DEVSERV command QT for device 1E0C, Read Device Characteristic returns the serial number under LIBID.

To determine what library serial number *sssss* is used, enter the MVS command shown in Example 12-4.

Example 12-4 DS QT command

```
DS QT,1E0C,RDC
 IEE459I 11.14.29 DEVSERV QTAPE 272
 UNIT DTYPE  DSTATUS CUTYPE  DEVTYPE  CU-SERIAL  DEV-SERIAL ACL LIBID
 1E0C 3590L  ON-RDY  3590A60 3590E1A* 0113-45731 0113-45731 I   CA001
   READ DEVICE CHARACTERISTIC
 3590603590100190 4EDC0000B4D7FD48 6900000000000000 3590603590200009
 0CA0010200000200 4683800004000000 0400001113800000 0000000000000000
 ****      1 DEVICE(S) MET THE SELECTION CRITERIA
 ****      1 DEVICE(S) WITH DEVICE EMULATION ACTIVE
```

The command response shows that tape unit address 1E0C belongs to a library with serial number CA001.

In addition, you might find helpful the following MVS command DEVSERV Query Library. It shows the attached devices and the associated LIBPORT-IDs of an ATL. See Example 12-5 on page 401.

Note: The DS QL,*nnnnn* can only be used for already initialized libraries.

Example 12-5 DS QL command

```
DS QL,CA001
IEE459I 15.15.43 DEVSERV QLIB 301
LIBID  PORTID       DEVICES
CA001  01           1E00* 1E01* 1E02* 1E03* 1E04* 1E05* 1E06* 1E07*
                    1E08* 1E09* 1E0A* 1E0B*
       02           1E0C* 1E0D* 1E0E
```

12.6.5 Tape management system

The tape management systems typically track the recording format of a tape volume. With encryption, a new recording format (EEFMT2) is used. Refer to your tape management system provider for details about their encryption support.

12.6.6 DFSMSrmm support for tape data encryption

DFSMSrmm, as a feature of z/OS, has been updated for the new media types, recording technology, and encryption key label support.

The following RMM TSO commands are expanded for MEDIATYPE and RECORD FORMAT:

- ADDVOLUME
- CHANGEVOLUME
- SEARCHVOLUME

Table 12-1 shows media names and DFSMSrmm media names. We compare the media names used in DFSMS with the media names used in DFSMSrmm. Only MEDIA5 (ETC) up to MEDIA10 (EEEWTC) can be used for encryption on TS1120 tape drives, because only these media names are requesting 3592 media.

Table 12-1 Media types

DFSMS media name	DFSMSrmm media name	Cartridge media	Capacity without compression[a]
MEDIA 1	CST	3490 Standard	400 MB
MEDIA 2	ECCST	3490 Extended	800 MB
MEDIA 3	HPCT	3590 J cartridge	10, 20, or 30 GB
MEDIA 4	EHPCT	3590 K cartridge	20, 40, or 60 GB
MEDIA5	ETC	3592 JA cartridge	300 or 500 GB
MEDIA6 (WORM)	EWTC	3592 JW cartridge	300 or 500 GB
MEDIA7	EETC	3592 JJ cartridge	60 or 100 GB
MEDIA8 (WORM)	EEWTC	3592 JR cartridge	60 or 100 GB
MEDIA9	EEETC	3592 JB cartridge	700 GB
MEDIA10 (WORM)	EEEWTC	3592 JX cartridge	700 GB

a. Effective capacities with compression are three times as much, assuming a 3:1 compression ratio. Your compression ratios will vary with the data. The larger capacities are applicable for 3592 cartridges written in encrypted format.

Note: Media 9 and Media 10 support is only available on z/OS V1R5 and higher.

Figure 12-16 shows an RMM panel indicating an encrypted tape volume. Panel ID EDGPT110 shows for tape volume J1G151 with a recording format of EEFMT2.

```
EDGPT110              DFSMSrmm Volume Details - J1G151
Command ===> _

Volume . . . . . : J1G151     VOL1 volser :          Rack number : J1G151
Media name . . . . : 3480                            Status  . . : MASTER
                                                                   More:
Recording format . : EEFMT2          Erase volume . . . . . : NO
Compaction . . . . : YES             Notify owner . . . . . : NO
Attributes . . . . : NONE            Expiry date ignore . . : NO
Availability . . . : VITAL RECORD    Scratch immediate  . . : YES

Owner  . . . . . : BARISKA        Owner access . . . . . : ALTER
Assigned date  . . : 2006/221     Assigned time  . . . . : 10:00:45
File 1 system ID . : PC5
Security name  . . : UNCLASS
Classification . . : UNCLASSIFIED
Account number . . : CONSOLE
                                  MVS use  . . . . . : YES
Jobname  . . . . . : ENCREADL     VM use . . . . . : NO
 F1=Help      F2=Split     F3=End       F4=Return    F7=Up        F8=Down
 F9=Swap      F10=Left     F11=Right    F12=Retrieve
```

Figure 12-16 Recording format EEFMT2

The mountable tape volume list option in ISMF (PANEL ID=DGTLGP31) under Recording Technology can also be used to determine if a volume is encrypted. EEFMT2 is the indication for encryption. See Figure 12-17.

```
  Panel  List  Utilities  Scroll  Help
------------------------------------------------------------------------------
DGTLGP31                  MOUNTABLE TAPE VOLUME LIST
Command ===> _                                          Scroll ===> HALF
                                                  Entries 7-18 of 104
Enter Line Operators below:                       Data Columns 6-10 of 20

     LINE       VOLUME LIBRARY   STORAGE   MEDIA     RECORDING
     OPERATOR   SERIAL NAME      GRP NAME  TYPE      TECHNOLOGY  COMPACTION
    ---(1)----  -(2)-- --(6)---  --(7)---  --(8)---  ---(9)----  ---(10)---
                J1G151 CASH02    SGCASH02  MEDIA5    EEFMT2      YES
                J1G152 CASH02    SGCASH02  MEDIA5    EEFMT2      NO
                J1G153 CASH02    SGCASH02  MEDIA5    EEFMT2      YES
                J1G154 CASH02    *SCRTCH*  MEDIA5    EFMT1       ---
                J1G155 CASH02    SGCASH02  MEDIA5    EEFMT2      YES
                J1G156 CASH02    SGCASH02  MEDIA5    EEFMT2      YES
                J1G157 CASH02    *SCRTCH*  MEDIA5    EEFMT2      YES
                J1G158 CASH02    *SCRTCH*  MEDIA5    EEFMT2      YES
                J1G159 CASH02    *SCRTCH*  MEDIA5    EEFMT2      YES
                J1G270 CASH02    *SCRTCH*  MEDIA5    EEFMT2      YES
                J1G271 CASH02    *SCRTCH*  MEDIA5    EFMT1       ---
                J1G272 CASH02    *SCRTCH*  MEDIA5    EEFMT2      YES
 F1=Help    F2=Split   F3=End     F4=Return  F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cursor
```

Figure 12-17 Tape Volume List in ISMF

The only way to see more information for a volume is to enter a LISTVOLUME (LV) command. It shows the key labels and method used (Label or Hash).

The LISTVOLUME command can be issued from the ISPF option 6 command prompt or as a TSO command. The command syntax can be any of the following lines:

- RMM LV *VOLSER* VOL
- RMM LV *VOLSER* ALL
- TSO RMM LV *VOLSER*

Example 12-6 shows the response to an RMM LV command.

Example 12-6 RMM LV response

```
Volume information:
 Volume = J1G153   VOL1  =          Rack  = J1G153    Owner   = DPA1
   Type = PHYSICAL         Stacked count  = 0         Jobname = CO2STRWG
   Worldwide ID =

 Creation:  Date = 08/31/2006  Time = 08:29:02  System ID = SYS6
 Assign:    Date = 09/08/2006  Time = 14:39:42  System ID = SYS6
 Expiration date = 09/09/2006 Original      =
 Retention date  = WHILECATLG Set retained  = NO
 Data set name = TAPE.CO2M5CX2.PC5.NOPOOL.CO2STRSG.MASTER
 Volume status:
 Status = MASTER    Availability = Vital Record      Label = SL
 Current label version =            Required label version =
 Media information:
 Density = IDRC  Type = ETC       Format - EEFMT2    Compaction - YES
 Special attributes   = NONE      Vendor -
 Encryption Key Labels:                                          Method:
 1=tape_sol_tst_shr_pvt_1024_lbl_01                              LABEL
 2=tape_sol_tst_shr_pvt_1024_lbl_01                              HASH
 Action on release:
 Scratch immediate  = Y  Expiry date ignore = N
 Scratch = Y  Replace = N  Return = N  Init = N  Erase = N  Notify = N
 Actions pending:
 Scratch = N  Replace = N  Return = N  Init = N  Erase = N  Notify = N
 Storage group  = SGCASH02
 Loan location  =          Account = CONSOLE
 Description    =
 Security class = UNCLASS   Description = UNCLASSIFIED

 Access information:
 Owner access = ALTER   Volume access = NONE    Last change = *OCE
 VM use = N   MVS use = Y
 Access list:

 Statistics:
 Number of data sets = 1       Data set recording= ON
 Volume usage(Kb)= 54          Use count         = 17639
 Volume capacity = 286102      Percent full      = 0
 Date last read  = 09/14/2006  Date last written = 09/08/2006
 Drive last used = 07C1        Write mount count = 2
 Volume sequence = 1           Media name        = 3480
 Previous volume =             Next volume       =
 Product number  =             Level             = V  R  M
```

```
Feature code    =
Error counts:
Temporary read = 0      Temporary write = 0
Permanent read = 0      Permanent write = 0

Movement tracking date = 08/31/2006       Intransit = N
In container           =                  Move mode = AUTO

Location:    Current      Destination  Old          Required     Home
Name       = CASH02                                             CASH02
Type       = AUTO                                               AUTO
Bin number =
Media name =
```

In Example 12-6 on page 403, two key labels are used:

- Key label with method LABEL
- Key label with method HASH

Note: For tape management systems other than DFSMSrmm, contact your vendor.

12.6.7 DFSMSdfp access method service

Use the commands in this section to change, enter, or display information in the tape control database (TCDB) catalog.

In support of tape data encryption, the access method service (AMS) commands, CREATE VOLUMEENTRY, ALTER VOLUMEENTRY, DCOLLECT, and LISTCAT have been changed to support the recording technique for encryption. One subparameter, EEFMT2 for the parameter RECORDING, has been added for CREATE VOLUMEENTRY and ALTER VOLUMEENTRY.

ALTER VOLUMEENTRY

Use the ALTER VOLUMEENTRY command to change the recording technology fields in the volume records of a tape library:

- EEFMT2 subparameter indicates read/write on an EEFMT2 track device.
- EEFMT2 subparameter of RECORDING is only allowed with media types MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, or MEDIA10. The use of MEDIA1 through MEDIA4 produces an IDC3226I error message that is generated twice: one time for EEFMT2 and one time for the media type.

You can alter a tape volume with the following command:

```
ALTER VOLUMEENTRY RECORDING(EFMT1|EFMT2|EEFMT2)
```

Note that we list only the recording technologies currently supported with the IBM TS1120 Tape Drive:

- EFMT1 is J1A non-encrypted format.
- EFMT2 is E05 non-encrypted format.
- EEFMT2 is E05 encrypted format.

CREATE VOLUMEENTRY

Use the CREATE VOLUMEENTRY command to create volume records of a tape library:

- EEFMT2 subparameter indicates read/write on an EEFMT2 device.
- EEFMT2 is only allowed with media types MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, or MEDIA10. Any use of MEDIA1 through MEDIA4 produces an IDC3226I error message that is displayed twice: one time for EEFMT2 and one time for the media in question. The double display indicates an incompatibility between the EEFMT2 subparameter and the media type displayed.
- If MEDIA5, MEDIA6, MEDIA7, or MEDIA8 is specified and RECORDING is not specified, default to EFMT1 for RECORDING value.
- If MEDIA9 or MEDIA10 is specified and RECORDING is not specified, default to EFMT2 for RECORDING value.

The following example shows the EEFMT2 subparameter for CREATE VOLUMEENTRY:

```
CREATE VOLUMEENTRY RECORDING(EFMT1|EFMT2|EEFMT2|UNKNOWN)
```

Note that we only list the recording technologies currently supported with the IBM TS1120 Tape Drive.

DCOLLECT

The DCOLLECT command has values added to its definitions for DDCRECTE to allow the constant DDCEEFM2 for EEFMT2 devices.

LISTCAT

Use the LISTCAT command to display the new value associated with the RECORDING parameter for VOLUME entries, as shown in Example 12-7.

Example 12-7 LISTCAT command

```
LISTCAT VOLUMEENTRIES ALL

IDCAMS SYSTEM SERVICES
LISTING FROM CATALOG --  SYS1.VOLCAT.V0

VOLUME ENTRY----V0A2991
DATA-VOLUME
LIBRARY---------ATLIB02
RECORDING--------EEFMT2
MEDIA-TYPE-------MEDIAx
```

MEDIA*x* represents either MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, or MEDIA10.

12.6.8 Data Facility Data Set Services considerations

Data Facility Data Set Services (DFSMSdss), as a functional component of z/OS, allows you to copy, move, dump, and restore data sets and volumes. With this support, hardware encryption joins software (or host-based) encryption as a means of encrypting your installation's tape volumes. Because DFSMSdss avoids performing double encryption of tape data, you must determine which type of encryption, if any, is used for your tape volumes. DFSMSdss prevents you from combining both types of encryption to avoid double encryption of tape volumes.

12.6.9 DFSMS Hierarchal Storage Manager considerations

The Data Facility System Managed Subsystem Hierarchical Storage Manager (DFSMShsm), also a functional component of z/OS, automatically manages low activity and inactive data in both system-managed and non-system-managed environments. DFSMShsm also provides automatic backup and recovery of active data in those environments. DFSMShsm can use the encryption-capable IBM TS1120 Tape Drive (3592-E05) for all functions. DFSMShsm normally uses non-WORM media (MEDIA5, MEDIA7, or MEDIA9) for non-aggregate backup and recovery support (ABARS) functions. DFSMShsm uses all media, including WORM (MEDIA6, MEDIA8, and MEDIA10) for ABARS processing. DFSMShsm can use the WORM media for non-ABARs processing if it is specifically enabled in your installation.

To use tape hardware encryption, you must modify your SMS Data Class definitions to request encryption from the encryption-capable tape drives. With the support for the encryption-capable IBM TS1120 or TS1130 tape drive, hardware encryption joins software (or host-based) encryption as another means of encrypting your installation's dump data. As a result, the method for requesting encryption now depends on whether you plan to use hardware encryption or host-based encryption:

- To request hardware encryption for a dump class, specify it in the SMS Data Class for the dump data.
- To request host-based encryption for a dump class, use the DFSMShsm DEFINE DUMPCLASS(ENCRYPT) command. With ENCRYPT, include the RSA or KEYPASSWORD subparameters (or NONE) to specify the type of host-based encryption.

Note: Although software (or host-based) encryption supports only dump data, all HSM functions are supported with hardware encryption.

If your dump classes are currently defined to use host-based encryption (and possibly host-based compression before encryption), we recommend that you remove the host-based encryption requests from any dump classes for which you plan to use tape hardware encryption.

During the process of migrating your dump classes to use hardware encryption, you might have several dump classes that are still defined to use host-based encryption, while their associated SMS Data Classes are defined to use tape hardware encryption. Here, DFSMSdss ignores requests for host-based encryption for these tape volumes and, instead, uses hardware encryption. This processing allows you to complete the migration to hardware encryption without having to modify your dump-requesting jobs. However, removing host-based encryption requests from a dump class when tape hardware encryption is also requested can avoid confusion concerning which process is active.

Important considerations:

- To determine whether hardware encryption or host-based encryption was used for a particular tape volume, check the associated dump volume record (DVL).
- If more than one dump class is specified (creating more than one dump copy), if those dump classes specify host-based encryption, if each dump class has a unique Data Class assigned, and if some but not all of the associated Data Classes request tape hardware encryption, all dump copies fail. In other words, tape hardware encryption can override host-based encryption for all dump classes associated with a source volume or none of the dump classes, but it cannot override a subset of those dump classes.

You can request information for encrypted tape volumes with list commands to the tape table of contents (TTOC) in the offline control data set (ODS) through any of the following commands:

- `LIST TTOC SELECT(EEFMT2) ODS(`*`ttoc.out.dataset`*`)`
- `LIST TTOC SELECT(ENCRYPTION) ODS(`*`ttoc.out.dataset`*`)`
- `LIST TTOC SE`LECT(NOENCRYPTION) ODS(*ttoc.out.dataset*)

For a list of the information for the dump volumes of the requested status managed by DFSMShsm, specify the LIST command with the DUMPVOLUME parameter without the volume serial number. Instead, include a status parameter, such as AVAILABLE, UNAVAILABLE, EXPIRED, UNEXPIRED, or NORETENTIONLIMIT. The command lists the volumes in alphanumeric sequence by volume serial number. For the commands, refer to *z/OS V1R7.0 DFSMSdfp Storage Administration Reference*, SC26-7402-05.

For more details, also refer to *z/OS V1R3.0-V1R8.0 DFSMS Software Support for IBM TotalStorage Enterprise Tape System 3592 (Model E05)*, SC26-7514-02.

12.7 z/VM implementation steps

z/VM provided support for encryption in the TS1120 or TS1130 tape drive (3592-E05) with z/VM Version 5. z/VM can enable encryption on behalf of guests that are not capable of doing so themselves. Guests that understand encryption, for example, z/OS, do not have to do anything special in z/VM for encryption.

Note the following information about z/VM versions and releases:

- z/VM V5 Releases 1 and 2 require the program temporary fix (PTF) for APAR VM64063.

  These two releases provide support for default keys only. Encryption is enabled with *ATTACH* or *SET RDEVice* commands. Encryption can also be enabled with the z/VM DASD Dump Restore (DDR) utility, but it utilizes its own unique externals in order to run in the absence of an underlying z/VM system, that is, stand-alone dump or restore.

- z/VM V5 Release 3 provided encryption support at the general availability (GA) level.

  This release expanded the encryption support to allow any *key label* defined to the EKM to be used when encrypting for guests. A key label represents a public-private key where the public key is used for encryption and the private key is used for decryption. In addition to the key label, z/VM also requires the *key encoding mechanism* operand to identify how the key label is to be used in the encryption of the data key. z/VM combines the two components (key label and key encoding mechanism) to form a *key alias* used by the ATTACH and SET RDEVice commands.

To perform the implementation:

1. Ensure the necessary z/VM software levels installed to support encryption.
2. Verify that the EKM has been implemented and that the key labels are stored in its keystore.
3. Set up tape drives for System-Managed Encryption using tape library panels. We discuss this in 12.7.1, “Tape library and tape control unit implementation” on page 408.
4. Set up the tape control units for encryption by having your IBM SSR install FC9595 (Plant) or FC5595 (Field) on the 3592 Model J70 Tape Control Unit or the TS1120 Model C06 Tape Control Unit.

5. Set up the tape control units for *out-of-band encryption* by using the Web interfaces of the TS3500, 3494, or TS3400 library to set the EKM TCP/IP addresses. We describe this in 12.7.2, “Out-of-band encryption” on page 408.
6. Define the key aliases to z/VM.
7. Use the ATTACH, DETACH, and SET RDEVice commands to control tape encryption.

12.7.1 Tape library and tape control unit implementation

Like z/OS, you must enable the TS1120 and TS1130 drives for System-Managed Encryption and configure the tape control units for encryption. We describe these steps in 12.4, “Tape library implementation” on page 386 and in 12.5, “Tape control unit implementation” on page 394 and we do not repeat the steps here.

12.7.2 Out-of-band encryption

Out-of-band encryption is required when the operating system does not have the support necessary to act as a proxy of the EKM. This is primarily for the operating systems: z/VM, z/VSE, and z/TPF. *Out-of-band encryption* uses the tape control unit as a proxy for communication with the EKM. In these cases, the 3592 Model J70 or the TS1120 Model C06 tape controller must be updated with the TCP/IP addresses of the EKM. This is done through the Library Manager interface if the drives are attached to a TS3500, 3494, or TS3400 tape library.

If the drives are in a C20 Silo frame or are rack-mounted, you must order *FC9595 (Plant)* or *FC5595* for each of the tape control units that will use out-of-band encryption. In this case, the IBM SSR is shipped instructions for setting up encryption in the tape control unit.

The implementation steps described in this section are the same for TS3500, 3953, and 3494 tape libraries and only necessary in case of out-of-band encryption. We will discuss the procedures for out-of-band encryption on the TS3400 tape library later in this section.

Note: It is important to know that the following steps can only be done at the Library Manager (LM) panels. These definitions are only necessary for *out-of-band encryption.*

Out-of-band encryption implementation on the TS3500 and 3494

We show the panels of the Library Manager (LM) to enable the libraries for out-of-band encryption. The steps are:

1. From the Library Manager console, select **Commands** → **System management** → **Manage Encryption**. See Figure 12-18 on page 409**.**

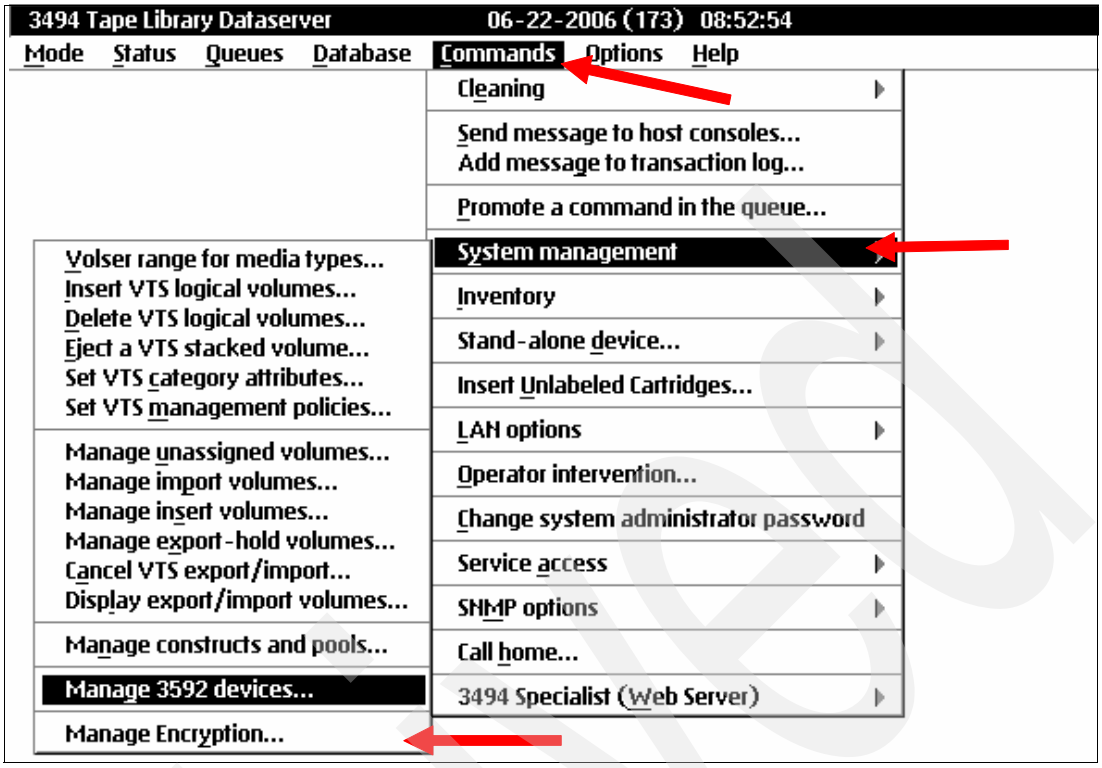


Figure 12-18 3494/3953 navigation for out-of-band encryption

The 3494/3953 Manage Encryption list box displays with the next Library Manager panel, as shown in Figure 12-19.

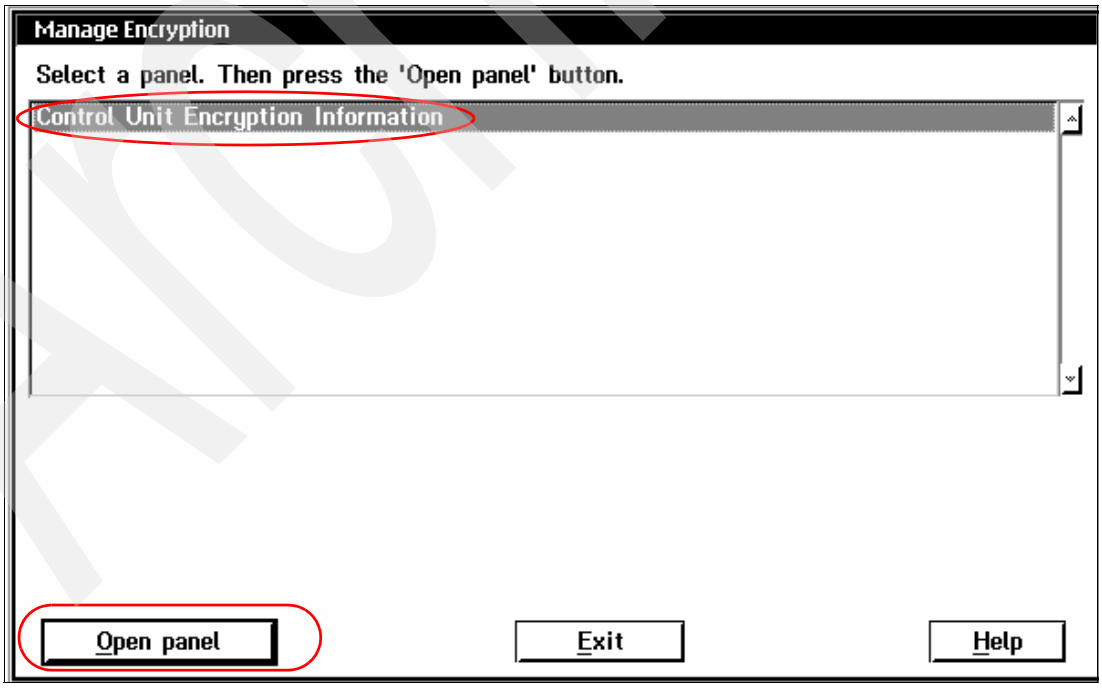


Figure 12-19 3494/3953 Manage Encryption selection list box

2. Select the **Control Unit Encryption Information** row and click **Open panel**. In the next panel (Figure 12-20), you insert the definitions for the control units.

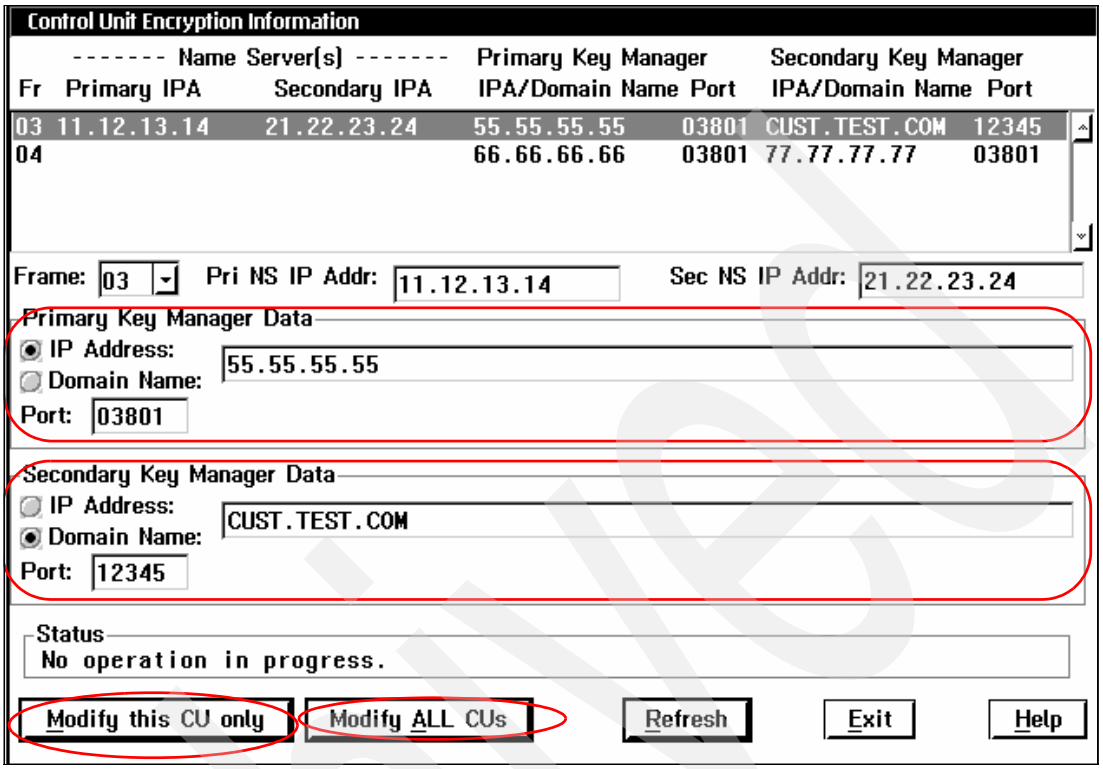


Figure 12-20 3494/3953 Control Unit Encryption Information entry panel

3. The Control Unit Encryption Information panel is available for every control unit or can be used for all of the control units in a tape library. You may specify up to two *key managers* for each control unit: one key manager for the primary EKM and, optionally, one key manager for a secondary EKM.

   If you specify a *domain name*, a *Network Services (NS) IP address* also has to be inserted. The information about this IP address can be obtained from the networking department. The IP addresses or domain names used must match the definitions of the EKM. Refer to 6.1.8, “Virtual IP Addressing” on page 175 for additional information.

   After setting the values in the panel, click **Modify this CU only**, or click **Modify ALL CUs** to set the same EKM addresses for all control units in the library.

Out-of-band encryption implementation on the TS3400

We show the panels of the TS3400 Library Specialist that are used to enable the control units for out-of-band encryption. To update the EKM server addresses, use the Web browser interface of the TS3400. You must log in as an administrator. Although the Java Security warning message appears, just click **Run** to start the specialist. Figure 12-21 on page 411 shows the initial System Summary panel of the TS3400 Tape Library Specialist.

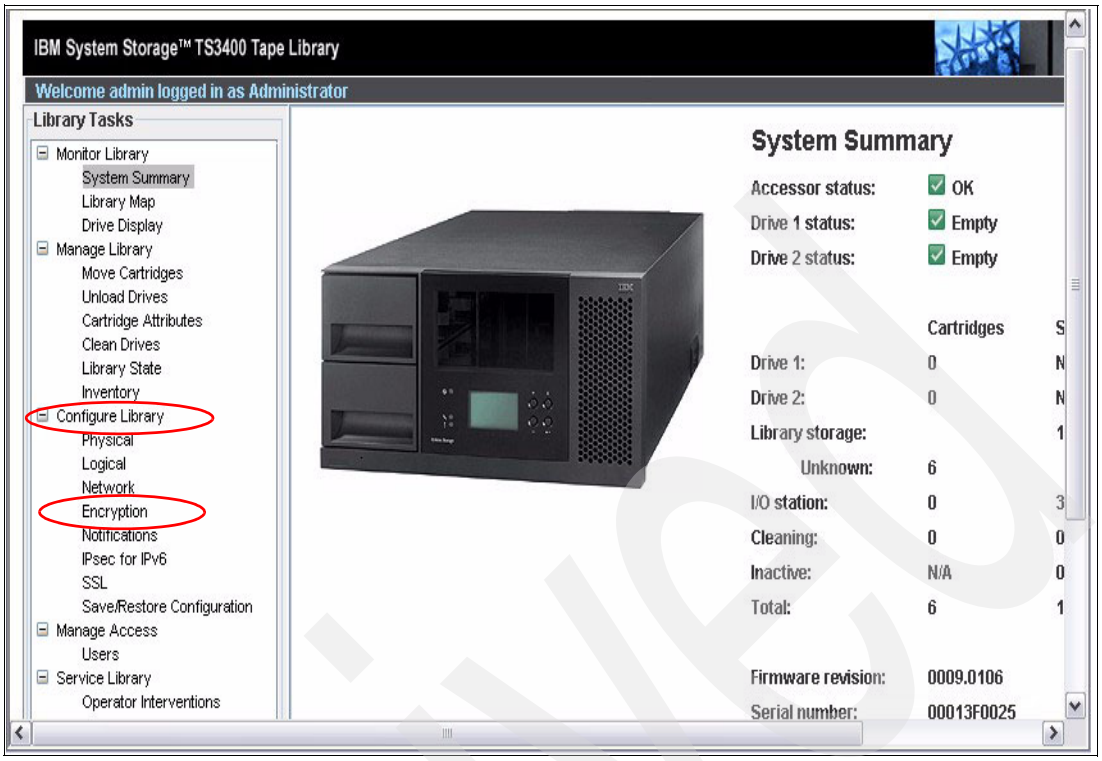


Figure 12-21 Main entry or welcome panel of the TS3400 Tape Library Specialist

To perform the implementation:

1. On the left side of the panel, select **Configure Library** → **Encryption**, which displays a panel similar to that in Figure 12-22.

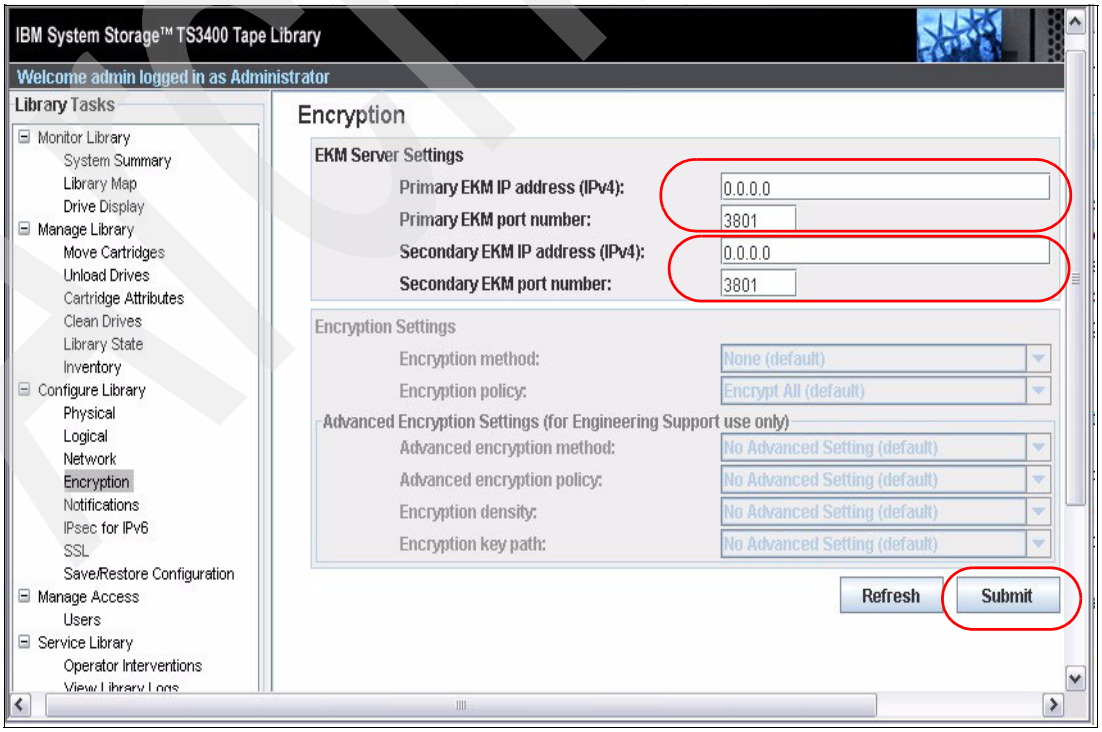


Figure 12-22 EKM Server Settings panel

2. In this panel, under EKM Server Settings, enter the EKM IP address and the EKM port number for the Primary EKM and (optionally) the Secondary EKM. Then, click **Submit.** Usually, the port number is 3801, but it might be different if this port number has a conflict within your network. The IP address can be an IPv4 address or an IPv6 address depending on which network settings you have set up. This procedure must be performed on each TS3400 library that is attached to the tape control unit.

12.7.3 Define key aliases to z/VM

Note: Information provided is only to be used in conjunction with guest operating systems that are not able to enable the hardware encryption environment themselves. Attempts to combine them are likely to create hardware problems, such as missing interrupts.

z/VM maintains a list of key aliases, which are defined with the SET KEYALIAS command, that represent the EKM's key labels to be used by z/VM. Each key alias is up to 32 characters long (including spaces) and contains the key label as well as a key mode of how that key label will be used to create an EEDK. A key label can be used directly (`LABEL`) or Hashed (`HASH`) when creating the EEDKs. Using `HASH` might be more convenient if the recipient of the encrypted cartridge has a different key label string defined to represent the matching keys. Using `LABEL` might be more convenient for internal data where the site that writes and the site that reads use the same key labels.

This information can be recalled with the QUERY KEYALIAS command for either a single alias or for the entire list of known aliases.

Important: You must set up KEYALIASs for all key label and key mode combinations that you plan to use. These will be used later in the ATTACH and SET RDEVice commands.

The formats of the SET KEYALIAS and QUERY KEYALIAS commands are:

```
SET KEYAlias aliasname {Label} KEYLabel keylabel
                       {Hash }

Query KEYAlias {ALL}
               {aliasname}
```

Refer to Example 12-8 for a sample of how the commands are used.

Example 12-8 SET KEYALIAS and QUERY KEYALIAS

```
set keyalias cow label keylabel 'moo moo moo'
set keyalias duck hash keylabel 'quack quack'
set keya 'old macdonald' label keyl 'Had a Farm'

q keyalias
KEYALIAS: (L) COW
          = MOO MOO MOO
KEYALIAS: (H) DUCK
          = QUACK QUACK
KEYALIAS: (L) OLD MACDONALD
          = HAD A FARM

q keya 'old macdonald'
KEYALIAS: (L) OLD MACDONALD
          = HAD A FARM
```

12.7.4 Using ATTACH and DETACH to control encryption

When dedicating a tape to a guest, the key aliases can be specified on the ATTACH command in order to select the key labels to use when creating the EEDKs. If an alias is not recognized, an error displays, and the ATTACH command fails. If no key aliases are specified, a set of default key labels, defined in the EKM, will be used to create the EEDKs.

At the time of the ATTACH, the tape drive must be either unloaded or at beginning-of-tape in order to establish the encryption environment. Thus, a tape cartridge is guaranteed to have a consistent set of data that is either completely encrypted with a unique set of EEDKs or completely unencrypted. Attempts to ATTACH a tape drive with encryption settings that do not meet these criteria will fail with an error message.

Guest operating systems that use the z/VM facilities to enable encryption are able to use the MULTIUSER option on ATTACH. However, all guests using the shared tape device must use the same encryption settings, in order to provide a consistent environment for enabling the encryption.

The encryption settings specified on an ATTACH persist until the drive is detached and unloaded. If a DETACH is issued with the LEAVE option, the encryption settings remain in place along with the mounted cartridge. This allows a subsequent ATTACH to be issued without any encryption settings, but it also allows the encryption environment to be passed to a new target user. In the case of the ATTACH of shared tape, the encryption settings persist until all instances are detached, and the device is marked as free again.

The format of the ATTACH command is shown here and followed by Example 12-9. Only the ATTACH parameters related to encryption are shown here:

- For z/VM V5.1 or V5.2:

  ```
  ATTach [NOQIOAssist] [KEY]
  ```

- For z/VM V5.3 and later:

  ```
  ATTach [NOQIOAssist] [KEY keyalias1 keyalias2]
  ```

Example 12-9 ATTACH examples

```
att 181 * key
TAPE 0181 ATTACHED TO JOECOOL 0181

det 181
TAPE 0181 DETACHED

att 181 * multi key cow duck
TAPE 0181 ATTACHED TO JOECOOL 0181

att 181 * multi key cow pig
HCPATR1128E Device 0181 not attached; a mismatch in hardware encryption settings
       was detected.
```

12.7.5 Using SET RDEVICE to control encryption

The SET RDEVICE FEATURE command (**set rdev feature**) can be used to enable the encryption environment without making changes to the ATTACH command. As with ATTACH, this command must also be issued when the drive is unloaded or at beginning-of-tape, but it also must be issued before it is dedicated to a guest. This permits the encryption environment to be enabled through the use of library or tape managers that issue a regular, unmodified,

ATTACH command. Up to two key aliases can be specified for each SET RDEVICE FEATURE command. If none are specified, the EKM's default keys are used.

The format of the SET RDEVICE FEATURE command is either of the following lines:

```
SET RDEVICE rdev FEATURE KEY [keyalias1] [keyalias2]
                         NOKEY

SET RDEVICE rdev1-rdev2 FEATURE KEY [keyalias1] [keyalias2]
                                NOKEY
```

Refer to the samples in Example 12-10. Only the SET RDEVICE FEATURE parameters related to encryption are shown here.

Example 12-10 SET RDEVICE FEATURE

```
set rdev 7e2 feature key
HCPZRP6722I Characteristics of device 07E2 were set as requested.
1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created

set rdev 7e2 feature key cow duck
HCPZRP6722I Characteristics of device 07E2 were set as requested.
1 RDEV(s) specified; 1 RDEV(s) changed; 0 RDEV(s) created
```

12.7.6 QUERY responses

There is new information available with different QUERY responses that provide encryption-related information to the issuer. This includes the Class B QUERY TAPES DETAILS <*rdev*> and the Class G QUERY VIRTUAL TAPES and QUERY VIRTUAL <*rdev*> DETAILS commands.

All three variations include text that indicates which drives are capable of encryption. The two commands with DETAILS options provide additional information regarding the key labels in use on the tape device. If any encryption settings were defined with the SET RDEVICE FEATURE command, those settings are displayed under the heading INACTIVE KEY LABELS. After an ATTACH command is issued, those same settings will reside under the heading ACTIVE KEY LABELS if ATTACH was performed unmodified. Otherwise, the active heading will contain the encryption information specified on the ATTACH command. If either heading indicates DEFAULT, the default keys defined in the EKM are being used.

Example 12-11 Variations of QUERY output

```
q v tapes
TAPE 0181 ON DEV 07E2 3590 R/W SUBCHANNEL = 0008 ENCRYPTION CAPABLE
Ready; T=0.01/0.01 16:42:02

q v 181 details
TAPE 0181 ON DEV 07E2 3590 R/W SUBCHANNEL = 0008 ENCRYPTION CAPABLE
   ACTIVE KEY LABEL(S):
      (H) the first mighty key label
      (L) the second mighty key label
   INACTIVE KEY LABEL(S): DEFAULT

q tapes details 7e2
TAPE 07E2 SEQUENCE NUMBER E0010 LIBPORT 1 ENCRYPTION CAPABLE
   ACTIVE KEY LABEL(S):
      (H) the first mighty key label
      (L) the second mighty key label
   INACTIVE KEY LABEL(S): DEFAULT

q tapes details 7e3
TAPE 07E2 SEQUENCE NUMBER E0010 LIBPORT 1 ENCRYPTION CAPABLE
   INACTIVE KEY LABEL(S):
      (H) the first mighty key label
      (L) the second mighty key label
```

12.7.7 z/VM DASD Dump Restore (DDR)

The *z/VM DASD Dump Restore (DDR)* utility also supports tape data encryption for TS1120 and TS1130 drives, but it utilizes its own unique externals in order to run in the absence of an underlying z/VM system. The Input/Output control statement includes a new option, *KEY*, that will cause the output device to be enabled for device encryption. This option is only valid on the output statement, because it has no meaning on an input device. A new HASH/LABEL control statement has been added to define the encoding mechanism used on the tape. These statements (up to two can be specified) determine what key labels are to be used for encrypting the tape, and how they will be used (as a label or as a Hash of the public key). These control statements are optional, and if not included, the default keys in the EKM will be utilized for encryption. Example 12-12 dumps all data from the volume labeled `SYSRES` onto the tape mounted on unit `181` and the data is encrypted using key `e3rw33rssesyqypsftqqpx0539`. Only one key label is specified in Example 12-12.

Example 12-12 DDR control statements

```
input 191 3390 sysres
label1 e3rw33rssesyqypsftqqpx0539
output 181 3592 (key
dump all
```

12.8 Miscellaneous implementation considerations

This section provides additional hints and tips for tape data encryption in a z/OS environment.

12.8.1 Data exchange with other data centers or business partners

In order to share tapes with data centers, other organizations, or contracting services, or for other purposes, EKM can store two sets of wrapped encryption keys on the tape. This capability allows another organization to read that specific tape without your providing to 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 EKM's keystore as a second alias (or key label).

When the tape is written, the encryption keys are stored on the tape in multiple places and protected by two sets of public-private keys, yours and the other organization's. The other organization is then able to use their EKM and their public-private certificate and private key to unwrap the data key, which allows them to read that specific tape. This gives you the flexibility to make a specific tape readable by both your own organization and another organization. If you want to take advantage of this capability, you must add that other organization's certificate and public key to your keystore.

In other words, if you want to send an encrypted tape cartridge to another data center for data access, two key labels must be used with the second key label using an encoding method of HASH. This method enables you to use a key label that is different than the receiving organization's key label for the same key.

12.8.2 Availability

When keystores are lost or damaged, there is no way to decrypt the encrypted data on tapes. Therefore, it is absolutely necessary to back up the keystore data sets.

If hardware-based keystores are used and the backup keystore data set's data is used in another data center, the same hardware equipment has to be available (IBM Integrated Cryptographic Service Facility (ICSF), plus hardware facilities.

If a hardware-based keystore is not being used, these backup keystores are protected, for example, by RACF profiles. Later, they can be restored or used at another data center. We recommend that you use utilities from RACF for backup purposes, for example, IRRUT400 to create snapshots of the database either for the primary or backup database with parameter NOLOCKINPUT. The output of this utility can be SMS-managed, which means the output can be stored or copied using normal backup programs and utilities, such as DFSMSdssm, DFSMShsm, and others.

12.9 TS1120 and TS1130 tape cartridge rekeying in z/OS

Rekeying is the process of changing the encrypted data keys on the TS1120 and TS1130 cartridge so that only the new keys can be used to read the data on the cartridge. You might want to do this in these situations:

- The existing keys have been compromised, and you want to prevent access to this cartridge with the compromised keys.
- You want to send this same cartridge to another department or business partner that can only understand keys that are different from those that currently exist on the cartridge.

Rekeying does not require that all of the data is read back in and then rewritten using a different set of keys. It is a much quicker process than having to copy the data.

This section describes the IEHINITT changes in support of the TS1120 model E05 and TS1130 model E06 rekeying capability, which is available on z/OS V1R6 and later; refer to APAR OA20076.

12.9.1 TS1120 Model E05 rekeying support in z/OS

When the TS1120 Model E05 encryption support was first delivered, as part of that solution the drive supported a function referred to as rekeying. At that time, support for this function was not available in z/OS and was targeted as a post-GA item. This section describes the support that was added in z/OS to enable the rekeying capability in the drive.

When a tape cartridge is encrypted, the Data Key (DK) used to encrypt the data is also encrypted, or wrapped, using the public key from a public-private key encrypting key (KEK) pair. This creates an Externally Encrypted Data Key (EEDK) that is stored on the tape cartridge in non-user data areas of the tape. The private key of that public-private key pair is then used to decrypt the data key.

When the first file sequence on a tape is written and encryption is requested, up to two key labels can be specified enabling the cartridge's data key (DK) to be encrypted or wrapped with two different public keys. This generates an EEDK for each of the key labels specified. One of the key labels can be used for local (on-site) usage and the second key label can be used for export (off-site) purposes. Allowing two key labels to be specified works well if it is known ahead of time what the usage of the tape will be, but what if the volume has to be exported, at a later date, using a different public key or the existing keys have been compromised? A mechanism is provided in the drive and the Encryption Key Manager (EKM) that enables a cartridge's data key (DK) to be re-encrypted with new key encrypting keys (using new key labels). This enables a tape cartridge to be rekeyed without having to copy the data to another volume. This is accomplished through the drive and the external key manager (EKM) by decrypting one of the existing EEDKs stored on tape to get the DK, re-encrypting it with different KEKs to create new EEDKs, and overwriting the existing EEDKs with the new EEDKs.

12.9.2 IEHINITT enhancements

The tape encryption rekeying support in z/OS is provided as an extension of the existing IEHINITT (tape initialization) utility. The rekey (REKEY) support takes as input a volume serial number and its new key labels and encoding mechanism. Two key labels can be specified; however, it requires that at least one key label is specified. The same value can be specified for both key labels. If only one key label is specified, the same value is used for the other key label. The new key labels can be the same key labels that were used to generate the existing EEDK structures on the tape. No checking is performed to determine whether the new key labels are the same key labels that were originally used.

When NUMBTAPE is specified, the volume serial number of the first volume is specified and is increased by one for each additional volume (maximum serial number is 999999). The serial number specified must be all numeric.

As with the existing INITT function, the new REKEY function uses as input a control dataset that contains the utility control statements and produces on output a dataset that contains:

- Utility program identification
- New key label information for each tape volume that is successfully rekeyed
- Message codes and error messages

Example 12-13 on page 418 shows the sample output of an IEHINITT REKEY operation.

Example 12-13 Sample IEHINITT output

```
SYSTEM SUPPORT UTILITIES                  IEHINITT
LABEL   REKEY SER=001000,KEYLABL1=First_Key_Label,KEYENCD1=L,
              KEYLABL2=Second_Key_Label,KEYENCD2=H,NUMBTAPE=2

001000 KEYLABL1=First_Key_Label            KEYENCD1=L
       KEYLABL2=Second_Key_Label           KEYENCD2=H

001001 KEYLABL1=First_Key_Label            KEYENCD1=L
       KEYLABL2=Second_Key_Label           KEYENCD2=H
```

The new REKEY control statement syntax is:

```
ddname    REKEY   SER=VOLSER
                  [,DISP=[rewind|unload}]
                  [,NUMBTAPE={n|1}}]
                  [,KEYLABL1=keylabel1 (64-char)]
                  [,KEYENCD1={L|H}]  (L=Label;H=Hash)
                  [,KEYLABL2=keylabel2 (64-char)]
                  [,KEYENCD2={L|H}]  (L=Label;H=Hash)
```

Note that all other INITT-related keywords are invalid. The new fields are in bold type.

The parameters are:

- SER=VOLSER

  The SER keyword can be specified in two forms:

SER=VOLSER	Specifies the volume serial number of one volume.
SER=VOLSER,NUMBTAPE=*n*	When NUMBTAPE is specified, SER is the volume serial number of the first volume and is increased by one for each additional volume (maximum serial number is 999999). The serial number specified must be all numeric.

- KEYLABL1=*keylabel1_name*

  KEYLABEL1 specifies the first key label for the key encryption key used by the EKM. A key label is from 1 to 64 characters with blanks padding the field on the right. A key label contains alphanumeric, national (special symbols used in specific languages), or special characters with certain additional characters also allowed. Enclose it in single quotation marks if it contains any blanks or special characters. IEHINITT does not validate the character set specified for the key label keyword.

  At least one key label and its encoding mechanism are required for REKEY. When you specify this operand, you must also specify a value for the key encoding mechanism using the KEYENCD1.

  Note that if only one key label and its associated encoding mechanism are specified, the same key label and mechanism values will be used for both key labels and mechanisms.

- KEYENCD1={L|H}

  Specifies the encoding mechanism of the first key label KEYLABL1. The encoding mechanism indicates how the label for the key encrypting key specified by the key label is to be encoded by the EKM and stored on the tape:

  L Label, which is encoded as the specified label
  H Hash, which is encoded as a Hash of the public key referenced by the specified key label

- KEYLABL2=*keylabel2_name*

  Specifies the second key label for the key encryption key used by the EKM. A key label is from 1 to 64 characters with blanks padding the field on the right. A key label contains alphanumeric, national, or special characters with some additional characters also allowed. Enclose the key label in single quotation marks if it contains any blanks or special characters. IEHINITT does not validate the character set specified for the key label keyword.

  At least one key label and its associated encoding mechanism are required for REKEY. When you specify this operand, you must also specify a value for the key encoding mechanism using the KEYENCD2. Note that if only one key label and its associated encoding mechanism are specified, the same key label and mechanism values will be used for both key labels and mechanisms.

- KEYENCD2={L|H}

  Specifies the encoding mechanism of the second key label KEYLABL2. The encoding mechanism indicates how the label for the key encrypting key specified by the key label is to be encoded by the EKM and stored on the tape:

  L Label, which is encoded as the specified label
  H Hash, which is encoded as a Hash of the public key referenced by the specified key label

- NUMBTAPE={*n*|1}

  Specifies the number of tapes to be rekeyed. The value *n* represents a number from 1 to 255. If more than one tape is specified, the volume serial number of the first tape must be all numeric.

Example 1: REKEY one tape volume

Example 12-14 shows one tape volume, SL1000, to be rekeyed with two key labels specified.

Example 12-14 Rekey one tape volume

```
//TAPEJOB   JOB   …
//STEP1     EXEC  PGM=IEHINITT
//SYSPRINT  DD    SYSOUT=A
//REKEY1    DD    UNIT=(TAPE,1,DEFER)
//SYSIN     DD    *
REKEY1      REKEY SER=SL1000,
                  KEYLABL1=firstkeylabel,KEYENCD1=L,
                  KEYLABL2=secondkeylabel,KEYENCD2=H

/*
```

Example 2: Multiple REKEY control statements

Example 12-15 on page 420 shows two tape volumes, SL8000 and SL8001, to be rekeyed. Each tape volume is to be rekeyed with a different set of key labels and encoding mechanisms.

Example 12-15 Multiple REKEY control statements

```
//TAPEJOB   JOB…
//STEP1     EXEC  PGM=IEHINITT
//SYSPRINT  DD    SYSOUT=A
//REKEY1    DD    UNIT=(TAPE,1,DEFER)
//REKEY2    DD    UNIT=(TAPE,1,DEFER)
//SYSIN     DD    *
REKEY1      REKEY SER=SL8000,
                  KEYLABL1=firstkeylabel,KEYENCD1=L,
                  KEYLABL2=secondkeylabel,KEYENCD2=H
REKEY2      REKEY SER=SL8001,
                  KEYLABL1=differentfirstkeylabel,KEYENCD1=L,
                  KEYLABL2=differentsecondkeylabel,KEYENCD2=H
/*
```

12.9.3 Security considerations

For rekeying, the following security considerations apply:

- IEHINITT currently provides volume level security checking using SAF/RACF but does not invoke data set level checking. If you do not use SAF/RACF TAPEVOL profiles, extra precautions might be required when using IEHINITT for the REKEY function.
- For the REKEY function, SAF/RACF will be invoked for authorization checking in both library and non-library environments. The level of authority required to proceed with the rekeying processing is UPDATE for CLASS=TAPEVOL.
- The rekey utility issues a RACROUTE call after the mounted tape volume label is read. The VOLSER that was read or from the sense bytes is passed to the RACROUTE call. If the tape does not have a VOLSER (NL tape), the requested VOLSER will be used as an internal VOLSER in all SAF/RACF checking. UPDATE access to the volume is required to rekey the volume, or the volume must not be protected; *not protected* means that either there is no TAPEVOL profile or the TAPEVOL class is not active.
- The level of authority in hierarchical order is:
  - READ
  - UPDATE
  - CONTROL
  - ALTER

  If a user has UPDATE authority, and ATTR=READ is specified, RACF returns a return code of 0 (zero). If ATTR=CONTROL, RACF returns a return code of 8.

12.9.4 Packaging

Support for the new tape enhancements is available on z/OS V1R6 and higher. TS1120 tape data encryption Rekeying was generally available on 08/31/2007 (refer to APAR OA20076).

12.9.5 Rekeying exits and messages

Detailed information about the new IEHINITT Dynamic Exits provided with the Rekeying support and information about the new and modified IEH messages are provided in Appendix E, “IEHINITT exits and messages for rekeying” on page 631.

13

Implementing TS7700 Tape Encryption

The TS7700 Virtualization Engine implements tape encryption for the physical back-end tape drives through storage pools. You can use up to 32 storage pools where each storage pool can hold non-encrypted cartridges or can hold encrypted cartridges, which were encrypted with the same or different keys.

13.1 TS7700 Encryption overview

TS7700 Tape Encryption uses System-Managed Encryption (SME). DFSMS constructs are assigned by automatic class selection (ACS) routines by selecting a Storage Group and Management Class for the logical volume. Then, the TS7700 has matching constructs that determine the storage pool where the logical volume will reside and then whether physical volumes in that storage pool will be encrypted.

Figure 13-1 graphically illustrates the connections between the DFSMS constructs and the TS7700 constructs. TS7700 Encryption is implemented outboard within the storage pools of the TS7700 but is indirectly controlled within DFSMS by your selection of the Storage Group and the Management Class for the data.

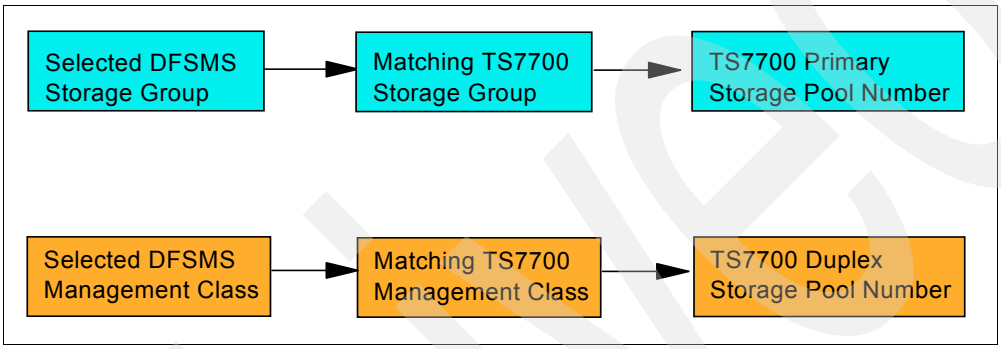


Figure 13-1 Invoking encryption in the TS7700

In this chapter, we describe the implementation steps in z/OS that are required to encrypt some or all of the physical stacked TS1120 volumes used by the TS7700 Virtualization Engine. These implementation steps might include the new installation of:

- IBM TS3500 Tape Library or IBM 3494 Tape Library
- IBM TS7700 Virtualization Engine
- IBM TS1120 Tape Drives

Or, the installation might just include firmware upgrades of the existing tape libraries, tape drives, and TS7700 along with the installation of the required software support.

We describe the following topics:

- Prerequisites
- Tape library implementation and setup
- Software implementation tasks in z/OS
- Basic SMS implementation steps
- TS7700 implementation and setup

All prerequisites necessary for hardware (microcode levels for the tape drives, the tape libraries, and the TS7700) and software for the various operating systems are discussed in Chapter 4, “Planning for software and hardware” on page 91.

13.2 Prerequisites

The prerequisites for back-end tape drive encryption within the TS7700 are:

- An encryption-capable TS1120 drive that has been enabled for encryption with the SME method
- A TS7700 at microcode levels that support encryption and with feature code (FC) 9900 installed
- A 3953 or 3494 Library Manager at microcode level 534.31 or higher
- An Encryption Key Manager (EKM) running, available, and communicating with the TS7700
- DFSMS Storage Group and Management Class constructs available that will be used to enable encryption policies at the TS7700
- Matching Storage Groups and Management Classes in the TS7700 that select TS7700 storage pools that will be encrypted
- TS7700 storage pool definitions that specify Encryption and the key labels to be used

13.2.1 Tape drives

Because data encryption is performed on the tape drives, TS1120 Model E05 encryption-capable tape drives must be attached to the TS7700. They also must be running in native (E05) mode rather than J1A Emulation mode. All TS1120 tape drives with FC5592 or FC9592 are encryption-capable. These drives must also be enabled for SME before they can be used for encryption with the TS7700.

13.2.2 TS7700 Virtualization Engine

The TS7700 must be running microcode level 8.2.0.19 or higher (8.2.0.27 or higher if the drives are in a 3494 library). FC9900 must be installed to access the encryption settings.

The TS7700 must not be configured to force the TS1120 drives into J1A mode. This setting can only be changed by your service support representative (SSR). If you have to update the microcode level, be sure the SSR checks and changes this setting if required.

13.2.3 Library Manager

The 3953 Library Manager for the TS3500 or the 3494 Library Manager requires microcode level 534.31 or higher.

13.2.4 Encryption Key Manager

Your Encryption Key Managers (EKMs) must be installed, configured, and operational before installing the encryption feature on the TS7700. You must also create the Key Encrypting Key (KEK) certificates that you plan to use for encrypting your backstore tape cartridges.

Refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418, for full details about installing and configuring your EKMs. Also, we discuss planning for the EKM and implementing the EKM in other chapters within this book.

13.3 Implementation overview

Implementation steps to implement the tape library and the TS7700 Virtualization Engine and then to enable and use tape data encryption within the TS7700 can be divided into four parts:

1. Implementation of the tape library and TS1120 tape drives

   An overview of the initial library implementation steps is provided in 13.3.1, “Initial Tape Library hardware implementation” on page 424.

2. Implementation of the TS7700 Virtualization Engine

   An overview of the initial TS7700 implementation steps is provided in 13.3.2, “Initial TS7700 implementation” on page 424.

3. Encryption Key Manager implementation

   An overview of the EKM implementation steps in z/OS is provided in 13.3.4, “EKM implementation overview” on page 425. The complete setup steps for implementing the EKM in z/OS are described in 6.1, “Implementing the EKM in z/OS” on page 160.

4. Setup of tape data encryption in the hardware and software

   The steps to set up tape data encryption in the library is provided in 13.4, “Tape library implementation and setup for encryption” on page 425.

   The z/OS software-related implementation steps are described in 13.5, “Software implementation steps” on page 430 and 13.5.3, “Basic z/OS DFSMS implementation steps” on page 431.

Finally, the TS7700 implementation steps are described in 13.6, “TS7700 implementation steps” on page 432.

13.3.1 Initial Tape Library hardware implementation

A prerequisite for the TS7700 Virtualization Engine is either an IBM TS3500 Tape Library or an IBM 3494 Tape Library. If you do not have either of these IBM system-managed tape libraries installed already, we refer you to these IBM Redbooks publications for a detailed discussion of all implementation and migration steps for these tape libraries:

- For TS3500:

  IBM TS3500 for System z attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation, SG24-6789

- For IBM 3494:

  IBM TotalStorage 3494 Tape Library: A Practical Guide to Enterprise-Class Tape Drives and Tape Automation, SG24-4632

13.3.2 Initial TS7700 implementation

If you do not have an IBM TS7700 installed already, refer to *IBM Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312, which has a detailed discussion of all implementation and migration steps for the TS7700. In this publication, we discuss only the additional steps necessary to implement encryption on TS7700.

Note: The Hardware Configuration Definition (HCD) is the same for the TS7700 whether you are using encryption or not.

13.3.3 Initial z/OS software definitions

For a brief overview of the z/OS software tasks for implementing an IBM tape library and an IBM Virtualization solution (TS7700 or IBM 3494 B10 or B20 VTS), refer to 13.5, “Software implementation steps” on page 430. For detailed information about the basic software setup steps, refer to *IBM Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312.

13.3.4 EKM implementation overview

In a z/OS environment, you must use SME. Before you will be able to encrypt any tape within the TS7700, you must implement the EKM software on a platform. We expect that most z/OS clients will implement at least one instance of the EKM on the z/OS platform; however, your requirements might involve EKM implementations on other platforms.

The complete setup steps for implementing the EKM on each platform are described in Chapter 6, “Implementing EKM” on page 159.

The following steps give a brief overview of the EKM implementation steps in z/OS:

1. UNIX System Services as a prerequisite for Java is already included with the z/OS operating system. Verify whether the installed version of Java is at the appropriate level for the EKM component.
2. Install the EKM after downloading the newest versions available.
3. Obtain the required security permissions for the UNIX System Services segment.
4. Create a keystore for EKM.
5. Set up the EKM environment in Java.
6. Start the EKM.
7. Set up the EKM at the TS7700.

13.4 Tape library implementation and setup for encryption

The TS1120 tape drives that are to be used by the TS7700 for encryption must be encryption-capable as well as encryption-enabled. *All TS1120 tape drives with FC5592 or FC9592 are encryption-capable.* The drives also must be running in native (E05) mode rather than J1A Emulation mode.

If you have 3592 Model J1A drives attached to the TS7700, they must be detached. The TS7700 does not allow a mixture of drive types to be used. The J1A drives can be redeployed in other subsystems or used as direct-attached drives for Open Systems hosts. If you have a mixture of J1A and TS1120 drives attached to your TS7700 and cannot detach the J1A drives right away, you can proceed as long as you have a minimum of four encryption-capable TS1120 drives attached. Be aware, though, that the J1A drives will not be used by the TS7700 after the TS1120 drives are put into native E05 mode.

Enabling the TS1120 drives for encryption is handled differently depending on whether the drives are installed in an IBM TS3500 Tape Library or in an IBM 3494 Tape Library. We discuss both tape libraries in the following sections.

13.4.1 Enabling drives for encryption in the IBM TS3500 Tape Library

If the TS1120 drives are installed or will be installed in an IBM TS3500 Tape Library, the TS3500 Tape Library Specialist has the ability to enable the TS1120 drives for encryption. To update the tape drive definitions, use the Web browser interface of the TS3500, the TS3500 Tape Library Specialist. To begin this procedure, you should have the World Wide Node Names (WWNNs) for the drives that will be attached to this TS7700. You can get these names from the Enterprise Automated Library Specialist by selecting **Monitor library manager** → **Physical Device Location**, which displays the physical location of the drives and their ownership.

Note: Your TS7700 must be *offline* during the remainder of this procedure.

Figure 13-2 shows the Welcome panel of the TS3500 Specialist.

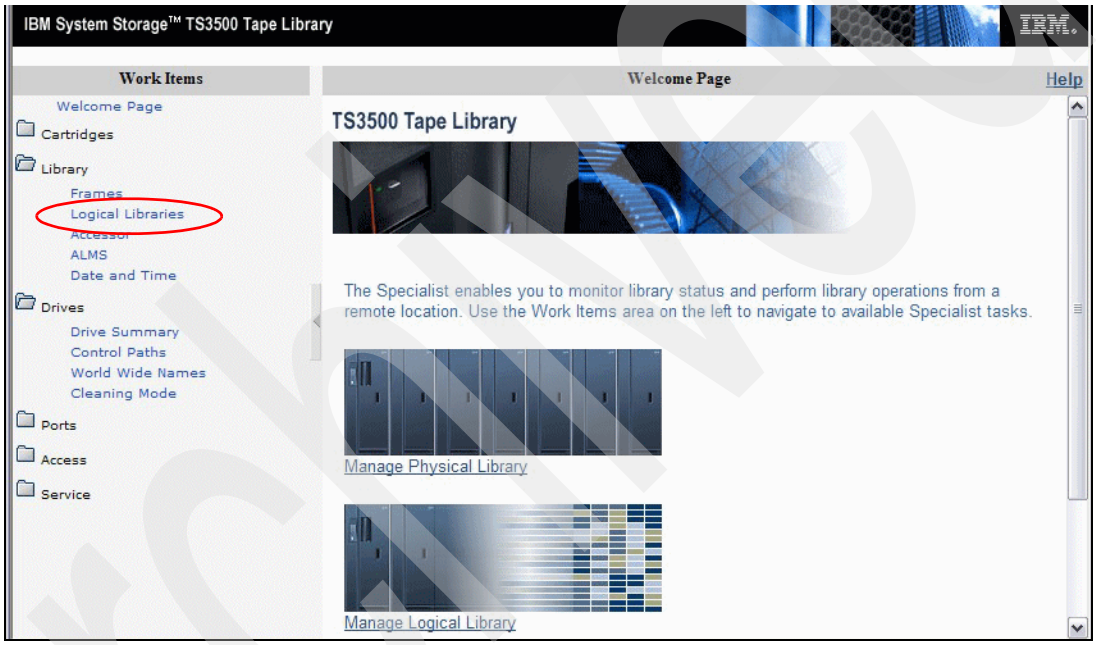


Figure 13-2 Entry or welcome panel of TS3500 Tape Library

To enable encryption on the TS1120 drives that will be attached to the TS7700:

1. On the left side of the panel, select **Library** → **Logical Libraries,** which leads you to the next panel called Manage Logical Libraries, which is shown in Figure 13-3 on page 427.

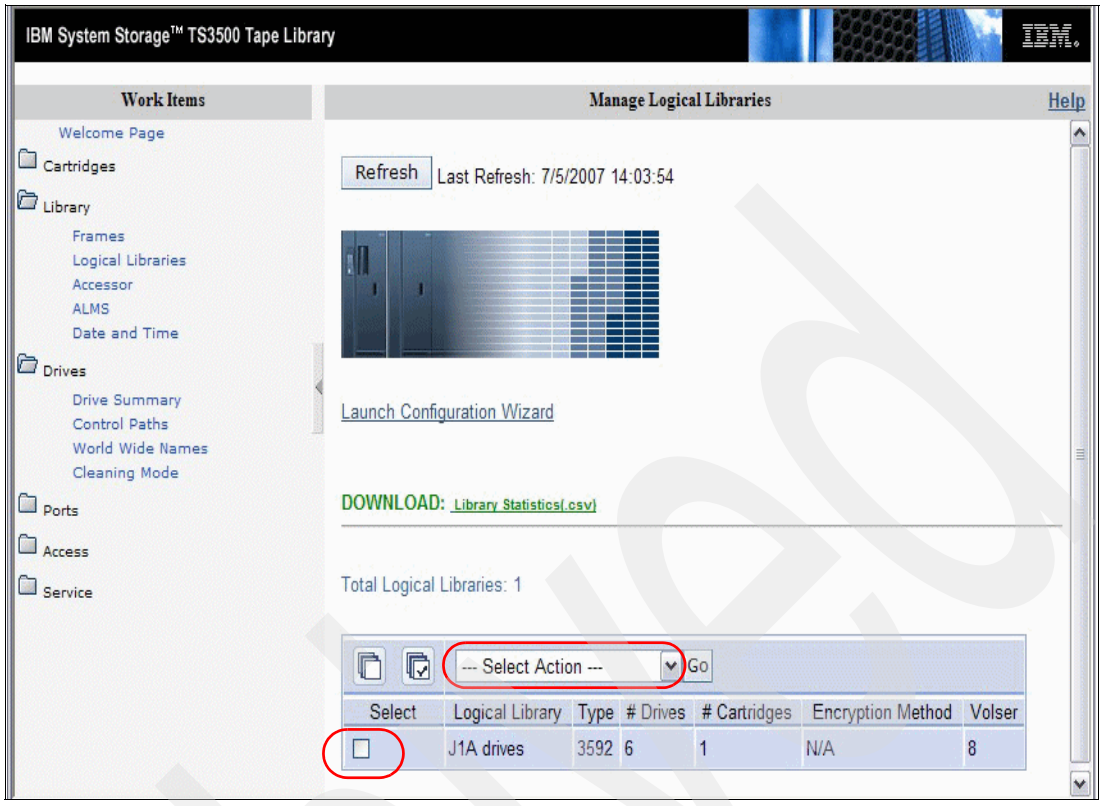


Figure 13-3 Manage Logical Libraries panel

2. Check the logical library that is connected to the 3953 Library Manager and the TS7700. Then, choose **Select Action** → **Modify Encryption Method**, and then click **Go**. This step assumes that you have already assigned the TS1120 drives to this logical library. The panel shown in Figure 13-4 displays.

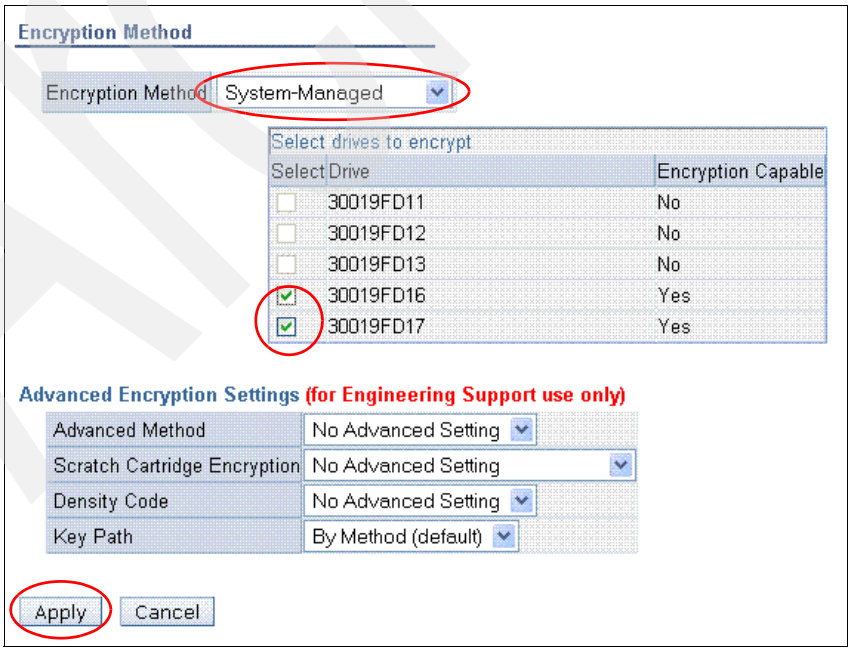


Figure 13-4 Panel to mark tape drives used for encryption

3. In this panel, select the Encryption Method **System-Managed**. Next, check all of the encryption-capable drives to be used by the TS7700, and click **Apply**. If the drives that you want to use with the TS7700 do not **Yes** in the Encryption Capable column, encryption enablement is not set and you will have to determine why they are not encryption-capable.
4. Review the panel that contains the following message and then click **OK**:

   ```
   Once the encryption change is complete, the associated host application(s)
   require a reset or rediscovery of the devices. Do you want to continue with the
   encryption change?
   Click OK to continue.
   ```

5. Review the panel that has the following message and then click **Close**:

   ```
   Drive Encryption change request has completed.
   ```

You have enabled the drives for System-Managed Encryption.

13.4.2 Enabling drives for encryption in the IBM 3494 Tape Library

Prior to microcode level 535, if the encryption-capable TS1120 drives are installed or will be installed in a 3494 tape library, Encryption Configuration FC9596 (plant-installed) or FC5596 (field-installed and chargeable) must be ordered on the TS1120 drives to enable the encryption method on the drives. This feature code provides instructions and procedures for the SSR to enable the TS120 drives for encryption and to specify the encryption mode. The drives must be configured by the IBM SSR for the System-Managed Encryption method.

With the introduction of the 535 microcode level on the 3494 tape library, you now have the capability to enable encryption on the TS1120 drives through the ETL Web Specialist or User Interface of the 3494 Library Manager Console. The ETL Specialist is similar to the procedures just discussed for the TS3500, except that there is only one logical library on the 3494. We will discuss the procedures for the 3494 User Interface in detail next.

Enabling Encryption for the TS1120 drives using the 3494 User Interface

Note that as of the writing of this publication, the capabilities in microcode level 535 were still being developed, so the panels shown here might not be the final ones delivered but will be quite similar.

Only perform this procedure when the TS7700 is offline:

1. On the main 3494 Library Manager panel, select **Commands** → **System management** → **Manage encryption** as shown in Figure 13-5 on page 429.

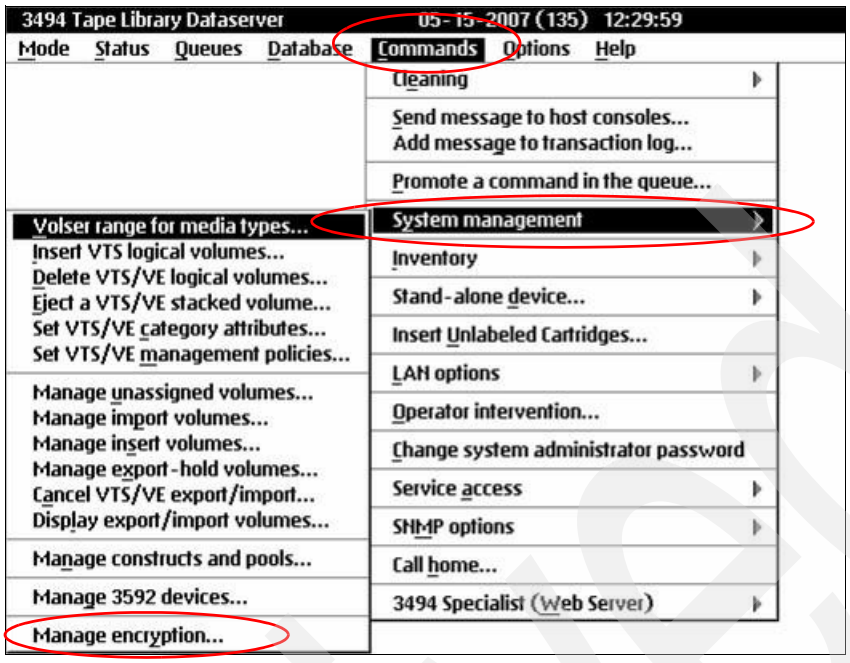


Figure 13-5 3494 User Interface main panel

This selection displays the Manage Encryption panel, as shown in Figure 13-6.

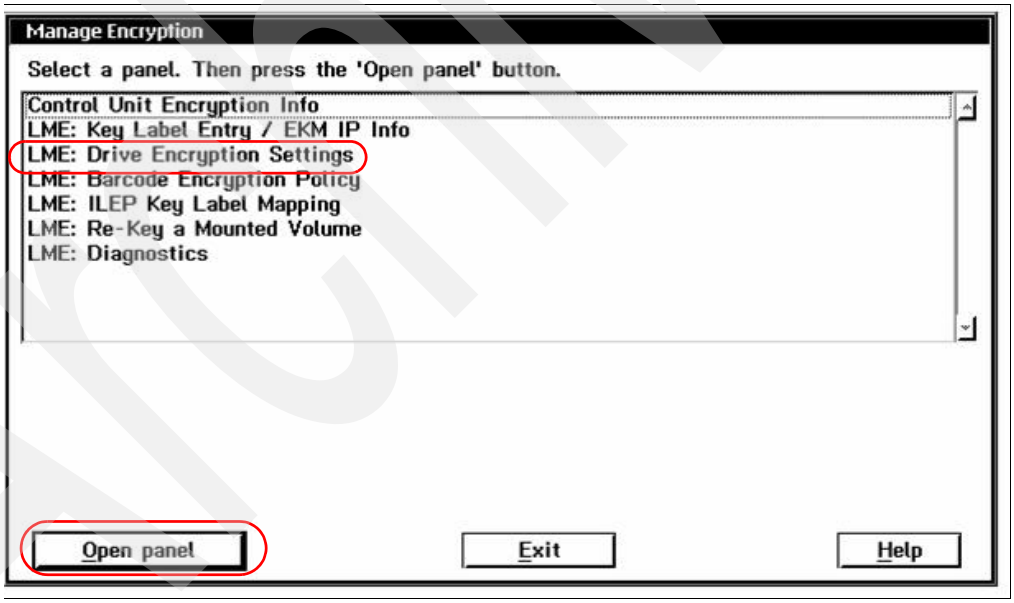


Figure 13-6 Manage Encryption panel

2. Highlight the row **LME: Drive Encryption Settings** (or **SME: Drive Encryption Settings** if present) and then click **Open panel**.

   This step displays the Drive Encryption Settings (Set VPD) panel shown in Figure 13-7 on page 430.

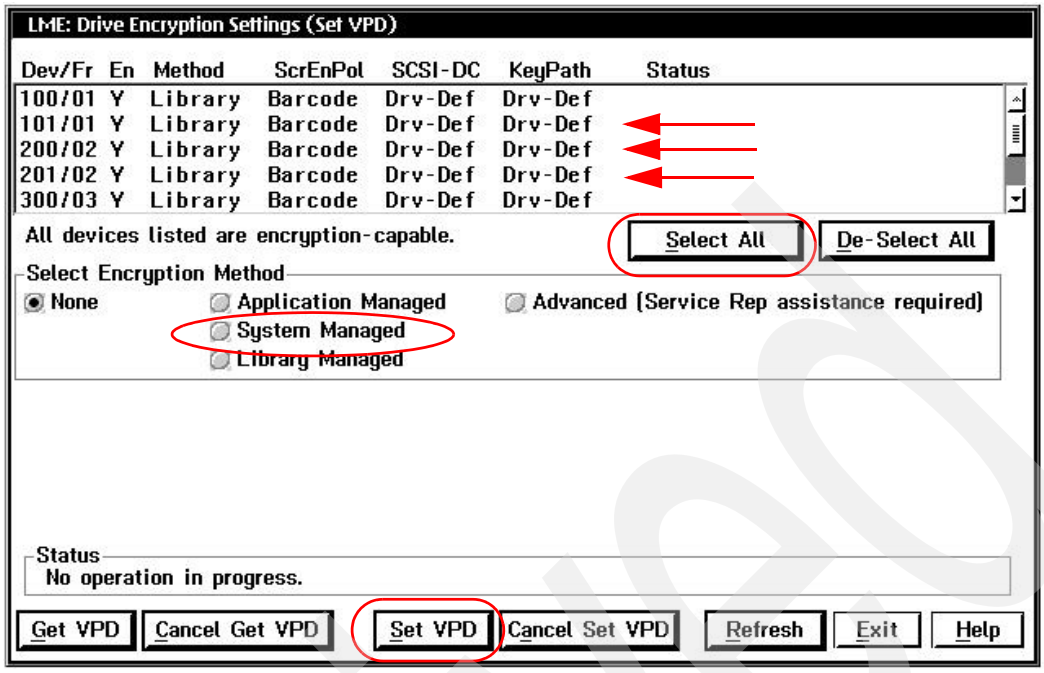


Figure 13-7 Drive Encryption Settings (Set VPD)

3. Highlight all drives to be used by the TS7700 (or click **Select All**), click **System Managed**, and then click **Set VPD**. This will enable System-Managed Encryption on all of these drives, a prerequisite for the TS7700 to be able to use encryption.

13.4.3 Encryption-enabled drives

Note: When the TS7700 is using drives for encrypted physical tape volumes, it will put drives (that are not properly enabled for encryption) offline to the subsystem. Inform the tape library operators that they must leave TS7700-attached drives in SME mode at all times so that the drives are not taken offline.

13.5 Software implementation steps

In this section, we describe the host software implementation steps that are required for TS7700 Tape Encryption.

13.5.1 z/OS software maintenance

As of the writing of this publication, there was no known z/OS software maintenance to support encryption in the TS7700 Virtualization Engine because the implementation of encryption within the TS7700 is:

- Implemented outboard within the TS7700 microcode using Outboard Policy Management
- Invoked with existing DFSMS Storage Group and Management Class constructs

13.5.2 EKM installation

A separate software installation will be required for EKM. The complete setup steps for implementing the EKM on z/OS or other platforms is described in Chapter 6, “Implementing EKM” on page 159.

13.5.3 Basic z/OS DFSMS implementation steps

Encryption for native TS1120 drives is controlled by the DFSMS Data Class specifications. Encryption on the TS7700 is not controlled by DFSMS Data Class specifications. Instead, encryption is controlled on a TS7700 storage pool basis by using Storage Group and Management Class DFSMS constructs. Storage Group and Management Class DFSMS constructs (specified for logical tape volumes) determine, through mapping in the Library Manager, which storage pools are used for the primary copies and secondary copies (if used) of the logical volumes. The TS7700 storage pools, originally created for management of physical media, have been enhanced to include encryption characteristics.

You might want to set up additional Storage Group and Management Class constructs in DFSMS. For detailed information about setting DFSMS policy constructs, refer to the *DFSMS Object Access Method Planning, Installation, and Storage Administration Guide for Tape Libraries*, SC35-0427.

We discuss the setup of Storage Groups and Management Classes for DFSMS in the following sections.

Setting up a Storage Group in DFSMS

You can use ISMF to define your storage groups by selecting option **6**, Storage Group, from the ISMF Primary Option Menu for storage administrators:

1. In the Storage Group Name field, specify the name of the storage group you are defining.
2. In the Storage Group Type field, specify `TAPE`.
3. Select **2** to Define a new Storage Group.

The only other fields that have to be entered on the Tape Storage Group Define panel are the library names. Up to eight library names can be specified. The library names must be the names of TS7700 Virtualization Engine libraries.

The Tape Storage Groups defined in DFSMS must have identically named Storage Groups defined in the TS7700s referenced in the library names. Refer to 13.6, “TS7700 implementation steps” on page 432 for setting up the Storage Groups in your TS7700.

Setting up a Management Class in DFSMS

You can use ISMF to define your Management Classes by selecting option **3**, Management Class, from the ISMF Primary Option Menu for storage administrators:

1. In the Management Class Name field, specify the name of the Management Class that you are defining.
2. Select **3** to define a new Management Class.

You do not have to define any of the Management Class attributes when the Management Class is to be used only for virtual tape.

The Management Classes defined in DFSMS that are to be used for the TS7700 must have identically named Management Classes defined in the TS7700s. Refer to the next section for setting up the Management Classes in your TS7700.

13.6 TS7700 implementation steps

In this section, we assume that you have already set up your TS7700 without encryption. If you do not have an IBM TS7700 installed already, refer to the following publication for a detailed discussion of all TS7700 implementation and migration steps without encryption, *IBM Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312.

In this publication, we only describe the additional steps necessary to implement Encryption on the TS7700.

13.6.1 Configuring the TS7700 for encryption

Configuring the TS7700 to support encryption requires the following steps:

1. Define Storage Groups (matching your DFSMS Storage Group names) that specify the storage pool to be used for the primary copy of the logical volume.
2. Define Management Classes (matching your DFSMS Management Classes) that specify the storage pool to be used for the backup (duplex) copy of the logical volume.
3. Activate the encryption feature license.
4. Set up the EKM TCP/IP addresses.
5. Define whether encryption for a storage pool will be enabled. If the storage pool is enabled, define the key labels and the key modes to be used.

We discuss these steps in detail in the following sections. The procedures will use the TS7700 Management Interface (MI) and the Enterprise Automated Tape Library Specialist (ETL) or the Library Manager User Interface (UI). Most of the tasks can be performed either with the ETL Specialist or the Library Manager panels (UI). In general, we recommend using the ETL Specialist, because it is a more user friendly Web interface. For all of these procedures, you will start by choosing the TS7700 cluster that is to be modified. In our examples, we are using *Administer VTS 1*. The procedures are similar for the IBM TS3500 Tape Library and for the IBM 3494 Tape Library.

For details about using the Library Manager panels, refer to:

- *IBM TotalStorage 3953 Tape Frame Model F05 and Library Manager Model L05 Operator Guide,* GA32-0473
- *IBM TotalStorage 3494 Tape Library Operator Guide*, GA32-0449

The panel shown in Figure 13-8 on page 433 shows the Welcome Page for the Enterprise Automated Tape Library Specialist (ETL).

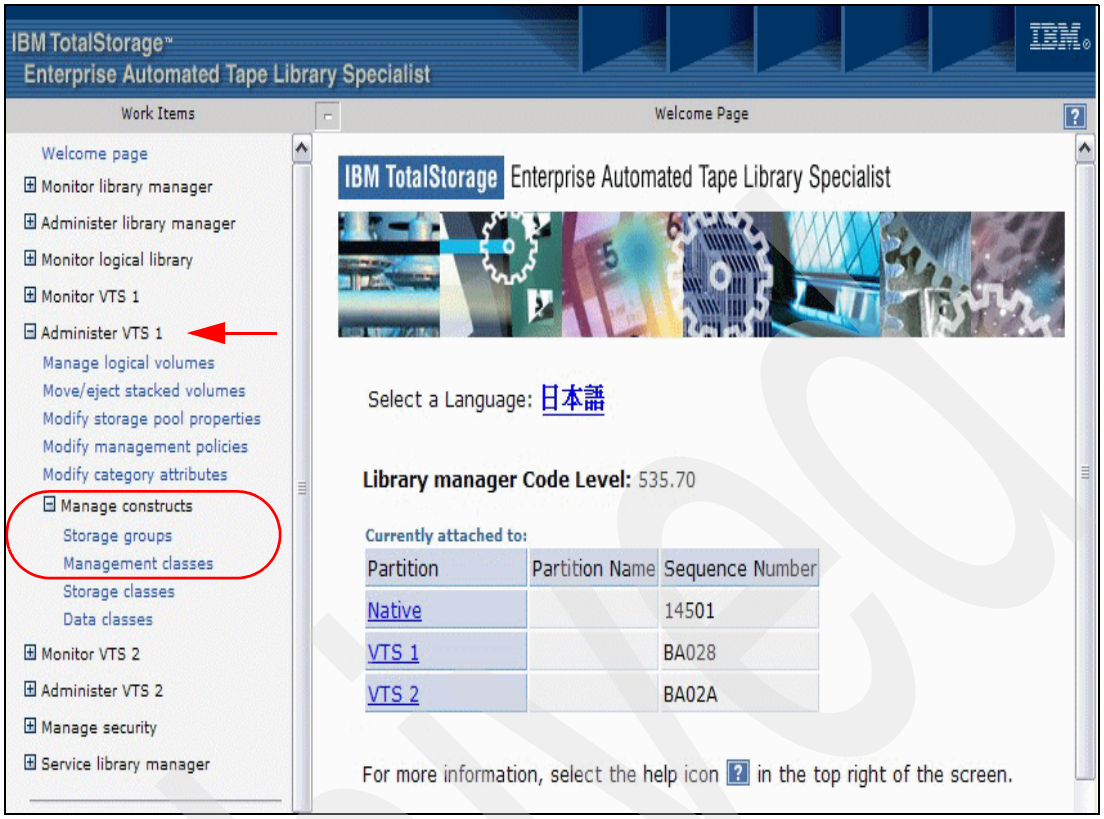


Figure 13-8 Manage Constructs with the ETL Specialist

Note that all ETL Specialist panels that allow modifications of definitions require that you to enter a user ID and password that have the proper authority.

13.6.2 Creating TS7700 Storage Groups

On the z/OS host, the Storage Group construct determines into which tape library a logical volume is written. Within the TS7700, the Storage Group construct allows you to define the storage pool to which you want the logical volume written. Multiple Storage Groups can be associated with any given storage pool.

Remember, this Storage Group name must match a Storage Group name defined in DFSMS. Up to 256 Storage Groups (including the default) can be defined.

Even before you define the first TS7700 Storage Group, there is always at least one Storage Group present: the default Storage Group that is identified by eight dashes (--------). Although the Storage Group cannot be deleted, you may modify it, for example to point to a different default storage pool.

Note: If you assign a Storage Group at the host to a logical volume and if this Storage Group has not been defined on the Library Manager before, this Storage Group will be created on the Library Manager using the specifications of the default Storage Group.

To define a new Storage Group, or modify or delete an existing Storage Group, select **Administer VTS *n*** → **Manage constructs** → **Storage Groups.** Figure 13-9 on page 434 shows the Storage Groups definition panel of the ETL Specialist.

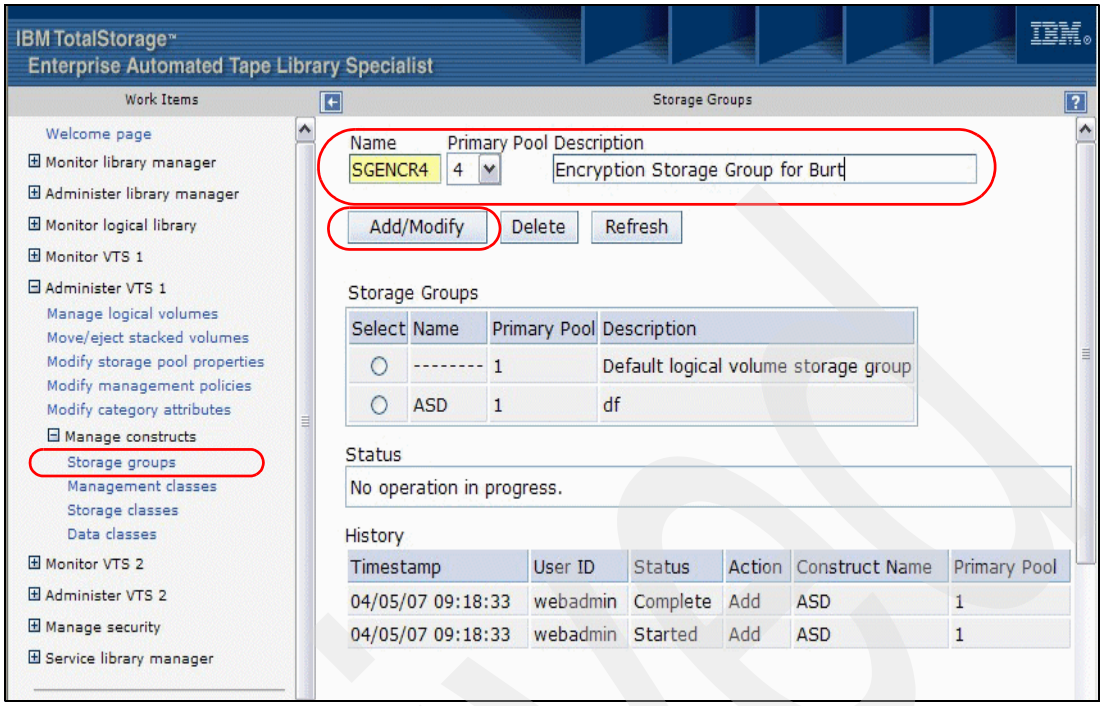


Figure 13-9 Manage Storage Groups with the ETL Specialist

To create a Storage Group:

1. Enter a 1-to-8 character alphanumeric name in the Name field. This name must be unique within the Storage Group construct names. Use your host-defined DFSMS Storage Group name.
2. Enter the number of the pool you want to use for this Storage Group in the Primary Pool field. You can choose from any of the 32 storage pools, but you cannot enter 0 for the Common Scratch Pool. The default Primary Pool is Pool 1.
3. Optionally, enter a short description in the Description field.
4. Click **Add/Modify**.

To modify a Storage Group:

1. Select from the current Storage Groups presented in the list box. Use the mouse or keyboard to highlight the Storage Group that you want to modify.
2. Modify the primary pool and description.
3. Click **Add/Modify**.

The purpose of the History Table at the bottom of the ETL Specialist panel shown in Figure 13-9 is to:

- Indicate actions that are currently in progress or have already completed.
- Coordinate remote users (3953 Specialist and Library Manager Console operator).
- Notify the current user if another user has performed the same kind of action while the current user is preparing to perform the same or a similar action.

Tips:

- In an environment with multiple pools in use, we suggest not to assign a Storage Group to Pool 1. If you then encounter physical cartridges and logical volumes in Pool 1, you have an indication that your Storage Group definitions on the host and on the Library Manager might not match.
- The TS7700 can use up to 32 storage pools. However, keep the number of storage pools used to the minimum necessary to meet your requirements. Excessive storage pools in use can have a performance impact on the TS7700, because cartridges from the various pools might have to be mounted and demounted more often to write data to the pools. Storage pools that are defined, but not used, do not have an impact.

13.6.3 Creating TS7700 Management Classes

You can define, through the Management Class, whether you want to have a dual copy of a logical volume within the same TS7700. In a Grid configuration, you will most likely choose to copy logical volumes over to the other TS7700 instead of creating a second copy in the same TS7700. In a Single Cluster configuration, however, you might want to protect against media failures by using the dual copy capability.

If you want to have dual copies of selected logical volumes, you must use at least two storage pools, because the copies cannot be written to the same storage pool as the original logical volumes. The Management Class determines the storage pool for the second copy. Up to 256 Management Classes (including the default) can be defined.

A Default Management Class is always available. It is identified by eight dashes (--------) and cannot be deleted. If you assign a Management Class at the host to a logical volume, and if this Management Class has not been defined on the Library Manager before, it will be created on the Library Manager using the specifications of the Default Management Class.

To define a new Management Class, or modify or delete an existing Management Class, select **Administer VTS *n*** → **Manage constructs** → **Management Class.** Figure 13-10 on page 436 shows the Management Classes panel of the ETL Specialist.

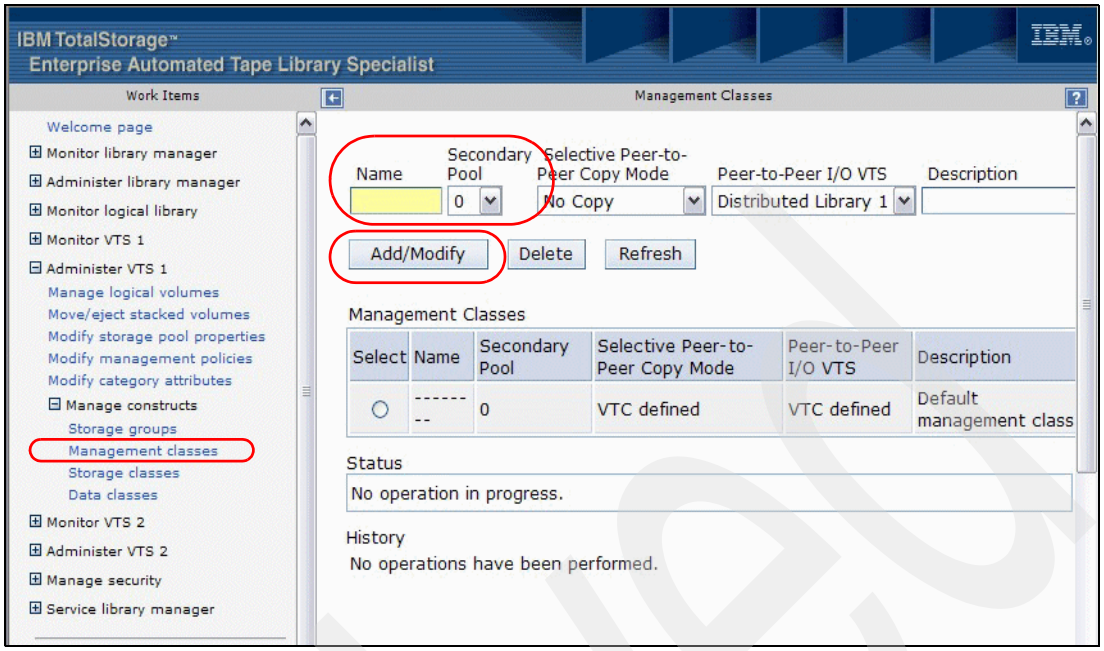


Figure 13-10 Management Classes panel

To add a Management Class:

1. Enter a 1- to-8 character name in the Name field. This name must be unique in the Management Class construct names.
2. Specify the number of a pool in the Secondary Pool field. Select one:
   - Specify the secondary pool number (1-32). This determines in which physical pool the dual copy of the logical volumes will reside.
   - If '**00**' is selected, no secondary copy will be made.
3. Enter a short description in the Description field.
4. Click **Add/Modify**.

Important: The settings for PTP Copy Control and PTP I/O VTS only apply for IBM TotalStorage Peer-to-Peer Virtual Tape Servers (PtP VTS).

For implementation of an IBM TS7700 Virtualization Engine, leave the "VTC defined" defaults in those fields.

To modify a Management Class:

1. Select a Management Class from the current Management Classes presented in the list box. Use the mouse or keyboard to highlight the Management Class that you want to modify. You can modify pools and description.
2. Click **Add/Modify**.

To delete a Management Class:

1. Select a class from the current Management Classes presented in the list box. Use the mouse or keyboard to highlight the Management Class that you want to delete.
2. Click **Delete**, and then confirm that you wanted to delete this Management Class.

13.6.4 Activate the TS7700 Encryption Feature License

Both this section and the following sections require that you invoke the TS7700 Management Interface (MI). Click **Manage Virtualization Engine *n*** near the bottom of the left panel to invoke MI. Alternatively, you can just enter the TCP/IP address of the TS7700 Console interface to go directly to the MI. A panel similar to Figure 13-11 opens.

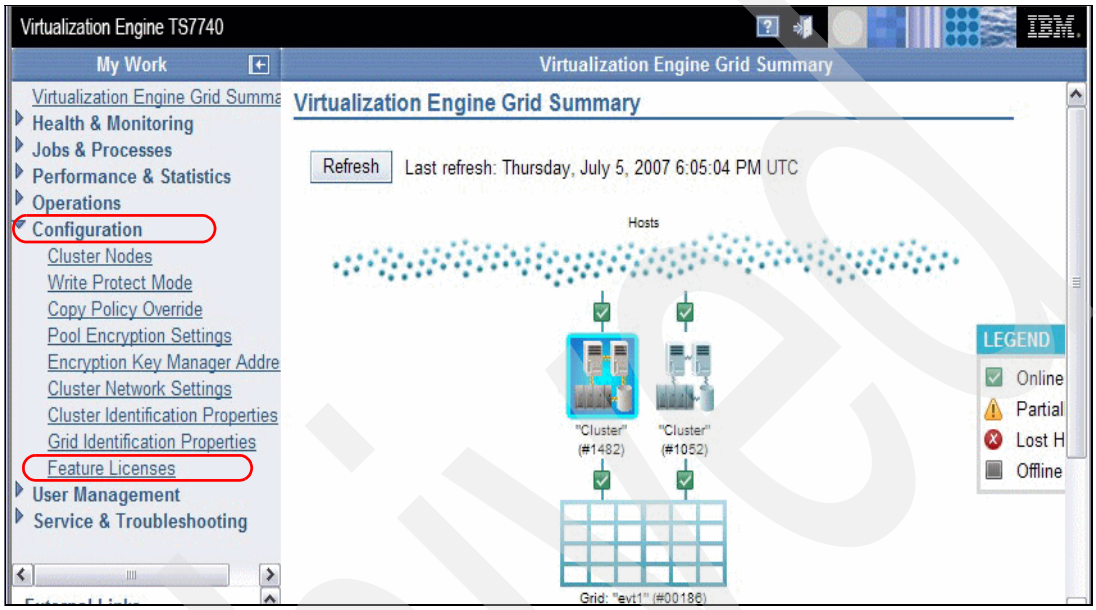


Figure 13-11 TS7700 Virtualization Engine Grid Summary

To set up the TS7700 Cluster for Encryption, select **Configuration** → **Feature Licenses**. This displays a panel similar to that shown in Figure 13-12.

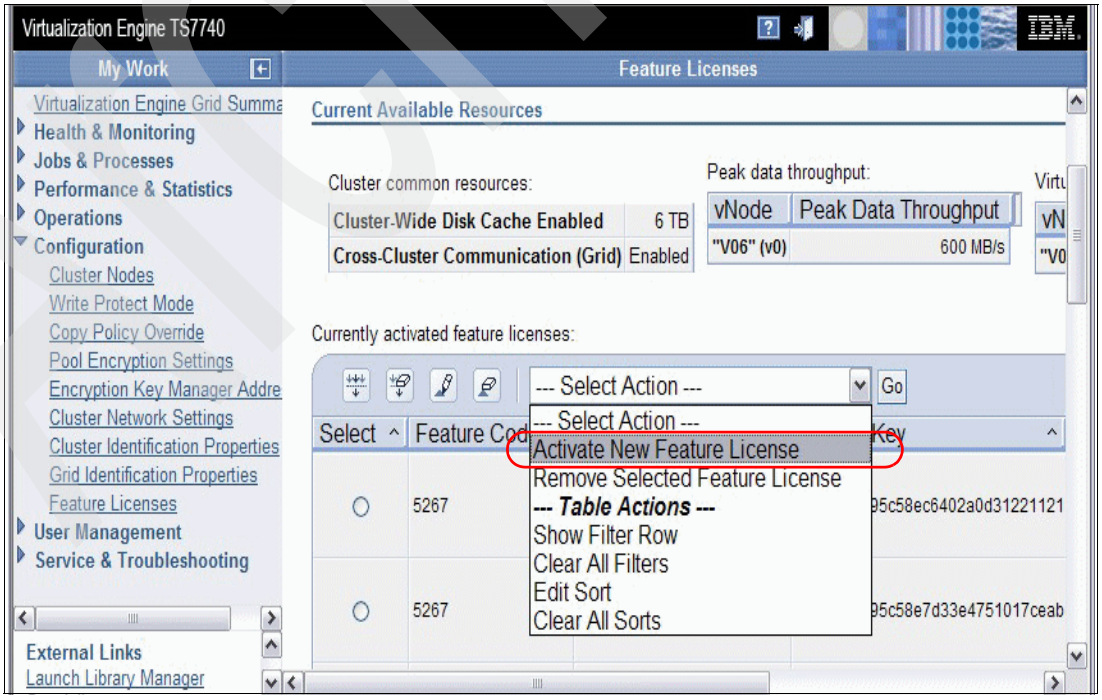


Figure 13-12 TS7700 Feature Licenses

This panel lists the currently activated feature licenses. Do either task:

– If Feature Code 9900 is listed here, the TS7700 has already been enabled for Encryption and you can proceed to the next section.

– If Feature Code 9900 is not listed here:

i. Choose **Select Action** → **Activate New Feature License**, and then click **Go.** A panel similar to Figure 13-13 opens.

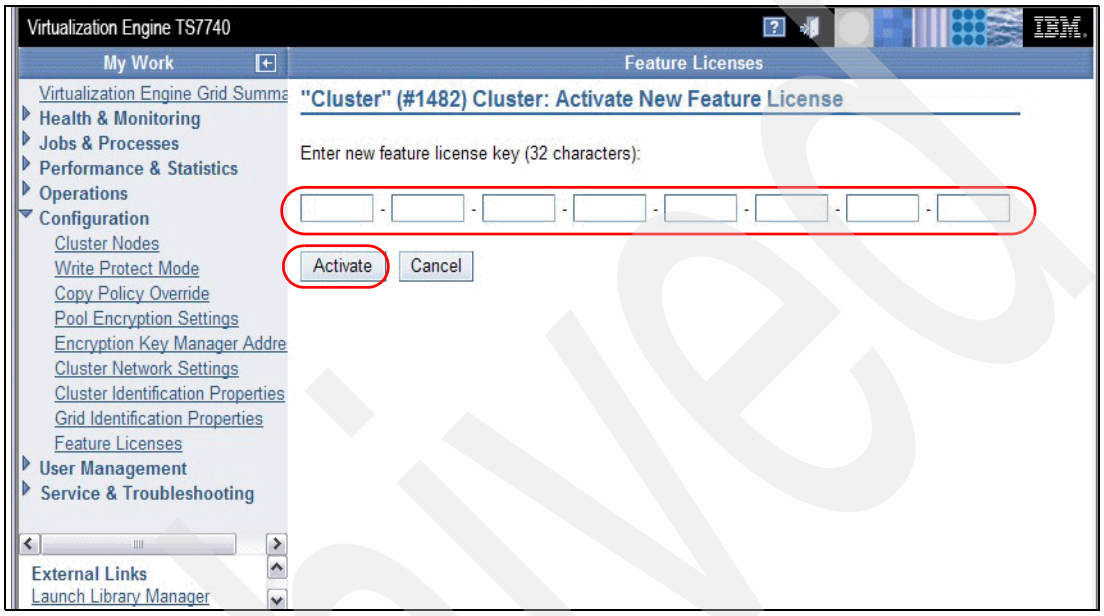


Figure 13-13 Activating a TS7700 Feature License

ii. When you ordered FC9900 for 3957-V06, you were supposed to receive instructions and a license key for the feature. Enter that 32-character Feature Code license key in the boxes on the panel in Figure 13-13 and then click **Activate** to enable the TS7700 for Encryption.

iii. At the Confirm Feature Activation panel, click **Yes**.

13.6.5 EKM addresses

To set the TCP/IP addresses of the EKM, select **Configuration** → **Encryption Key Manager Addresses** to bring up a panel similar to Figure 13-14 on page 439.

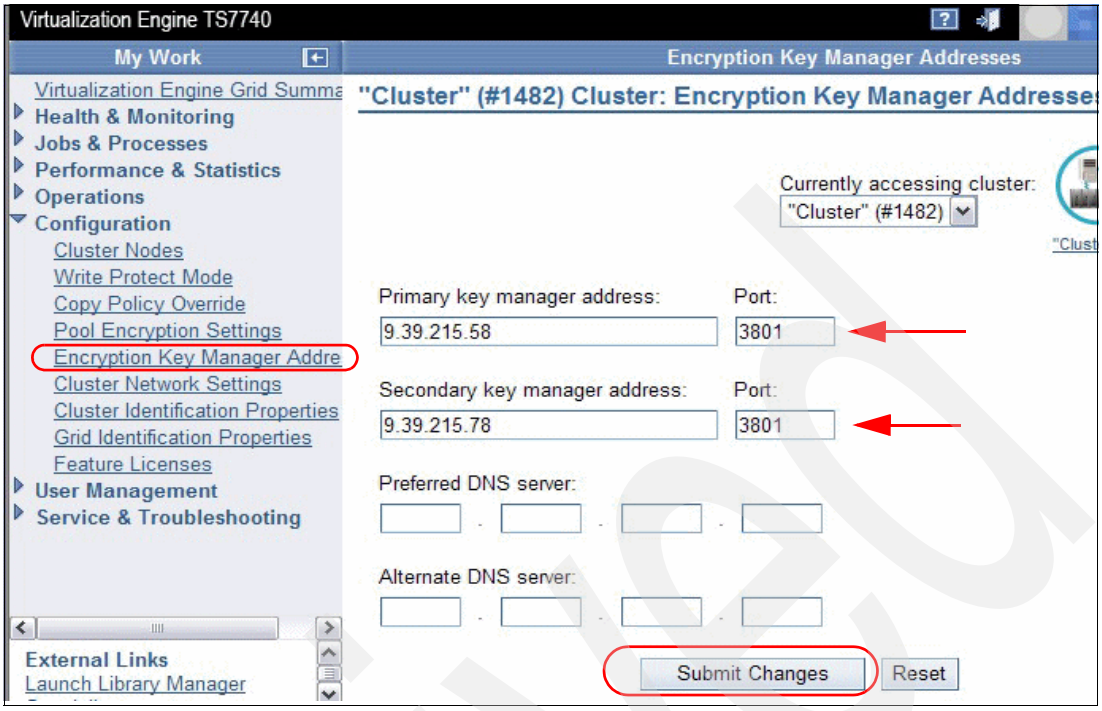


Figure 13-14 Encryption Key Manager Addresses

Enter the information that you obtained from your EKM or network administrator, and then click **Submit Changes**.

Your network or EKM administrator can provide you with the two EKM TCP/IP addresses or domain names.

The standard (and default) EKM port is 3801, but your EKM administrator might have selected a different port number for the EKM configuration file because of possible conflicts on the system on which the EKM runs.

The secondary key manager address on the panel is optional. However, we recommend that you always set up and configure for two EKMs. To maintain continuous access to your data, it is critical to take advantage of all possible redundancies in the subsystem.

Domain Name Server (DNS) addresses are only required if you specify a symbolic domain name for one of the EKMs rather than a numeric IP address. If you have to specify DNS, be sure to specify both a primary and an alternate DNS server so that you do not lose access to your EKM because of one of the DNS servers being down or inaccessible.

If you are sharing keys (KEKs) across different EKMs, be sure to import the necessary certificates into their corresponding keystores. Do not create separate certificates with the same key labels.

13.6.6 Testing EKM connectivity

To test EKM connectivity:

1. Click **Launch Library Manager** (located in the bottom left corner of the panel in Figure 13-14).
2. Click **Monitor Library Manager**.

3. Select **Operator Interventions**.
4. Verify that no EKM-related operator interventions are listed. If you did not configure a secondary EKM, disregard any interventions pertaining to the secondary EKM.
   - If no EKM-related interventions are listed, the EKM setup was successful.
   - If EKM-related interventions are listed:
     i. Verify the IP address of your EKM with your network personnel.
     ii. Verify the EKM port number with the EKM administrator.
     iii. Refer to the EKM-Reported Errors section of the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418.

13.6.7 Configuring Pool Encryption Settings for the TS7700

Many properties can be set for a TS7700 physical stacked volume pool. The properties are discussed in detail in *IBM Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312. Here, we only discuss the Encryption properties that can be defined for a pool.

Be sure to define all key labels in the keystores associated with your EKMs before they will be used by the system. The safest way to ensure this is done is to set up the certificates in the keystores before entering the key labels on this panel. Refer to Chapter 6, "Implementing EKM" on page 159 for details about this process.

The key modes are also entered here. Refer to Chapter 6, "Implementing EKM" on page 159 for descriptions of the key modes.

To set up pools for Encryption, click **Configuration** → **Pool Encryption Settings.** If you have met the prerequisites for Encryption on the TS7700, you see a list of 32 Physical Stacked Volume Pools, as shown in Figure 13-15 on page 441. This panel determines whether encryption is to be enabled or disabled for a storage pool. If Encryption is enabled, you may specify one or two key labels and their key modes.

If storage pools are not displayed, the TS7700 has not met the prerequisites for tape data encryption. You probably have not activated the Encryption Feature License.

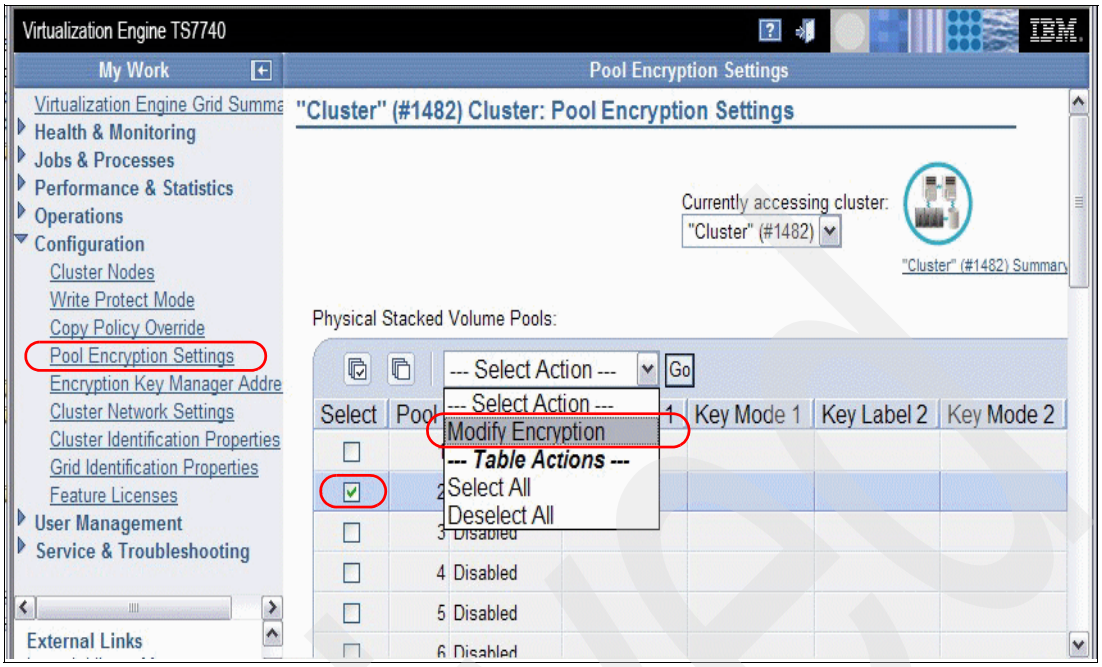


Figure 13-15 TS7700 Pool Encryption Settings

This panel displays the current status of all of the Physical Stacked Volume Pools. Under the Encryption column, a pool is either *Enabled* or *Disabled* for encryption. If you are just beginning this process, all the pools are in a Disabled status. If a pool is Enabled, it has one or two Key Labels/Key Mode pairs listed.

To enable encryption for a pool, check the pool or pools that you want to change or click **Select All** for all pools. Next, choose **Select Action** → **Modify Encryption**, and then **Go**. The panel shown in Figure 13-16 opens.

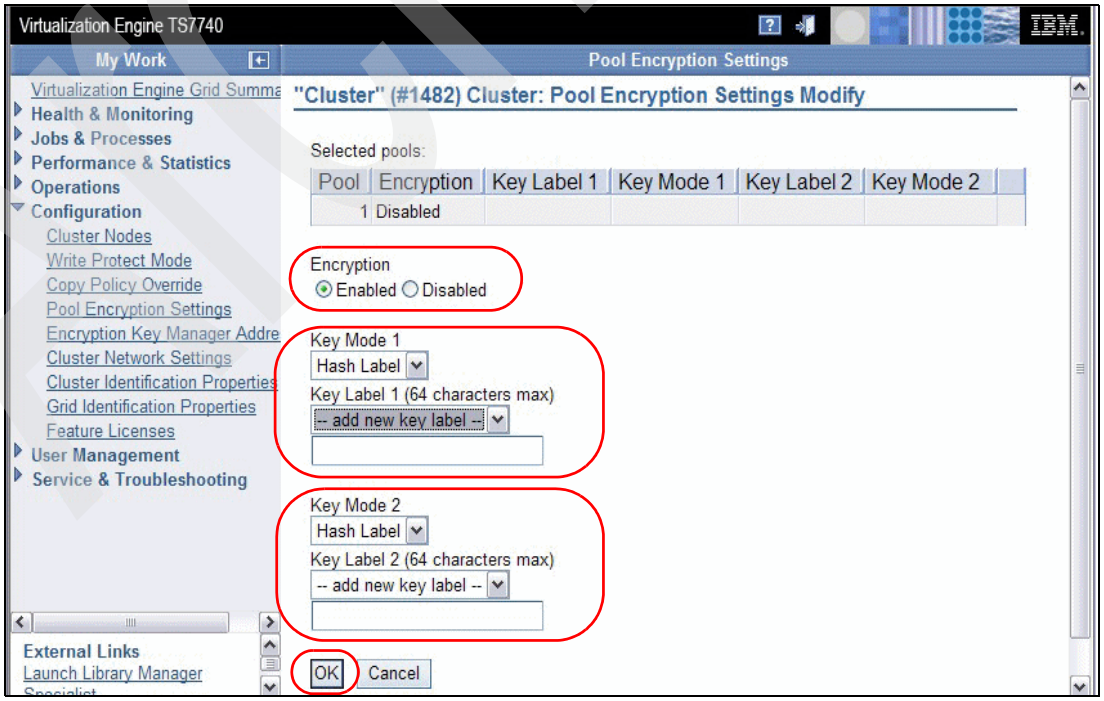


Figure 13-16 Physical Stacked Volume Pool Encryption Settings

Click **Enable** and then enter the Key Mode and Key Label information to be used for this pool. One or two sets can be specified. You can either use the **add new key label** selection from the **Key Label *n*** drop-down list and enter the new key label in the box beneath it, or select a previously added key label from the drop-down list. When you have specified the fields, click **OK** to update the selected pools' encryption settings.

13.7 Implementation considerations

In this section, we summarize additional implementation considerations.

13.7.1 Management construct definitions and transfer

Because the TS7700 Virtualization Engines of a Multi Cluster Grid configuration are managed by different Library Managers, you have to make the construct definitions on *both* Library Managers, or you can transfer the definitions from one Library Manager to the other Library Manager. Refer to *IBM Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312, for the procedures for copying your constructs to another cluster.

13.7.2 Changing storage pool encryption settings

Storage pool encryption settings can be changed at any time. Remember that these settings are applied only when the physical volume is written from beginning of tape. As a result, there might be a mix of encryption usage and key use on physical volumes in a given storage pool until volumes are reclaimed and reused.

Figure 13-17 shows settings of physical volumes in a storage pool over time. In the beginning, encryption is not enabled for the storage pool, so all physical volumes are unencrypted. Encryption is then enabled for the pool using key label `KEK 1`. As unencrypted physical volumes are rewritten from the beginning of tape, they will be encrypted using that key label.

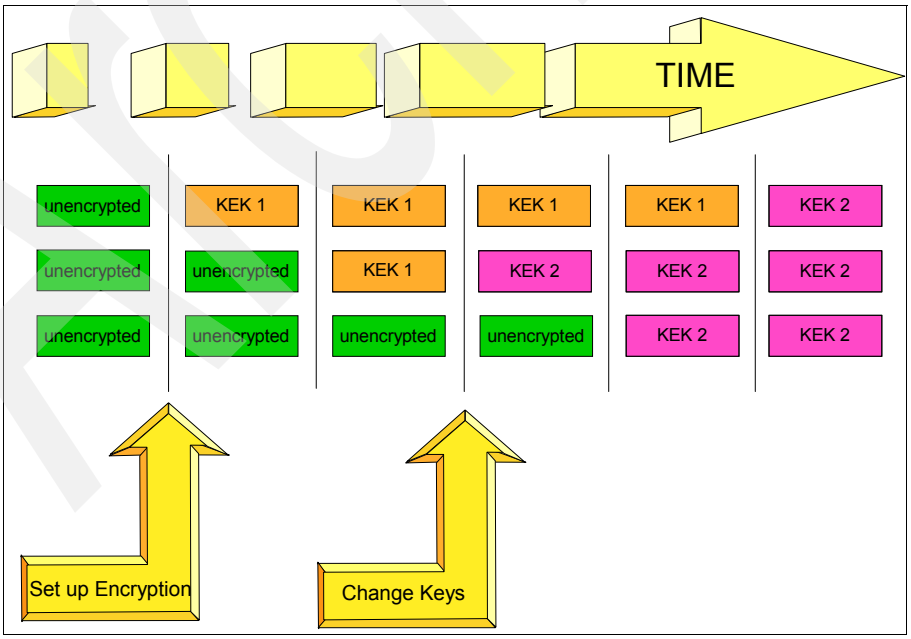


Figure 13-17 Encryption settings changed over time

When the pool is reconfigured to use key label `KEK 2` instead of `KEK 1`, physical tapes exist that are still encrypted with the old key label. The certificate must be maintained in the EKMs long enough for all the volumes to be rewritten with the new `KEK 2` key label.

The transition from unencrypted tapes to encrypted, as well as from one set of KEKs to another set of KEKs, occurs naturally at the pace the tapes are reused by the TS7700. In the meantime, the physical volumes in the storage pool can be a mix of unencrypted tapes and encrypted tapes with different KEKs.

Progress of the migration of data from unencrypted media to encrypted can be monitored by using a new storage pool for encryption rather than modifying the current pool. The Tape Library Specialist *Search Database* function can then be used to see remaining volumes in the old unencrypting storage pool.

13.7.3 Moving data to encrypted storage pools

This section discusses several of the methods available for expediting movement of data onto encrypted tapes. All of the methods begin with the same configuration changes:

- Change the storage pool associated with a storage group construct to a new (empty) pool with encryption enabled.
- Change the properties of the old storage pool to specify the new encryption-enabled storage pool as its reclaim pool.
- Change the properties of the old storage pool to specify that borrowed volumes are to be returned to the scratch pool.

After this change is made, all new data associated with the storage group will migrate to encrypted media, old data recalled to cache will migrate to encrypted media, and other old data will migrate to encrypted media as tapes are reclaimed. The migration can be accelerated by using one or more of the following methods.

If the dual copy function is used, change the secondary pools in a similar manner.

Note: If a migration to new media is planned for the same time frame as encryption implementation, the new storage pool can be set up for both the new media type and the encryption attributes, and both migrations can be combined into one.

Recall data method

After the previously described changes have been made, old data is automatically migrated as it is accessed (*recalled*). To force this process to occur with selected logical volumes, you may run a job that causes them to be recalled. When they are subsequently unloaded and demounted, the TS7700 will see that the pool now associated with the volumes is different than the pool the last time they were copied from the cache, and the TS7700 will copy the volumes to the new pool. This method, combined with directing all new allocations as just described, is probably sufficient if the time period allowed for migration is long enough so that most of the old data has expired or recalled as part of normal job processing, leaving a relatively small amount of data to migrate by forcing a recall. If there are a large number of volumes to migrate, the effort to set up the host jobs to force the recalls becomes cumbersome and cache space will be taken up by the recalled volumes.

Note that if there are a large number of volumes to migrate and this method is desired, the efficiency for recalling the data from tape can be improved by using a map of the logical volumes on them. Rather than randomly recalling the logical volumes, the map can be used to structure the recall job so that the volumes are requested in the order they are located on

the tape. The Bulk Volume Information Retrieval (BVIR) function can provide the physical volume to logical volume mapping information. Refer to the *3494 Bulk Volume Information Retrieval Function User's Guide* white paper for details about how to use the function. The paper is available at:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100430

Note: If a logical volume is in cache at the time that its primary pool assignment is changed, the pool assignment will not affect where the data will be copied until the volume is subsequently unloaded and demounted. It is during the rewind/unload command processing time that the primary pool destination for a logical volume is determined.

Move logical volumes method

With this method, the valid logical volumes on one or more physical volumes are moved to a target pool that has encryption specified. This is accomplished by using the move stacked volume function of the library. Through that function, the logical volumes resident on a range of stacked volumes are moved to a target pool immediately or as each stacked volume becomes eligible for reclaim. This method is a better alternative than the force recall method to move a small number of logical volumes. The TS7700 manages the migration internally, and no host jobs have to be run. Data is moved from tape to tape so no cache space is require.

Reclamation processing method

This method uses reclamation policies to affect how quickly the old data is moved to encrypted volumes because of reclamation.

Reclamation policies

The following policies are supported:

- Reclaim percentage threshold

  A physical volume is eligible for reclaim when the amount of active data on the volume falls below the threshold defined for the pool. A value of 5 - 95% can be specified for this field. This is the default policy.

- Days since last accessed

  A physical volume is eligible for reclaim when the number of days defined in the days without access field has elapsed since any data on the volume has been accessed because of a recall. A value of 1 - 365 days can be specified as a criteria for reclaim. A value of 0 (zero) disables this criteria for reclaim.

- Days since last written

  A physical volume is eligible for reclaim when the number of days defined in the age of last data written field has elapsed since any data was written to the volume. A value of 1 - 365 days can be specified as a criteria for reclaim. A value of 0 (zero) disables this criteria for reclaim.

- Days since last data inactivation

  A physical volume is eligible for reclaim when the number of days defined in the days without data inactivation field has elapsed since any data was invalidated on the volume and the amount of active data on the volume falls below the threshold defined in the maximum active data field. A value from 1 - 365 days can be specified as a criteria for reclaim. A value of 0 (zero) disables this criteria for reclaim. A value of 5 - 95% can be specified for the maximum active data field.

Note: The maximum active data field is only used in conjunction with the days since last data inactivation policy. It is independent of the reclaim percentage threshold field.

A portion of the data on a physical volume is invalidated when the logical volume it represents has been modified and rewritten or deleted (as part of the delete expired volume data function). The remaining data on the physical volume is considered active data.

When reclamation is allowed, the TS7700 examines the physical volumes that have been filled and takes into account all of the policies specified for a pool independently when determining the physical volumes in the pool that are eligible for reclamation. Each pool can have a different set of reclamation policies. During reclamation processing, the TS7700 will prefer eligible physical volumes that have the least amount of active data to move, considering all pools. This means that although a pool might have volumes that meet one or more of the reclamation policies for that pool, if another pool has physical volumes with less active data to move, it will get preference during reclaim. Physical volumes that have not yet been filled are not considered for reclamation.

Note: The reclamation workload for a VTS will increase while the migration of old data is taking place. The impact of that increased workload must be monitored and if it is impacting production use of the VTS, the inhibit reclamation policy setting for the TS7700 must be used to minimize that impact during peak production demand times.

If the dual copy function is used, the reclamation policies have to be set up for both the primary and secondary pools.

Clean-up

After all of the previously full volumes in a pool have been migrated (and assuming you redefined the pool to return scratch volumes), a few physical volumes will be left in the pool. Those volumes are ones that were partially filled when the migration began and are not considered for reclamation. You will have to use the move stacked volume method to complete the migration of the data from the pool.

After you have completed the migration, you may redefine old storage pools for other uses.

13.7.4 EKM operation

The TS7700 tries to maintain TCP/IP connections with all configured EKMs at all times. When contact is lost with a configured EKM, an operator intervention is raised at the Library Manager (LM). This is displayed on the user interface and Web panels for the LM. In addition, hosts are notified of the degraded mode that is displayed on z/OS host consoles. When contact is regained, the intervention is cleared, and the hosts are notified.

Operators have to be trained to recognize and respond to EKM communications loss, because loss of the second EKM will cause a temporary loss of access to encrypted data.

The TS7700 will always perform key exchanges with the primary EKM when it can. If EKMs differ in host or network performance or reliability, the best choice must be specified as the primary EKM.

If communications with no EKMs are established when a key exchange is required, or if contact with EKMs is lost during the exchange, a call-home communication is generated to alert IBM service of the problem. Loss of communications with both EKMs will prevent you

from accessing encrypted data, so this is treated as a serious issue, even though the problem might be with the network or the host on which the EKMs are installed rather than the TS7700 Virtualization Engine.

13.7.5 Tracking encryption usage

If new storage pools are configured for use with encryption, database queries and statistics records give an indication of the amount of data being encrypted on the system.

You can then use the Library Manager specialist or user interface panels to query tape volumes in the encryption storage pools. This is most useful when you configure the pools to borrow and return tapes from the common scratch pool.

You can also use the TS7700 statistics records to gather information regarding storage pools used for encrypted data.

Messages for the host console are sent when a tape has been filled and a database backup has been written to it. For unencrypted tapes, the message is:

"`Database Backup written to Physical Tape XXXXXX`"

For encrypted tapes, the message is:

"`Database Backup written to Encrypted Physical Tape XXXXXX.`"

13.7.6 Data exchange with other data centers or business partners

To share tapes with data centers, other organizations, or contracting services, or for other purposes, EKM can store two sets of wrapped encryption keys on the tape. This approach allows another organization to read that specific tape without your providing to them any shared secret information or compromising the security of your certificates and keys. This approach is accomplished by adding the public part of the other organization's public-private certificate and keys to your EKM's keystore as a second alias (or key label). When the tape is written, the encryption keys are stored on the tape in multiple places and protected by two sets of public-private keys: your public-private keys and the other organization's public-private keys. The other organization is then able to use their EKM and their public-private certificate and private key to unwrap the data key that allows them to read that specific tape. This gives you the flexibility to make a specific tape readable by both your own organization and another organization. If you want to take advantage of this capability, you must add that other organization's certificate and public key to your keystore.

In other words, if you want to send an encrypted tape cartridge to another data center for data access, two key labels have to be used with the second key label using an encoding method of HASH. This enables you to use a key label that is different than the receiving organization's key label for the same key.

13.8 TS7700 Encryption with z/VM, z/VSE, or z/TPF

Encryption within the TS7700 is supported when z/VM, z/VSE, or z/TPF is attached to the TS7700. It is supported totally with the Outboard Policy Management capability within the TS7700. These operating systems will not be able to dynamically assign the Storage Group and Management Class constructs, because they do not have the capability to pass these constructs to the TS7700.

However, a static assignment of Storage Group and Management Class can be implemented within the TS7700 to control encryption. All of the TS7700 implementation steps that have been described earlier in this chapter (except for the DFMSM Storage Group and Management Class constructs) must still be implemented for encryption with these operating systems. Also, one additional step can be performed at the TS7700 that allows a static assignment of TS7700 Storage Group and Management Class to logical volume ranges within the TS7700.

To implement Outboard Policy Management for these operating systems when they are attached to a TS7700:

1. Define your storage pools and constructs as described earlier in this chapter.
2. Insert the logical volumes (to be used by these operating systems) in groups through the TS7700 Management Interface. Refer to *IBM System Storage TS7700 Virtualization Engine: Tape Virtualization for System z Servers,* SG24-7312, for this procedure.
3. Go to the 3953 Library Manager or the 3494 Library Manager and assign the required static construct name to these logical volume ranges through the Manage Logical Volumes function. A best practice is to define groups of logical volumes with the same construct names assigned. Then, during insert processing, direct them to different LM volume categories so that all volumes in one LM volume category will have identical constructs assigned.

Managed Logical Volumes procedure

From the ETL Specialist, select **Manage Logical Volumes** from the **Administer VTS *n*** work items, which displays the panel shown in Figure 13-18.

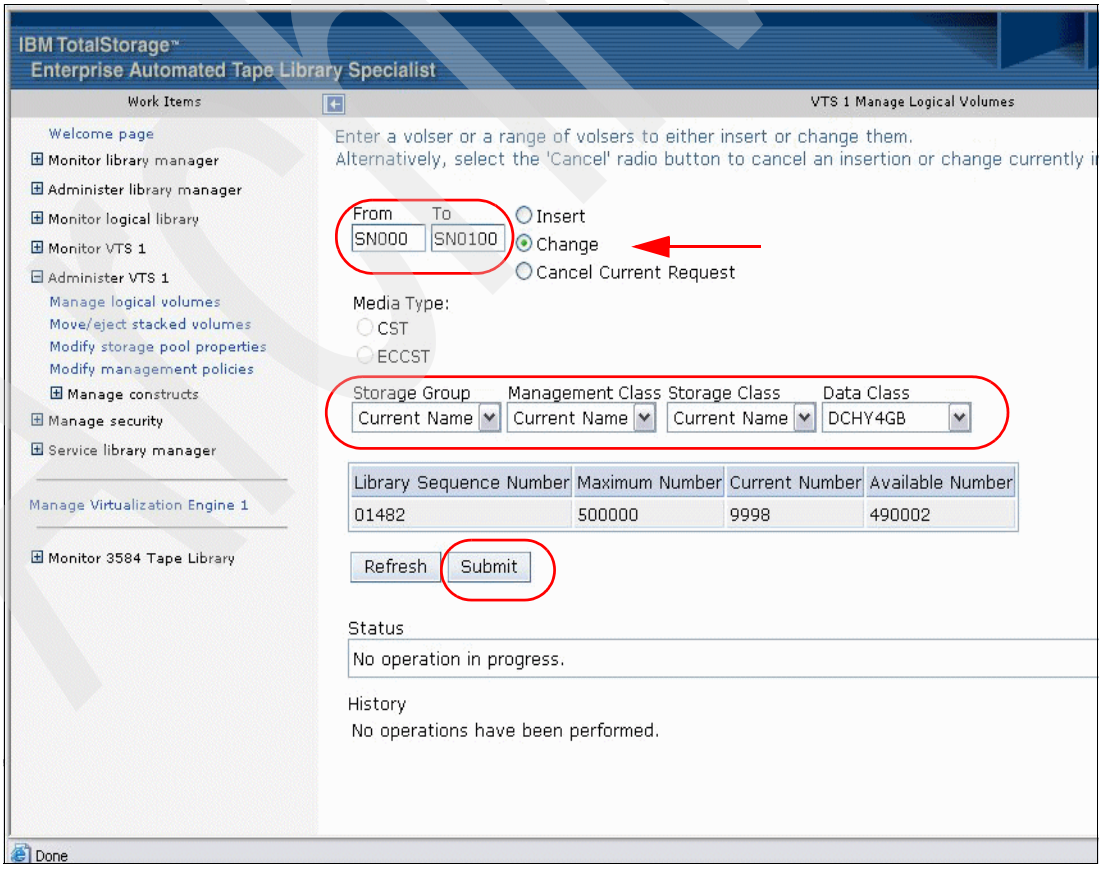


Figure 13-18 Manage Logical Volumes panel

To set up for encryption:

1. Enter the VOLSER ranges in the **From** and **To** boxes.
2. Click **Change**.
3. Enter or select the **Storage Group**, **Management Class**, **Storage Class**, and **Data Class** to be used for this VOLSER range.
4. Click **Submit**.

Repeat this as necessary for other VOLSER ranges. Only the Storage Group and Management Class are required to invoke Encryption, but you probably want to specify the other constructs, also. For Encryption, the **Storage Group** and **Management Class** selected must point to a storage pool where Encryption is specified.

14

Implementing TS1120 and TS1130 Encryption in an Open Systems environment

This chapter provides detailed information for the installation and implementation of Encryption in both the AIX and Windows environments using the IBM System Storage TS3500 Tape Library and the IBM System Storage TS1120 Tape Drive and IBM System Storage TS1130 Tape Drive.

We discuss the following topics:

- Implementation checklist
- Implementing System-Managed Encryption (SME)
- Implementing Library-Managed Encryption (LME)
- Implementing Application-Managed Encryption (AME)

14.1 Encryption overview in an Open Systems environment

The three methods to implement encryption in the Open Systems environment are Library-Managed Encryption (LME), System-Managed Encryption (SME), and Application-Managed Encryption (AME). Using the TS3500, we provide implementation information and examples of each encryption method as reflected in Table 14-1.

Table 14-1 Open Systems encryption solutions supported

	AME	SME	LME
IBM library	IBM Tivoli Storage Manager (ITSM) Only	AIX only using Atape Windows Solaris Linux	TS3500 TS3400 3494
Rack or silo	ITSM Only	AIX only using Atape Windows Solaris Linux	N/A

In addition, Table 14-1 reflects all the environments that are supported, as of this writing, of the Encryption solution. All applications and their associated operating systems that currently support TS1120 or TS1130 drives in an Open Systems-attached environment. For a current list see the *IBM Tape Device Drivers Installation and User's Guide*, GC27-2130. Refer to:

`ftp://ftp.software.ibm.com/storage/devdrvr/Doc/IBM_Tape_Driver_IUG.pdf`

An important note is that the installation of the Advanced Library Management System (ALMS) on a TS3500 library affects your ability to use more than one method concurrently. On a partitioned TS3500 library that has ALMS installed, all three encryption implementation methods can be used concurrently, although a non-ALMS-partitioned library can only have one method implemented at a time. For example, in an ALMS-enabled TS3500 library that has three logical libraries, each logical library can use a different encryption method.

Note: To use a TS1130 tape drive in a TS3500 tape library the ALMS feature is required.

Figure 14-1 is an example of an ALMS-enabled TS3500 partitioned into three logical libraries using all three encryption methods simultaneously. The ability to implement all three methods concurrently can only be accomplished if ALMS is enabled.

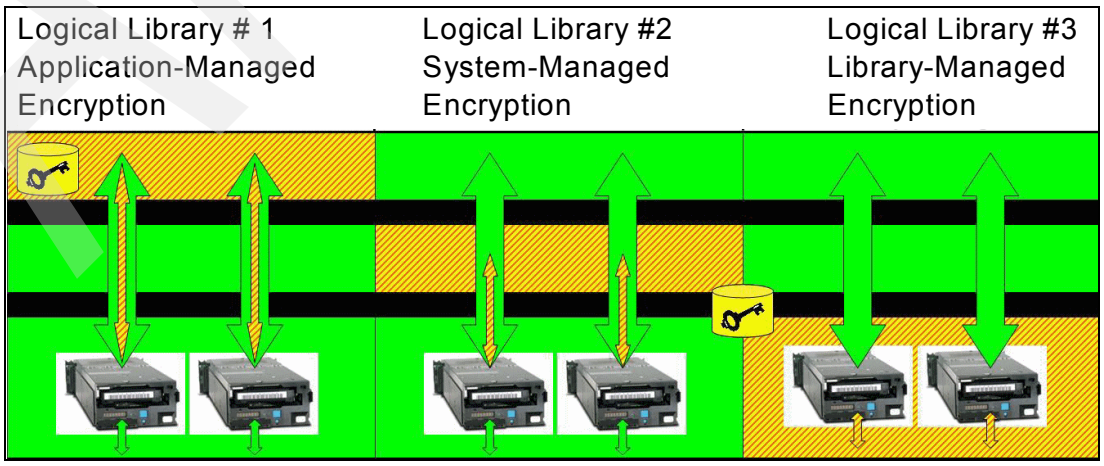


Figure 14-1 ALMS-partitioned TS3500 using all three encryption methods

Important: The LME and SME methods can share the same Encryption Key Manager (EKM) or Tivoli Key Lifecycle Manager (TKLM), which allows the two modes to be interchanged. At this time, ITSM cannot share and cannot utilize encryption keys between the AME method and either the LME or SME methods of encryption. ITSM provides its own structure for key management, so consequently, storage tapes encrypted with ITSM and sent to a business partner will require ITSM at the business partner to decrypt them.

14.2 Adding drives to a logical library

When you add a drive to a logical library, be aware that certain guidelines exist depending on whether the Advanced Library Management System (ALMS) is enabled or not.

14.2.1 Advanced Library Management System considerations

When sharing your TS3500 Tape Library between Open Systems and System z hosts, ALMS is required. In the following section, note the use of the terms *encryption-enabled* as opposed to *encryption-capable*. The following paragraphs attempt to describe the impact in detail.

ALMS-enabled

In a TS3500 Tape Library, which uses ALMS, encryption-enabled drives in a logical library must be defined as all library-managed, all application-managed, or a combination of system-managed and no encryption.

A drive can only be shared in homogeneous logical libraries; that is, all drives within a logical library must be set to the same encryption method.

A drive that is not encryption-capable cannot be added to a logical library defined to use LME or AME.

Within a logical library, encryption-capable drives can reside with non-encryption-capable drives, but encryption-capable drives are not allowed to become encryption-enabled. Encryption-capable and encryption-enabled drives can coexist in the same frame. A non-encryption-capable drive can be added to a System-Managed logical library.

ALMS-disabled or non-ALMS libraries

Certain guidelines apply when you configure encryption-capable drives in non-ALMS libraries. The following drive configurations are possible:

- Encryption-capable drive is added to a library that contains non-encryption-capable drives.

  You can add a drive that is encryption-capable to a library that previously did not have encryption-capable drives, but you cannot enable encryption.

- Non-encryption-capable drive is added to a library that contains encryption-capable drives, which are not encryption-enabled.

  You can add a non-encryption-enabled drive to a library that is installed with encryption-capable drives, which are not set for encryption. The drives that are encryption-capable are restricted from becoming encryption-enabled.

- Non-encryption-capable drive is added to encryption-capable and encryption-enabled physical library.

  A drive that is not encryption-capable cannot be added to a library that has encryption-enabled drives and will be restricted (not available for use). In the event that you create this configuration, the Operator Panel or Tape Library Specialist Web interface displays the number of restricted drives.
- TS1130 Drives are not supported.

14.3 Managing the encryption and business partner exchange

In addition to keeping your on-site data safer, encryption adds security to your data when it leaves your premises. After it leaves your premises, the security procedures that you have established that limit access to this data are pretty much null and void. This is where encryption really proves its worth.

Figure 14-2 reflects the usage of both symmetrical and asymmetrical key usage in the IBM Encryption solution. The user data is encrypted with a symmetrical key. Symmetric encryption is both fast and secure and allows the data to be written to the cartridge at near line speed. The key that was used to encrypt the user data is then encrypted with an asymmetrical key and then stored on the cartridge. In this example, KEK refers to *key encrypting key*, which is also referred to as the *public-private key* or the Rivest-Shamir-Adleman algorithm (*RSA) key*.

Encryption key management allows you to specify two key encrypting keys (KEKS), which are used to create the two Externally Encrypted Data Keys (EEDKs) that are placed on the cartridge both in the cartridge memory (CM) and on three non-user accessible places on the tape media.

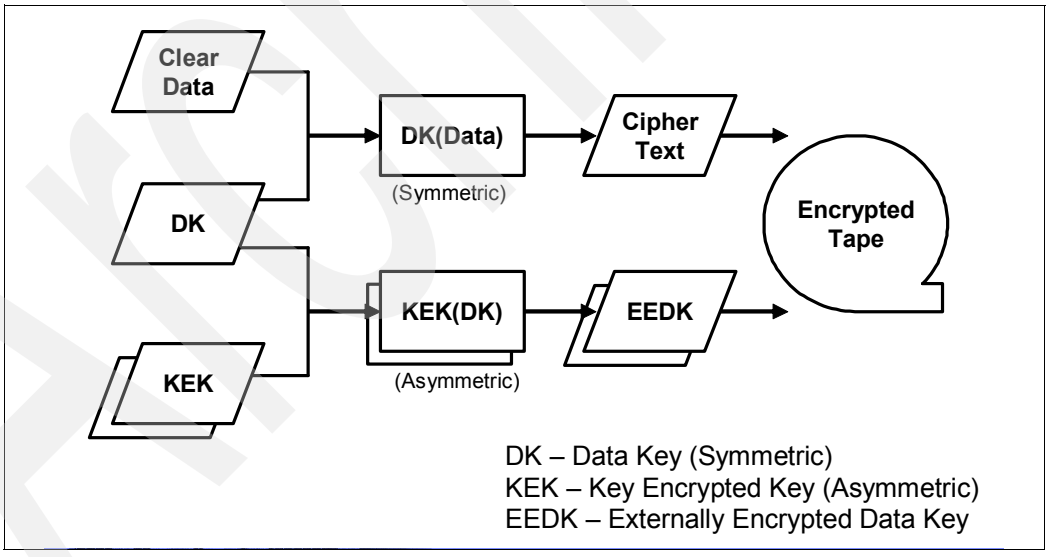


Figure 14-2 Key flow

Table 14-2 on page 453 is a useful reference table when you are in the process of deciding how to implement encryption. The encryption method that you choose affects how you manage which files get encrypted and as a by-product, how and where to influence which keys are used to create those files. AME provides the most flexibility, because you can decide at the file level whether it is eligible for encryption.

SME is done at the drive level; so, for files that have to be encrypted, you have to direct those allocations to specific tape drives. LME policy is controlled and specified at the tape cartridge VOLSER level, or, in the case where Internal Label Encryption Policy is selected, and NetBackup is being used, encryption is based on the pool ID supplied by NetBackup. AME using ITSM is managed at the file level, controlled by IBM Tivoli Storage Manager.

Table 14-2 Managing Open Systems encryption

Type of encryption management	Description	Notes	Sample environment
AME	The application directly controls whether a cartridge is encrypted.	► The application manages allocations of cartridge format types to appropriate drives in heterogeneous environments. ► This method supports a large number of systems (for example, IBM and other libraries that are not IBM). ► You can control encryption based on existing *DEVCLASS* type policies. ► Program updates are required for the application and, in some cases, the underlying operating systems or components.	ITSM
SME	The device driver controls whether data is encrypted. All cartridges in the logical library will become encrypted when written from beginning of tape.	► *Encryption management is by drive,* rather than by cartridge. ► This method transparently supports a large number of systems (for example, other libraries that are not IBM). ► New IBM device drivers must be deployed on tape servers. ► This method is available for only IBM device drivers. ► The IBM device driver supports failover to the drive's alternate Fibre Channel connection.	IBM Atape AIX device driver in an Open Systems Environment
LME	The library controls whether a specific cartridge is encrypted.	► Based on drive settings, the library allows you to *encrypt data by cartridge*. ► This method is the most transparent, because the application and device driver might not require updates. ► New library firmware must be deployed. ► This method is supported only on IBM libraries.	IBM 3584 Tape Library

14.3.1 Disaster recovery considerations

Your disaster recovery (DR) hardware environment has to be considered when implementing encryption, because the method that you choose will impact the equipment required at your disaster recovery site.

Your recovery process will have to be changed to ensure that the EKM and its associated keystore or an identical TKLM system and a current backup are available and running prior to you having to access encrypted data. With this in mind, ensure that you do *not* encrypt your EKM keystore or TKLM backup.

Important: Do *not* encrypt your EKM or TKLM backup files.

Encryption compatibility

LME data and SME data are fully compatible. In this case, data encrypted using LME can be decrypted using SME, when AIX is the host operating system. The reverse is also true for AIX hosts, so that data encrypted using SME can be decrypted using LME.

However, it is also important to note that if LME is implemented, a TS3500, TS3400 or the ability to run SME must be available at the DR site. If a TS3500 is not available, a TS3400 or SME environment (AIX and Atape) along with the EKM and its associated files are necessary and will have to be available prior to you having to read encrypted data.

Important: If LME is implemented, a TS3500, TS3400 or a SME compatible environment will be necessary at your disaster recovery site.

14.3.2 Keeping track of key usage

To keep track of the key labels by VOLSER that were used to create the EEDKs on your encrypted cartridges, refer to the EKM or TKLM audit log. The EKM KeyManagerconfig.properties file holds the location of the audit log. The audit log created by EKM has very useful information in it. Example 14-1 is an EKM audit log entry. The log entry contains the time and date of file creation, tape drive serial and worldwide name (WWN) used, cartridge VOLSER, and key alias/labels used to create the EEDKs on the cartridge.

Example 14-1 EKM audit log example

```
Runtime event:[
  timestamp=Tue Sep 28 23:21:02 MST 2000
  event source=com.ibm.keymanager.c.fb
  outcome=[result=successful]
  event type=SECURITY_RUNTIME
  resource=[name= Drive Serial Number: 000001365071 WWN: 500507630F0c8501 VolSer:
J1G490 Key Alias/Label[0]: tape_sol_tst_shr_pvt_1024_lbl_01 Key Alias/Label[1]:
tape_sol_tst_shr_pvt_1024_lbl_01;type=file]
  action=stop
  ]
```

Using the information in the audit log, we created a table as shown in Figure 14-3 on page 455. This information can be referenced if required to verify which key labels were used to encrypt the data on the cartridge, and consequently, who can access the data.

The audit subsystem writes textual audit records to a set of sequential files as various auditable events occur during the EKM processing of requests. The audit subsystem writes to a file. The directory, file name, and size are configurable and set in the EKM configuration file. 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 time stamp, 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 with their names sequenced by the time stamp 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 file system, a script or program must be created, which monitors the set of files in the configured audit directory/folder/container. As files are closed and named based on the time stamp, the file's contents must 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 EKM while EKM is running (this file does not have a time stamp in the file name).

For the remainder of the audit entry types created, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418, which has a chapter about audit records.

	Month	Date	Time	Drive	WWN	Volser	Label 0	Label 1
Fri	Sep	22	13:57:57	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Fri	Sep	22	13:59:09	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Fri	Sep	22	14:00:19	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Fri	Sep	22	14:01:30	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	10:23:26	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	10:29:30	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	10:34:36	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	10:40:43	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:03:35	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:13:22	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:23:00	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:32:13	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:41:16	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:50:15	1365135	50050763024093A6	J2R140	tape_sol_tst_Shr_pvt_1024_lbl_01	tape_sol_tst_Shr_pvt_1024_lbl_02
Tue	Sep	26	11:58:57	1365012	500507630F0C8509	J1G159	tape_sol_tst_shr_pvt_1024_lbl_01	tape_sol_tst_shr_pvt_1024_lbl_01
Thu	Sep	28	23:17:08	1365046	500507630F0C8511	JA1049	tape_sol_tst_shr_pvt_1024_lbl_01	tape_sol_tst_shr_pvt_1024_lbl_01
Thu	Sep	28	23:17:41	1365039	500507630F0C850A	J1G490	tape_sol_tst_shr_pvt_1024_lbl_01	tape_sol_tst_shr_pvt_1024_lbl_01
Thu	Sep	28	23:20:51	1365012	500507630F0C8509	J1G159	tape_sol_tst_shr_pvt_1024_lbl_01	tape_sol_tst_shr_pvt_1024_lbl_01
Thu	Sep	28	23:21:02	1365071	500507630F0C8501	J1G490	tape_sol_tst_shr_pvt_1024_lbl_01	tape_sol_tst_shr_pvt_1024_lbl_01

Figure 14-3 Formatted EKM audit log entries

Tape System Reporter on the TS3500

You may also use the Tape System Reporter (TSR) tool on Windows to monitor tape drives and encrypted cartridges in TS3500 tape libraries. The TSR tool saves the tape library statistics logs into a database and provides a front end to the database to assist with tracking and trending your tape libraries behavior. TSR requires ALMS and can be downloaded from here:

`ftp://ftp.software.ibm.com/storage/358x/3584/TSR/`

One of the many parameters it tracks is when cartridges become encrypted and in which drives a cartridge is used.

14.4 Encryption implementation checklist

In this section, we discuss necessary planning and setup tasks.

14.4.1 Planning your EKM environment

Many factors must be considered when you plan how to set up an encryption strategy.

Encryption setup tasks at a glance

Before you can use the encryption capability of the IBM TS1120 or TS1130 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: Contact your IBM marketing representative for additional information about encryption on the IBM TS1120 or TS1130 Tape Drive.

14.4.2 EKM setup tasks

Before you can encrypt tapes, EKM or TKLM must first be configured and running so that it can communicate with the TS1120 or TS1130 tape drives. EKM or TKLM does not have to be running while tape drives are being installed, but it must be running to perform encryption. It is easier to verify a correct EKM configuration if it is running at drive install time as the IBM Service Representative will be able to run an end to end encryption test from the TS3500 operator panel. Note that you do not have to set up EKM or TKLM if you are implementing AME.

Perform the following tasks before using EKM:

1. Decide what systems to use as EKM servers.
2. Upgrade server operating system if necessary.
3. Upgrade IBM Java Virtual Machine if necessary.
4. Install IBM Java Unrestricted Policy Files.
5. Upgrade EKM JAR if necessary, which can be found at the IBM Web site at:

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

   Or visit the following Web site, click **Download,** and look for **IBM Encryption Key Manager for the Java platform**:

   `http://www.ibm.com/servers/storage/support/tape/ts1120/downloading.html`
6. Decide on the keystore type.

   The type of keystore that you select, depending on your environment, can affect your future flexibility for encryption implementation. Refer to 7.1, “Keystore and SAF Digital Certificates (keyrings)” on page 228 for more details.
7. Define the keystore:
   a. Create a keystore and certificates (AIX and other operating systems) as shown in Example 6-28 on page 180.
   b. Import or create keys and certificates into the keystore. Refer to “Importing and exporting certificates and why” on page 181.
8. Define the EKM configuration file. Refer to Example 6-35 on page 191.
9. Define tape drives to EKM or set the *drive.acceptUnknownDrives* EKM configuration property value on.
10. Start the EKM.

Note: If you are implementing AME, these steps are *not* necessary, because the EKM function is handled by IBM Tivoli Storage Manager.

14.4.3 Application-Managed Encryption setup tasks

To perform encryption on the TS1120 Tape Drive, the following actions are required:

- Ensure that the TS1120 tape drives are encryption-capable.
- Ensure that library and tape drive firmware are at the correct levels to support encryption.

- Enable encryption on the IBM TS1120 (3592-E05) Tape Drive. Refer to *IBM System Storage TS3500 Operator's Guide* for configuring the IBM TS1120 Tape Drive on TS3500. For stand alone or rack mounted tape drives, this is an IBM service support representative (SSR) task.

To perform encryption on the TS1130 Tape Drive, the following actions are required:

- Ensure automation is updated to support the TS1130 tape drive
- Ensure that the TS1130 tape drives are encryption-capable.
- Enable encryption on the IBM TS1130 (3592-E06 or 3592-EU6). Refer to *IBM System Storage TS3500 Operator's Guide with ALMS* for configuring the IBM TS1130 Tape Drive on TS3500.

Common activities to setup AME include:

- Install the appropriate IBM tape device driver level (Atape, for example).
- Set up encryption policies. Refer to *IBM Tivoli Storage Manager for AIX Administrator's Guide,* GC32-0768.
- Perform write/read operation to test encryption.
- Verify encryption of the test volume by Autonomic Management Engine (AME):
  - Issue QUERY VOLUME FORMAT=DETAILED.
  - Verify that the Drive Encryption Key Manager is set to ITSM.

14.4.4 System-Managed (Atape) Encryption setup tasks

Setup tasks for SME can be split into a group of preliminary tasks that can be completed in advance and a group of primary tasks to enable Encryption.

Preliminary tasks

To perform SME on the IBM TS1120 or IBM TS1130 Tape Drive, the following tasks are required:

- Install encryption-capable TS1120 or TS1130 tape drives and TS3500 library.
- Create a certificate-populated keystore.
- Install the IBM Encryption Key Manager component for the Java platform (EKM) or Tivoli Key Lifecycle Manager (TKLM).

Primary tasks

Primary tasks include:

- Firmware updates:
  - If using a TS1130 with a TS3500 install ALMS feature.
  - Update TS3500 library firmware.
  - Update tape drive firmware.
- Encryption-enable the IBM TS1120 (3592-E05) or IBMTS1130 (3592-E06 or 3592-EU6) Tape Drive:
  - You can do this using the TS3500 Web GUI or as an IBM Service task.
  - Refer to *IBM System Storage TS3500 Tape Library Operator Guide,* GA32-0560, for configuring the IBM TS1120 Tape Drive on TS3500. For stand alone or rack mounted tape drives, this is an IBM Service task.

- Update the Atape device driver.
- Update the Atape EKM Proxy Configuration file with EKM or TKLM IP addresses.
- Update device attributes by using **rmtx**, which is the drive you want to system-manage using the Atape device driver. Update using SMIT, **tapeutil**, or TS3500:
  - Use System Encryption FCP Proxy Manager.
  - System encryption for write commands at beginning of partition (BOP), which is similar to beginning of tape (BOT).
- Use tapeutil functions to verify EKM paths and encryption configuration.

14.4.5 Library-Managed Encryption setup tasks

To perform encryption on the IBM TS1120 or TS1130 Tape Drive, the following components are required:

- Encryption-capable TS1120 or TS1130 tape drives
- Keystore
- IBM Encryption Key Manager component for the Java platform (EKM) or Tivoli Lifecycle Key Manager (TKLM)

14.5 Implementing Library-Managed Encryption

This method is best for TS1120 or TS1130 tape drives in an Open Systems-attached IBM System Storage TS3500 Tape Library. Barcode Encryption Policies and Internal Label Encryption Policies specifying when to use encryption are set up through the IBM System Storage Tape Library Specialist Web interface. Policies are based on cartridge volume serial numbers or on an internal label written in the tape header. Key generation and management are performed by EKM, a Java application running on a library-attached host. Policy control and keys pass through the library-to-drive interface; therefore, encryption is transparent to the applications.

SME and LME are transparent to one another. In other words, a tape encrypted using SME can be decrypted using LME, and alternatively, provided they both have access to the same EKM or TKLM keystore and use an IBM device driver that supports system managed encryption as the host operating system.

14.5.1 LME implementation tasks

To implement LME:

1. Install and cable the IBM TS1120 or TS1130 Tape Drive (IBM Service or SSR task).
2. Update the tape system library and the tape drive firmware. If using a TS1130 install ALMS on the TS3500
3. Use IBM System Storage Tape Library Specialist to enable the IBM TS1120 or IBM TS1130 Tape Drive and the IBM TS3500 Tape Library for LME (refer to *IBM System Storage TS3500 Tape Library Operator Guide,* GA32-0560 or *IBM System Storage TS3500 Tape Library Operator Guide with ALMS,* GA32-0594).
4. Add EKM or TKLM IP addresses.
5. Set up Barcode Encryption Policy (BEP) or Internal Label Encryption Policy (ILEP).

Note: Previously, the term Scratch Encryption Policy (SEP) referred to the only encryption policy available for the LME method. SEP uses a volume's VOLSER to determine the encryption policy. A new LME policy called Internal Label Encryption Policy (ILEP) has been added. This policy also relates to scratch volumes. Thus, SEP now refers in general to the various LME policies. A new term, Barcode Encryption Policy (BEP), is now used to refer to the policy previously known as SEP.

6. Use library diagnostic functions to verify EKM paths and encryption configuration. As of this writing this requires physical access to the TS3500 operator panel. Select **Menu** → **Service** → **Tests/Tools** → **Diagnostics** → **Test Encryption Key Path Setup**.

Notes: All levels of the TS1130 are encryption capable. To use the TS1130 with a TS3500 the ALMS feature must be installed and 8160 or later firmware must be loaded on the tape library.The firmware encryption versions are 6470 for the TS3500 library and D3I1_942 for the TS1120 tape drives.

14.5.2 Upgrading firmware

In this section, we review the steps necessary to upgrade both TS3500 library and TS1120 or TS1130 drive firmware.

Verify library firmware levels

Using the Web GUI, confirm that the TS3500 library is at firmware version 6470 or later for the TS1120 or 8160 or later for the TS1130 by selecting **Node Cards** under the **Service Library** Work Item. You then see Figure 14-4.

Note: If the Web GUI loads a blank page, clear the Web browser cookies, and then reload.

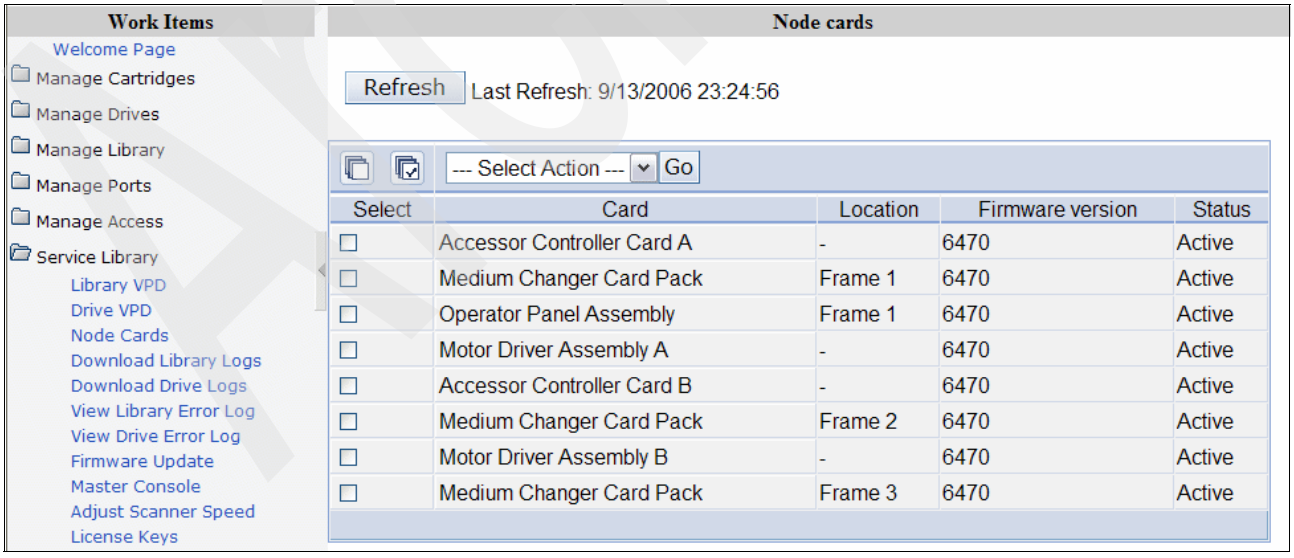


Figure 14-4 Library Firmware version level

Because the library is at the correct firmware level, we do not have to update anything. However, if you did have to update the library firmware level:

1. Perform the steps in “Installing software feature codes like ALMS” on page 460.
2. Upgrade the firmware as described in “Update library firmware” on page 460.

Installing software feature codes like ALMS

To install and enable ALMS on your TS3500 using the TS3500 Web interface:

1. Enter the ALMS license key.
2. Type the Ethernet IP address on the address line of the browser and press Enter. The Welcome page displays.
3. Select **Service Library** → **Download Library Logs**. Then, select the log/file type **NVRAM Backup/Memory Dump**, select the **Download** option from the Select Action pull-down menu, and click **Go**. This can take several minutes.

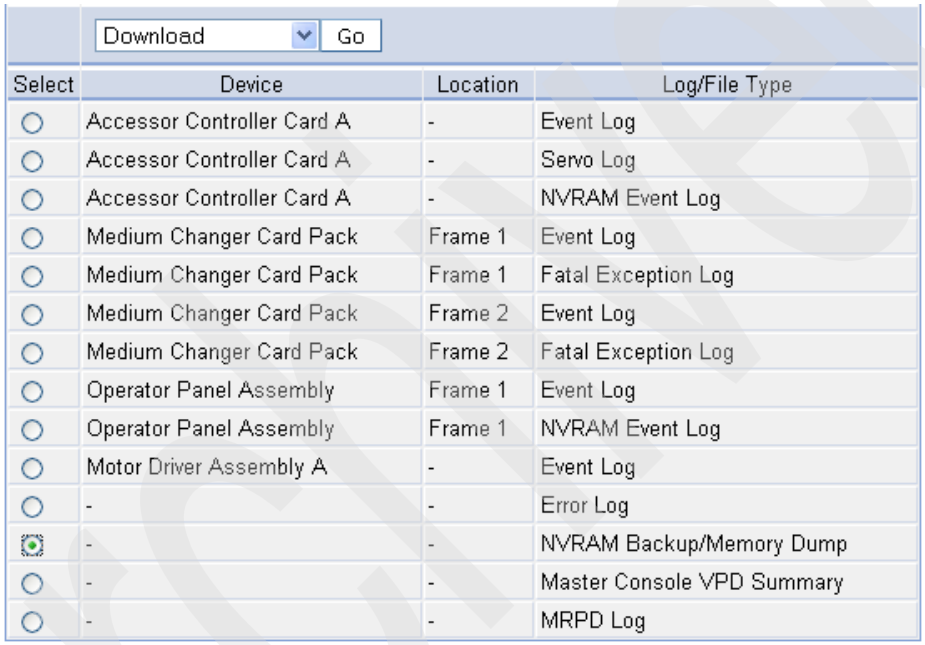


Figure 14-5 Backing up your current library configuration

4. At your next service window, select **Library** → **ALMS,** select **Enable**, and follow the on screen instructions. This process can take awhile depending on your firmware version and library configuration.

Update library firmware

To update library firmware by using the TS3500 Web interface:

1. Type the Ethernet IP address on the URL line of the browser and press Enter. The Welcome page displays.
2. Select **Service Library** → **Firmware Update**. The Firmware Update panel displays as shown in Figure 14-6 on page 461.

Figure 14-6 TS3500 Tape Library Firmware Update panel in the Wizard

3. If you do not already have the new firmware to install, click **Go To UltraScalable Tape Library Technical Support**. This takes you to the Web page where you can download the library firmware. Ensure that you extract it prior to continuing with the next step.
4. From the Firmware Update panel in Figure 14-6, select **Launch Firmware Update Wizard**.

   The panel shown in Figure 14-7 displays.

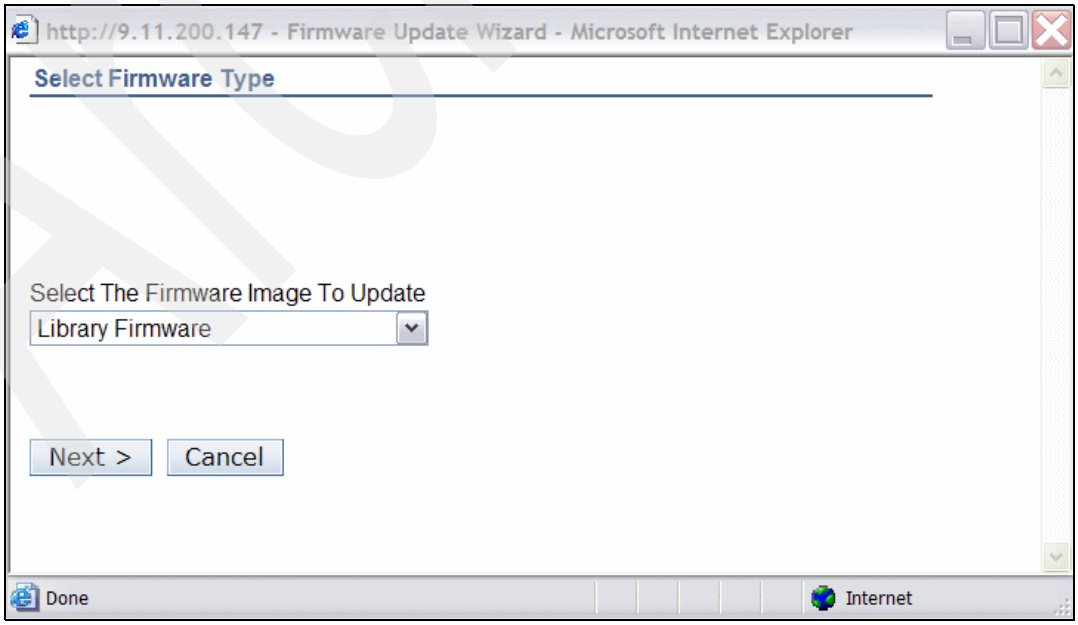


Figure 14-7 Selecting the firmware image to update

5. Select **Library Firmware** as the firmware image to update and click **Next**.

   The panel shown in Figure 14-8 displays.

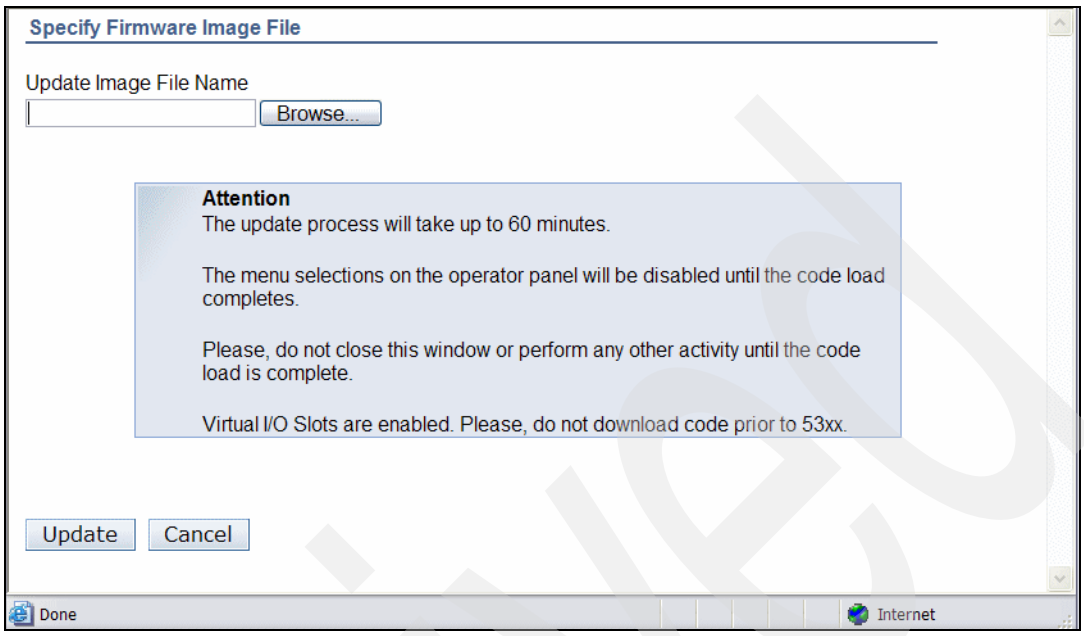


Figure 14-8 TS3500 Web GUI Library Firmware update

6. Enter or browse for the name of the update image file. Then, click **Update.**

   As indicated in the Attention box in Figure 14-8, the update can take up to 60 minutes to complete.

In the past, when library firmware required an update, scheduling downtime for the IBM 3584 Tape Library was necessary. Job flow was interrupted and productivity was reduced. The library now offers a nondisruptive library firmware update that either the service support representative (SSR) can perform by using the Customer Engineer Tool software application, or that you can perform by using the Tape Library Specialist Web interface. Certain levels of library and drive firmware are required. Table 14-3 lists the levels of firmware that are necessary for a nondisruptive library firmware update.

Table 14-3 TS3500 nondisruptive firmware levels

Device	Level of firmware
3584 Tape Library	6050
Ultrium 3 Tape Drives	62R0
Ultrium 2 Tape Drive	6750
TS1130 Tape Drive	All GA Levels
TS1120 Tape Drive	16E4
3592-J1A Tape Drive	0828

Upgrading the TS1120 firmware

The TS1120s on our library were not at the correct firmware level to support encryption, so we had to upgrade them to the required level. The drives for our logical library are in Frame 1, Rows 9 and 10. See Figure 14-9 on page 463. The process is the same for TS1130 drives.

Drive VPD

Location	Drive Type	Firmware Version	Serial Number
Frame 1, Row 1	03592E05	1935	0001350248
Frame 1, Row 2	03592E05	1921	0001365055
Frame 1, Row 3	03592E05	1942	0001365033
Frame 1, Row 4	03592E05	1935	0001350266
Frame 1, Row 5	03592E05	1935	0001350570
Frame 1, Row 6	03592E05	1931	0001365060
Frame 1, Row 7	03592E05	1949	0001350573
Frame 1, Row 8	03592E05	1942	0001365117
Frame 1, Row 9	03592E05	1935	0001365102
Frame 1, Row 10	03592E05	192B	0001365109
Frame 1, Row 11	03592E05	1942	0001350608
Frame 1, Row 12	03592E05	1942	0001350420

Figure 14-9 Drive Vital Product Data (VPD) reflects the installed firmware versions

Updating drive firmware

Updating the drive firmware is a fairly simple process. It is approximately 30 minutes from start to completion. We obtained the firmware from the following IBM Web site and downloaded it to our personal computer:

http://www-1.ibm.com/support/docview.wss?rs=564&context=STCTNLZ&dc=D420&uid=ssg1S4000317&loc=en_US&cs=utf-8&lang=en

Expand the **Service Library** folder and select **Firmware Update,** which displays Figure 14-10.

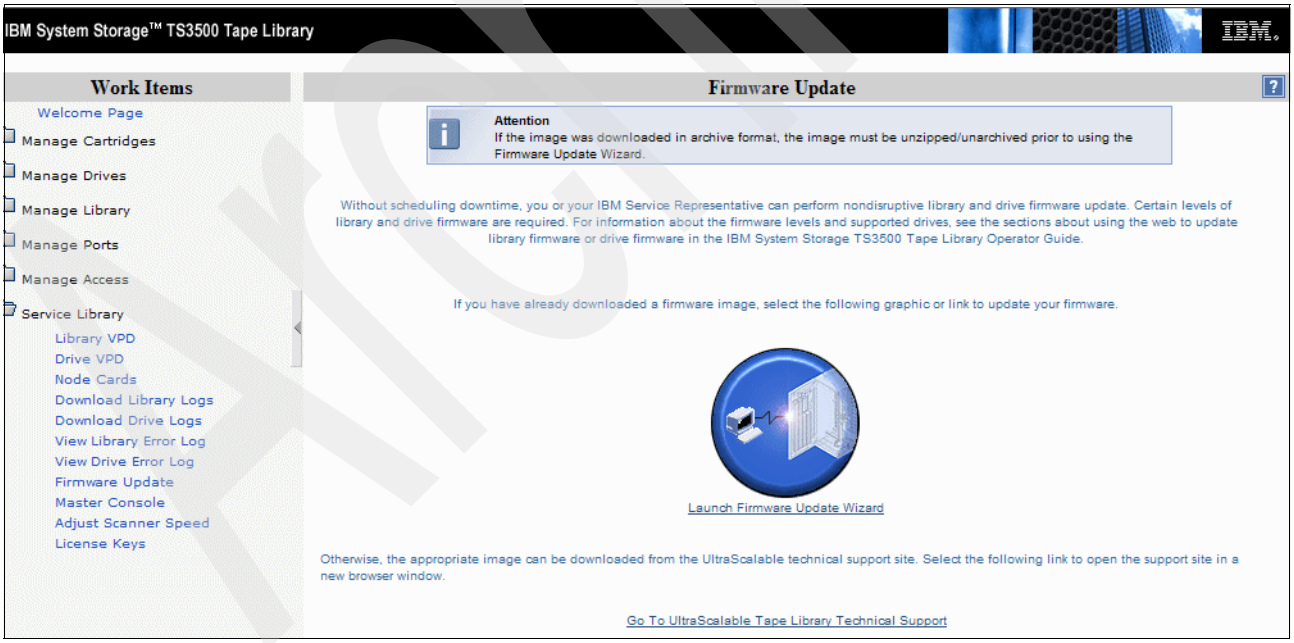


Figure 14-10 How to update drive firmware

In the past, a firmware update to a drive that was performed through the library was disruptive to the host server, because the operator had to stop host jobs that were queued to the drive

and take the drive offline. The library now provides two nondisruptive options for updating drive firmware:

- Activate when drive is empty
- Activate when drive is reset

These options do not affect the host. Activate immediately may affect the host if the drive is performing read/write operations.

To use the Tape Library Specialist Web interface to update firmware for drives in the 3584 Tape Library:

1. Select **Service Library** → **Firmware Update**. The Firmware Update panel displays.
2. Select the **Launch Firmware Update Wizard**, and then select the choice to update a drive. The Specify Firmware Image File panel displays as shown in Figure 14-11.

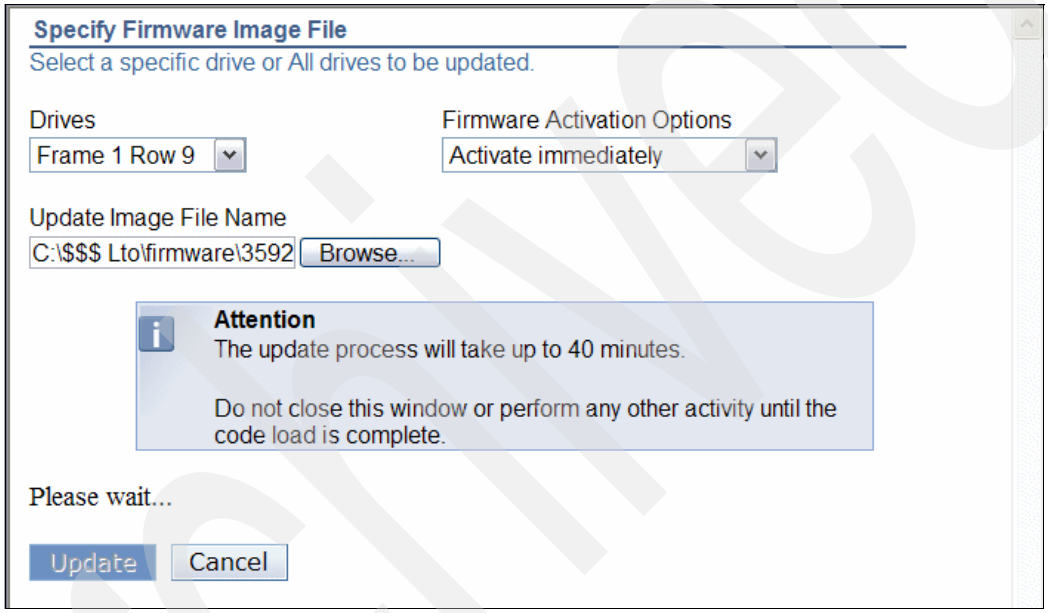


Figure 14-11 Drive firmware update window

3. Select one or more drives that you want to update, and then select one of the Firmware Activation options in Table 14-4.

Table 14-4 Firmware activation options

Option	Description
Activate immediately	Performs the update as soon as the code is transferred to the drive.
Activate when drive is empty	Performs the update only after the host has demounted the cartridge from each specified drive
Activate when drive is reset	Performs the update after the next time the drive resets. The drive may be reset using the Web UI.

4. Select **Update.**

Note: Only the *Activate immediately* option is available to the Ultrium 1 Tape Drive, which cannot be defined as a control path drive. The *Activate when drive is empty* option is supported for control path drives.

14.5.3 Add EKM or TKLM IP addresses

To set the IP address that the library will use, select **Manage Access** → **Key Manager Addresses.** The Key Manager Addresses panel opens, as shown in Figure 14-12.

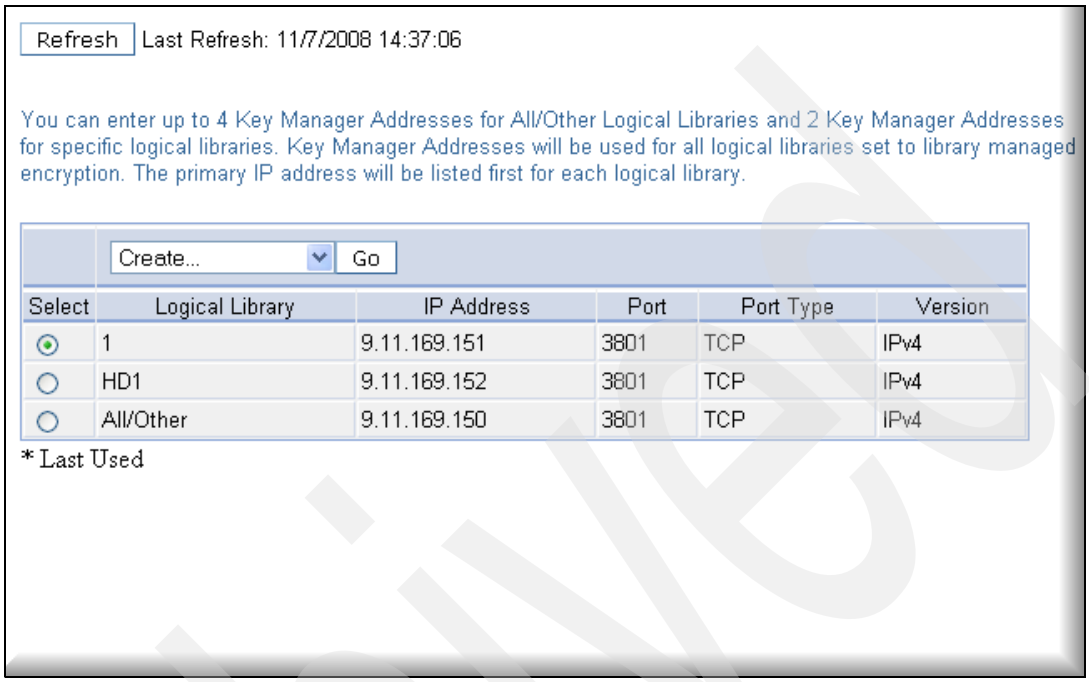


Figure 14-12 Setting Key Manager (KM) IP addresses

The order in which the Key Manager (KM) IP addresses are used by logical library is: If the first EKM or TKLM in the list is not available, the library fails over to the next EKM or TKLM, and on to the All/Other KMs if necessary. This is considered a failover process.

Note: Failover only occurs if a Key Manager (KM) cannot be contacted, for example, when it is not started. Failover does not occur if the EKM or TKLM is running but has an operational problem, for example, a configuration file setting is incorrect.

After a KM is contacted, it is at that point that we look for the correct label or Hash in the keystore. If these values are not in the keystore of the first contacted EKM or TKLM, the request fails. Although the Label or Hash might exist in the keystore of the next EKM or TKLM, the request does *not* fail over to the next EKM or TKLM. You have to ensure that all keystores are alike, so that this does not occur.

Setting the KM address is very straightforward. Select the **Select Action** list box, and select **Create.** At this point, you can enter the IP address of your KM along with the port address. The port address has to match the port address specified in the KM configuration. The data in Example 14-2 on page 466 is not intended to be all inclusive, because we have included only a few statements around the **`TransportListener.cp.port=3801`** parameter, which is the value that we refer to with this panel. If you have to see the entire EKM configuration file that we used, refer to Example 6-35 on page 191.

TKLM also defaults to listening on tcp port 3801. The port may be changed with the TKLM GUI under **Tivoli Key Lifecycle Parameters** → **Configuration** → **Key Serving Parameters**.

Example 14-2 Short edited version of EKM configuration

```
# listconfig
Audit.eventQueue.max=0
TransportListener.tcp.port=3801
TransportListener.ssl.truststore.name=tonyskeys.jcks
```

The parameter **TransportListener.tcp.port** is the port that the EKM server listens on for requests from the tape drives. Although this is a required parameter, you might notice that we used the default value that came with the sample configuration.

14.5.4 Enable Library-Managed Encryption

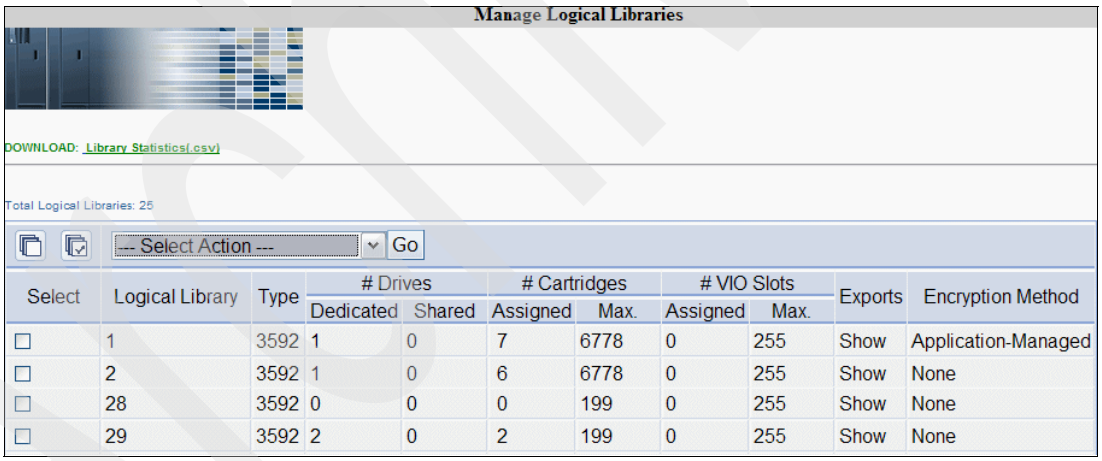
Now that you are at the correct levels of both drive firmware and library firmware, you are ready to enable Library-Managed Encryption (LME). In review, the basic steps are to:

1. Enable encryption for the logical library.
2. Modify the logical library EKM address panel to point to your EKM for that logical library.
3. Set up a Barcode Encryption Policy (BEP) or Internal Label Encryption Policy (ILEP).
4. Verify that the library can communicate with your EKM.

Enable encryption for the logical library

To enable encryption for your logical library:

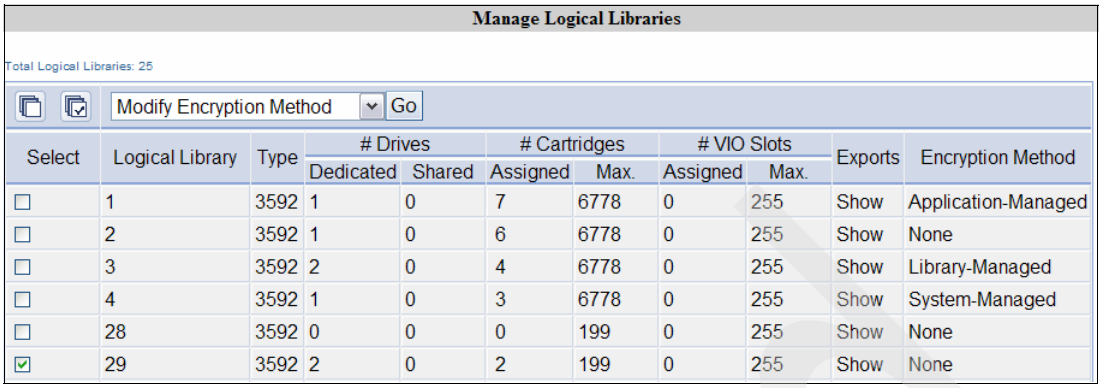
1. Select **Manage library** → **by logical library**, which results in the Manage Logical Libraries Panel as shown in Figure 14-13.



Select	Logical Library	Type	# Drives		# Cartridges		# VIO Slots		Exports	Encryption Method
			Dedicated	Shared	Assigned	Max.	Assigned	Max.		
☐	1	3592	1	0	7	6778	0	255	Show	Application-Managed
☐	2	3592	1	0	6	6778	0	255	Show	None
☐	28	3592	0	0	0	199	0	255	Show	None
☐	29	3592	2	0	2	199	0	255	Show	None

Figure 14-13 Setting Encryption method by logical library

2. From the Manage Logical Libraries selection panel, select **Modify Logical Library** from the **Select Action** pull-down menu, check the logical library to modify, and click **Go**.
3. As shown in Figure 14-14 on page 467, we use logical library 29, so we checked it, selected **Modify Encryption Method** from the list box, and then clicked **Go**.



Select	Logical Library	Type	# Drives		# Cartridges		# VIO Slots		Exports	Encryption Method
			Dedicated	Shared	Assigned	Max.	Assigned	Max.		
☐	1	3592	1	0	7	6778	0	255	Show	Application-Managed
☐	2	3592	1	0	6	6778	0	255	Show	None
☐	3	3592	2	0	4	6778	0	255	Show	Library-Managed
☐	4	3592	1	0	3	6778	0	255	Show	System-Managed
☐	28	3592	0	0	0	199	0	255	Show	None
☑	29	3592	2	0	2	199	0	255	Show	None

Figure 14-14 Modifying Encryption Method for logical library 29

The Encryption Method panel opens, as shown in Figure 14-15.

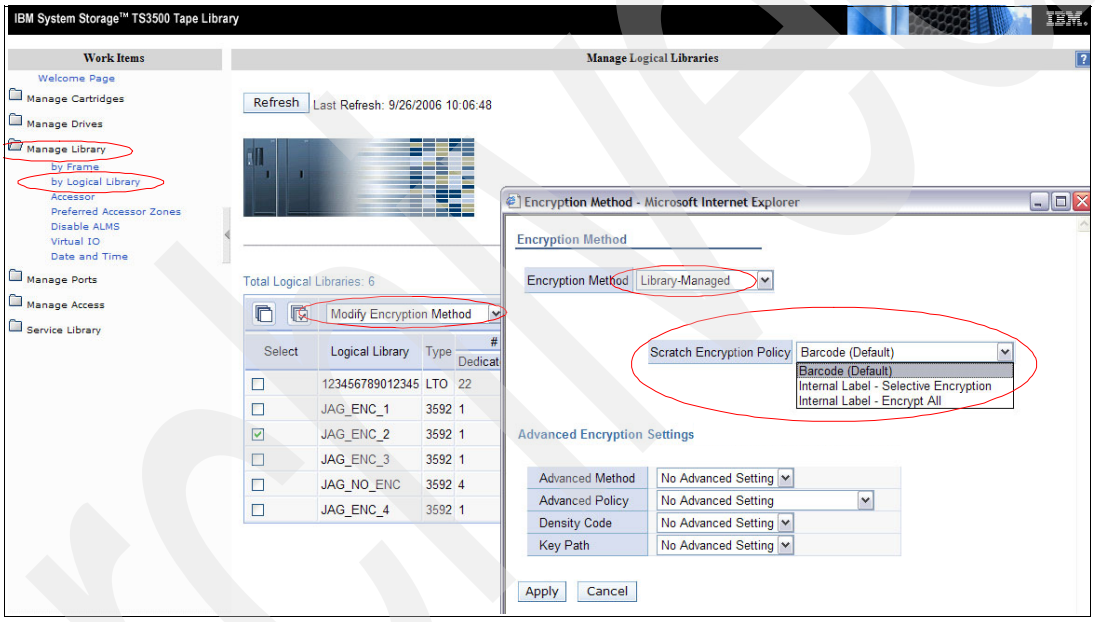


Figure 14-15 Encryption Method selection panel

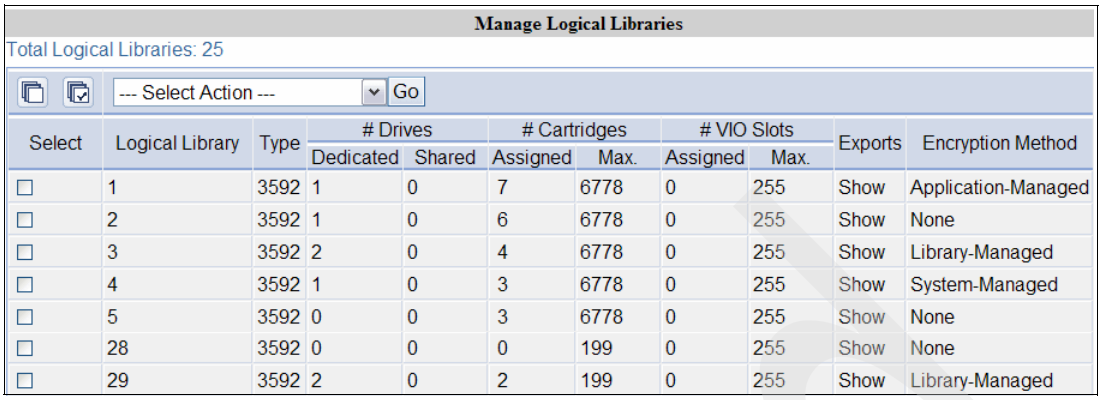
4. Choose the Encryption Method for the logical library with which you are working. Because we are setting up LME, select **Library-Managed**.

> **Note:** On this panel you may also select a different encryption method: either System-Managed and Application-Managed. We discuss these settings in their corresponding sections in this book

Select an encryption policy type to use, as follows, and then click **Apply**:

- Barcode (default)
- Internal Label - Selective Encryption
- Internal Label - Encrypt All

5. After refreshing the panel (In Figure 14-16 on page 468), you see that logical library 29 now has Library-Managed displayed in the Encryption Method column.



Manage Logical Libraries

Total Logical Libraries: 25

--- Select Action --- Go

Select	Logical Library	Type	# Drives		# Cartridges		# VIO Slots		Exports	Encryption Method
			Dedicated	Shared	Assigned	Max.	Assigned	Max.		
☐	1	3592	1	0	7	6778	0	255	Show	Application-Managed
☐	2	3592	1	0	6	6778	0	255	Show	None
☐	3	3592	2	0	4	6778	0	255	Show	Library-Managed
☐	4	3592	1	0	3	6778	0	255	Show	System-Managed
☐	5	3592	0	0	3	6778	0	255	Show	None
☐	28	3592	0	0	0	199	0	255	Show	None
☐	29	3592	2	0	2	199	0	255	Show	Library-Managed

Figure 14-16 Library-Managed Encryption panel

14.5.5 Barcode Encryption Policy

After enabling Barcode Encryption Policy for Library-Managed Encryption for the first time, you must create a Barcode Encryption Policy (BEP), even if you are encrypting all cartridges.

Note: Previously, the term Scratch Encryption Policy (SEP) referred to the only encryption policy available for the LME method. SEP uses a volume's VOLSER to determine the encryption policy. A new LME policy called Internal Label Encryption Policy (ILEP) has been added. This policy also relates to scratch volumes. Thus, SEP now refers in general to the various LME policies. A new term, Barcode Encryption Policy (BEP), is now used to refer to the policy previously known as SEP.

A *Barcode Encryption Policy* is a range of cartridge VOLSERs and specifies which scratch cartridges to encrypt on the next attempt to write from the beginning of the tape; it does not indicate which cartridges are currently encrypted. When used with LME, a Barcode Encryption Policy controls cartridge encryption by VOLSER ranges in *all* logical libraries for which Barcode Encryption Policy was chosen. Stated another way, if you have defined multiple policies (overlap not allowed), they all are in effect simultaneously, and they affect which cartridges are encrypted in every defined library-managed logical library in a TS3500 that selected Barcode Encryption Policy.

When implementing a Library-Managed BEP, you not only specify which cartridges to encrypt but also the public keys that are used to encrypt the data key. You manage this by directing future allocations to specific Barcode Encryption Policies depending on the cartridge destination. Using Table 14-5 on page 469 as an example of the BEPs that we have built in our test environment, we create a few encrypted tapes intended for different business partners.

For encrypted data that is to be sent to business partner `PART2`, we direct allocations to tape VOLSERs ABC100 - ABC199 to ensure that one of the EEDKs created on the tape can be decrypted by PART2's private key, allowing the business partner to access the data key (DK) and decrypt the data on the cartridge.

Sending an encrypted cartridge to business partner `PART1` dictates that we direct an allocation to tape VOLSERs ABC000 - ABC099, ensuring that EEDK 1 is built using business partner PART1's public key. This allows PART1's EKM at PART1's location to decrypt this EEDK using PART1's private key to then retrieve the DK so that the data can be decrypted by the TS1120 drive.

Table 14-5 Barcode Encryption Policy table

Barcode Encryption Policies (BEP)			
BEP	**VOLSER sequence**	**Key label #1[a]**	**Key label #2**
BEP #1	ABC000 - ABC099	Business partner PART1	Our key label
BEP #2	ABC100 - ABC199	Business partner PART2	Our key label
BEP #3	ABC200 - ABC299	Business partner PART3	Our key label
BEP #4	ABC300 - ABC399	Business partner PART4	Our key label

a. Key label can be in clear text or HASH format.

Creating a Barcode Encryption Policy

The Barcode Encryption Policy is nothing more than a sequence of VOLSERs specified to the library. To view, change, or add to these policies, from the TS3500 Web GUI, select **Manage Cartridges** → **Barcode Encryption Policy**. The panel in Figure 14-17 opens.

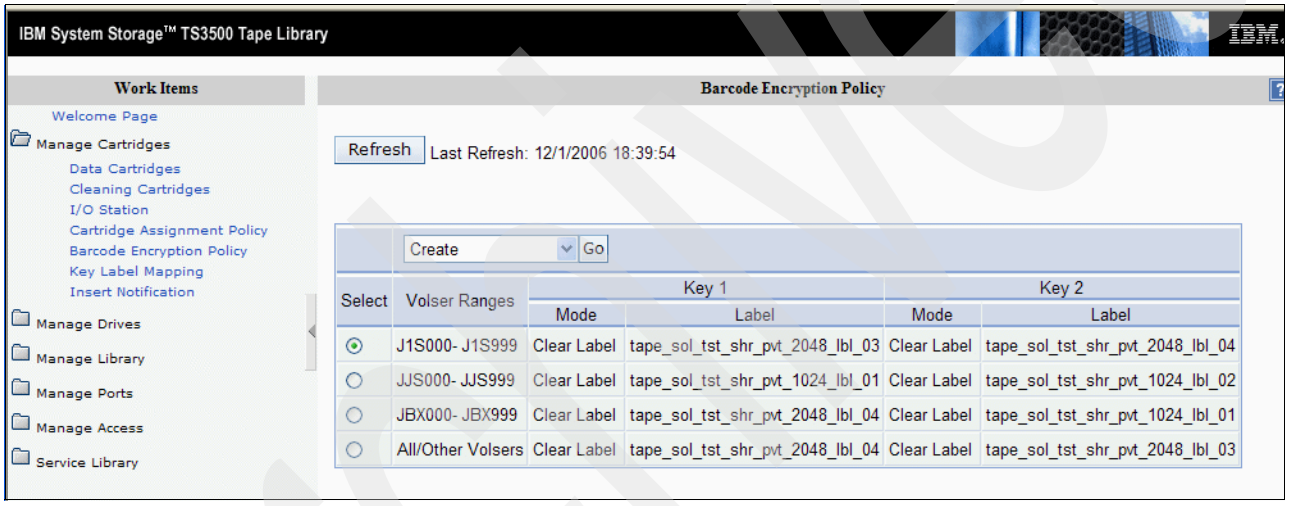


Figure 14-17 Barcode Encryption Policy GUI

Note: Only cartridge VOLSERs specified in defined Barcode Encryption Policies are encrypted.

On this panel, you may modify, create, or delete a VOLSER range. Let us look at how to create a Barcode Encryption Policy.

Select **Create** from the list box and click **Go** in the panel shown in Figure 14-17. Figure 14-18 on page 470 displays.

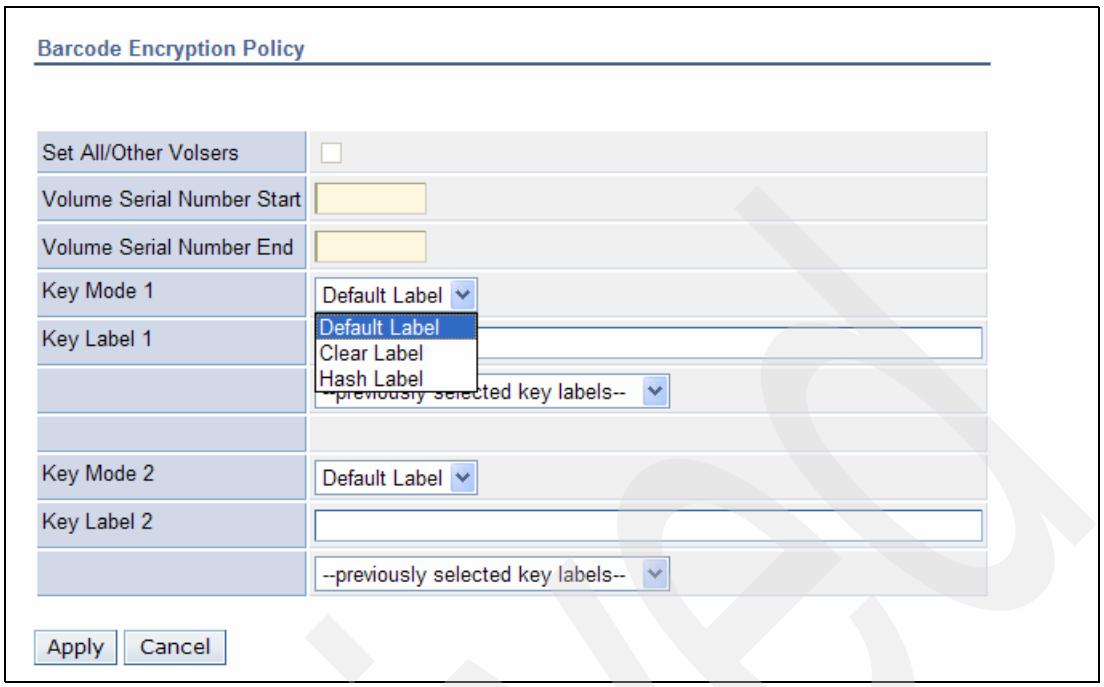


Figure 14-18 Barcode Encryption Policy panel

At this panel, for the fields **Volume Serial Number Start** and **Volume Serial Number End**, you enter the beginning and ending VOLSERs of the Barcode Encryption Policy for the BEPs that you want to create. If you try to enter a range that overlaps with a previously entered range, you receive an error and the BEP is not created.

There are three key modes that you can select. The *key mode* refers to the method by which the Externally Encrypted Data Keys (EEDKs) that are created on the cartridge are referenced:

Default Label The label was configured at the Encryption Key Manager and is specified within the EKM configuration file.

Clear Label The Externally Encrypted Data Key (EEDK) is referenced by the specified key label.

Hash Label The EEDK is referenced by a computer value, which corresponds to the public key that is referenced by the specified key label.

The Default Label is self-explanatory; however, we have to explain Clear Label and Hash Label.

When a certificate is imported into a keystore, a label is assigned to it. It is this label that you enter on this panel. When an encrypted cartridge is created, two EEDKs are also created and stored on the cartridge. In addition to the encrypted Data Key, each EEDK has a pointer to the public key that was used to encrypt the Data Key. The pointer can either be the label of the certificate or a Hash value based on the public key. Regardless of which is used, EKM uses the value to identify the correct private key to use to decrypt the encrypted data key.

Let us see how this might work in your environment. You decide that you want to encrypt cartridges sent to your business partner (BP); so, how do you proceed?

To create the BEP:

1. Contact your BP to obtain a certificate with the BP's public key in it.
2. Import this certificate into your keystore.
3. The import process assigns a label to the certificate, which you can consider as *My BP's certificate*.
4. Because you are implementing LME, you have to create a BEP, so that you can send encrypted cartridges to your business partner.
5. Create the BEP by specifying the `VOLSER range = BP001-BP100` as follows:
   a. Key1 mode = clear label and label = *My BP's certificate*, which corresponds to the value you specified when you imported this BP's certificate into your keystore.
   b. Key2 mode = clear label and label = *My certificate*.

The BEP is created.

You are now ready to create an encrypted cartridge to send to your BP, as follows:

1. You write data to cartridge BP001. The data is encrypted, and two EEDKs are placed on the cartridge with the data:

   EEDK1 Encrypted DK and clear label version of "My BP's certificate".

   EEDK2 Encrypted DK and clear label version of "My certificate".
2. Demount the cartridge.

You decide that prior to sending the cartridge, you want to verify that you can access the data, so the cartridge is mounted. The process is:

1. The tape drive sends both the EEDKs to the EKM.
2. The EKM unwraps both EEDKs and, using the label within each EEDK, finds the correct certificate, which contains the private key required to decrypt the encrypted DK.
3. The EKM finds certificate labeled *My Business Partner*, but it does not contain a private key, so EKM continues.
4. The EKM finds a certificate labeled *My certificate*, finds the private key, decrypts the DK, and sends that to the drive.
5. The drive uses DK to decrypt data and passes the decrypted data to your application, which reads it without issue.

Because you were able to read the cartridge at your site, you decide the cartridge is encrypted correctly and you send it to your business partner. To read the encrypted cartridge, the business partner proceeds. The steps in the following process can result in an error:

1. The cartridge is mounted on an encryption-enabled drive.
2. The tape drive sends both of the EEDKs to the EKM.
3. The EKM unwraps both and using the label within each attempts to find the corresponding certificate that contains the private key required to decrypt the encrypted DK.
4. The EKM looks for a certificate labeled *My Business Partner*, cannot find it, so it continues with the next EEDK.
5. The EKM looks for a certificate labeled *My certificate* and cannot find that one either.
6. The EKM passes an error code to the drive stating that the certificate is *not* in the keystore.
7. The application that is attempting to read the cartridge fails.

What happened, and how do you ensure that this does not happen again? The previous example is an error case where Hash should have been specified. You are now ready to investigate using the Hash label as opposed to the clear label.

Creating a BEP using Hash values

Why was your business partner unable to read the cartridge? A certificate that contained the correct private key did exist in the business partner's keystore; it is simply that the key manager was not able to find it. Why was the key manager unable to find it? You told the key manager to look for a certificate labeled *My BP's certificate*, and there simply was not a certificate in your BP's keystore with this label.

What questions to ask

Do you see the problem now? How do you ensure that you and your BP both label certificates with the same name? This issue does not seem too difficult if you have one or two BPs, because you can contact each BP and agree on a name, but what if you have several hundred BPs? What if you have a BP, whom you have never contacted, and to whom you now have to send an encrypted cartridge? What if you misspell the label when entering it?

Where Hash values are used

Within each certificate are several fields. One of these fields is named Subject Key Identifier (SKI). The content of the SKI field is a Hash value representing the public key contained within the certificate. Let us see how this is used when specified in the BEP:

1. You write data to cartridge BP001, the data is encrypted, and two EEDKs are placed on the cartridge with the data:

   EEDK1 Encrypted DK and HASH label of "My BP's certificate".
   EEDK2 Encrypted DK and Clear label version of "My certificate".

2. Demount the cartridge.

Now, when the BP receives the cartridge, the following occurs:

1. The cartridge is mounted on an encryption-enabled drive.
2. The tape drive sends both of the EEDKs to the EKM.
3. The EKM unwraps both EEDKs and, using the label within each EEDK, attempts to find the corresponding certificate that contains the private key required to decrypt the encrypted DK.
4. The EKM realizes that for EEDK1, it has a Hash value and not a certificate label, so the EKM compares this value to the SKI field in each certificate until the key manager finds a match or reaches the end of the keystore. Finding the match, the key manager then uses the private key in the certificate to decrypt the encrypted DK. The key manager then sends the DK to the drive.
5. The drive uses the DK to decrypt the cartridge data and passes the decrypted data to your application, which uses it without issues.

Barcode Encryption Policies for scratch cartridges

The BEP Web GUI page allows you to create, change, or delete an encryption policy for scratch cartridges. A *scratch cartridge* is a labeled cartridge that is blank or contains no valid data. A Barcode Encryption Policy allows the library to identify to an IBM TS1120 or TS1130 Tape Drive which scratch cartridges to encrypt on the next attempt to write from the beginning of the tape.

The following information also displays on this panel:

Refresh A button that allows you to redraw the contents of the page. You can select Refresh at any time.

Select Action A drop-down list box that allows you to select actions to perform. After you select a range of scratch cartridges by their volume serial (VOLSER) numbers, select the action that you want to perform on them, then click Go. Choices include:

- **Create** Allows you to create a Barcode Encryption Policy for the range of scratch cartridges that you specify.
- **Modify** Allows you to change the VOLSER range, key labels, or modes in the policy that you selected.
- **Delete** Allows you to remove the policy that you selected. If only one Barcode Encryption Policy exists, the library does not allow you to delete it.

Volser Ranges The range of cartridges to be encrypted from the beginning of the tape.

Key 1 Label An alias for an encryption key (cipher). It is used by the Encryption Key Manager software.

Key 1 Mode For Key 1, the method by which an Encryption Key Manager identifies the public-private keys that were used to encrypt it.

Creating a Barcode Encryption Policy

To create a Barcode Encryption Policy:

1. From the **Select Action** drop-down list box, select **Create** and click **Go**.
2. In the pop-up menu, check **Set All/Other Volsers** or enter the volume serial numbers of the scratch cartridges that begin and end the range to which you want to assign the Barcode Encryption Policy.
3. Enter two key labels for the policy. You can select an existing key label from each drop-down list box, or you can enter a new key label.
4. Select two key modes for the policy. The *key mode* is the method by which public-private key pairs encrypt a data key to form an Externally Encrypted Data Key. Values are:

   Default Label The label was configured at the Encryption Key Manager.

   Clear Label The Externally Encrypted Data Key (EEDK) is referenced by the specified key label.

   Hash Label The EEDK is referenced by a computer value, which corresponds to the public key that is referenced by the specified key label.

5. Click **Apply**.

Changing a Barcode Encryption Policy

To change a Barcode Encryption Policy:

1. In the **Select** column, select the policy that you want to edit.
2. From the **Select Action** drop-down list box, select **Modify,** and click **Go**.
3. In the pop-up menu, enter the volume serial numbers of the scratch cartridges that begin and end the range to which you want to assign the Barcode Encryption Policy.
4. Enter two key labels for the policy. You can select an existing key label from each drop-down list box, or you can enter a new key label.

5. Select two key modes for the policy. Values are:

   Default Label The label was configured at the Encryption Key Manager.

   Clear Label The Externally Encrypted Data Key (EEDK) is referenced by the specified key label.

   Hash Label The EEDK is referenced by a computer value, which corresponds to the public key that is referenced by the specified key label.

6. Click **Apply**.

Deleting a Barcode Encryption Policy

To delete a Barcode Encryption Policy:

1. In the **Select** column, select **All/Other Volsers** or the VOLSER range of the policy that you want to remove.
2. From the **Select Action** drop-down list box, select **Delete** and click **Go**.
3. A pop-up menu asks you to confirm the deletion.
4. Click **OK** to delete the policy.

14.5.6 Testing encryption

In this section, we run an encryption test using Library-Managed Encryption (LME). Figure 14-19 is a view of the cartridges that exist in our library prior to encryption. We run our test using cartridge VOLSER 447AAF. The window shows the VOLSER together with the media identifier JA (447AAFJA). The column on the right in the example reflects that the status of the cartridge is Not Encrypted.

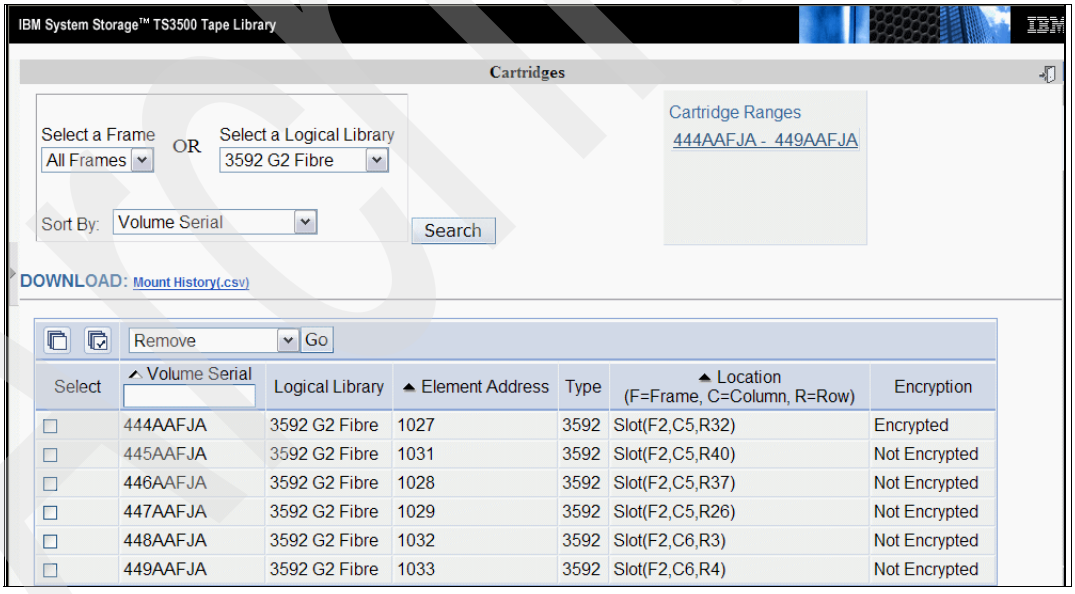


Figure 14-19 Cartridges prior to encryption

We selected and mounted 447AAF, and using **`tapeutil`**, we wrote to the cartridge. We closed and demounted the cartridge. Looking at Figure 14-20 on page 475, note that the cartridge status has changed to Encrypted.

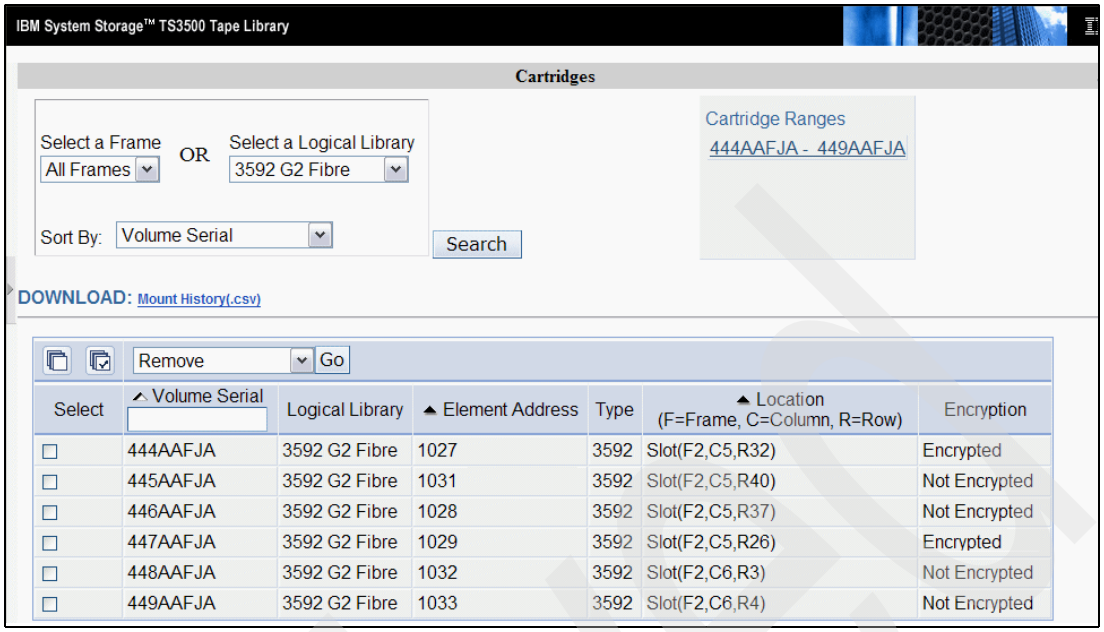


Figure 14-20 Cartridge after encryption

In the EKM audit log, you find an entry such as the entry shown in Example 14-3.

Example 14-3 Audit log entry

```
Runtime event:[
  timestamp=Mon Oct 23 12:14:02 EDT 2006
  event source=com.ibm.keymanager.c.fb
  outcome=[result=successful]
  event type=SECURITY_RUNTIME
  resource=[name= Drive Serial Number: YN1B00001388 WWN: 5005076300020216 VolSer:
447AAF Key Alias/Label[0]: cert1 Key Alias/Label[1]: cert2;type=file]
  action=stop
```

The audit log is the only place where you can see which keys were used to encrypt a cartridge. Refer to 14.3.2, “Keeping track of key usage” on page 454 for more details about the audit log.

14.6 Implementing System-Managed Encryption

Implementing System-Managed Encryption (SME) involves changes implemented at the system and library level that are transparent to the application as shown in Figure 14-21 on page 476. Although we describe SME implementation for the TS3500 Tape Library, be aware that SME for Open Systems is also supported with TS1120 or TS1130 tape drives installed in an IBM 3494 Tape Library or IBM TS3400 Tape Library

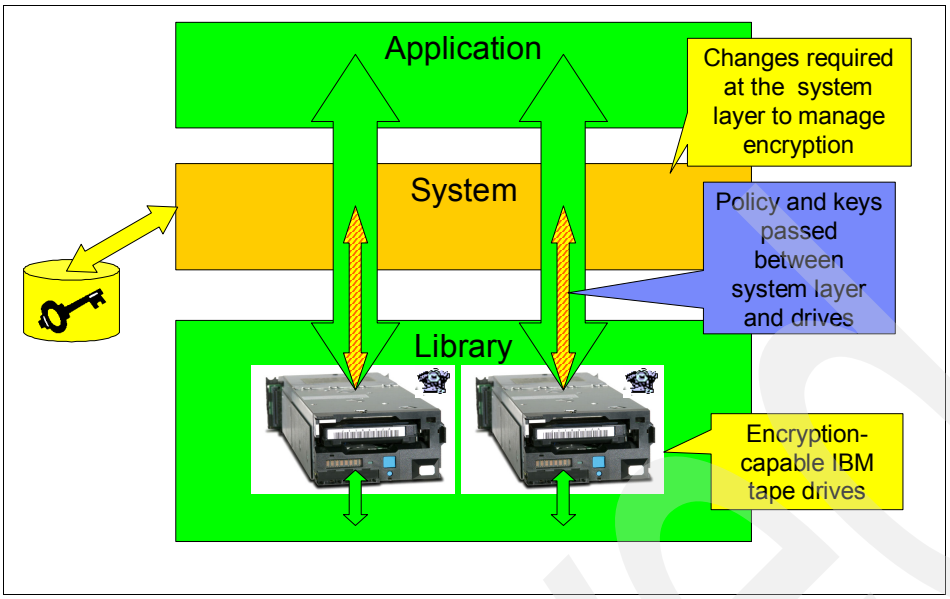


Figure 14-21 Policy, keys, and changes overview

Encryption policies are implemented at the system level using Atape on the AIX 5.1, 5.2, or 5.3 operating system. Figure 14-22 shows the data flow using the Atape Device Driver.

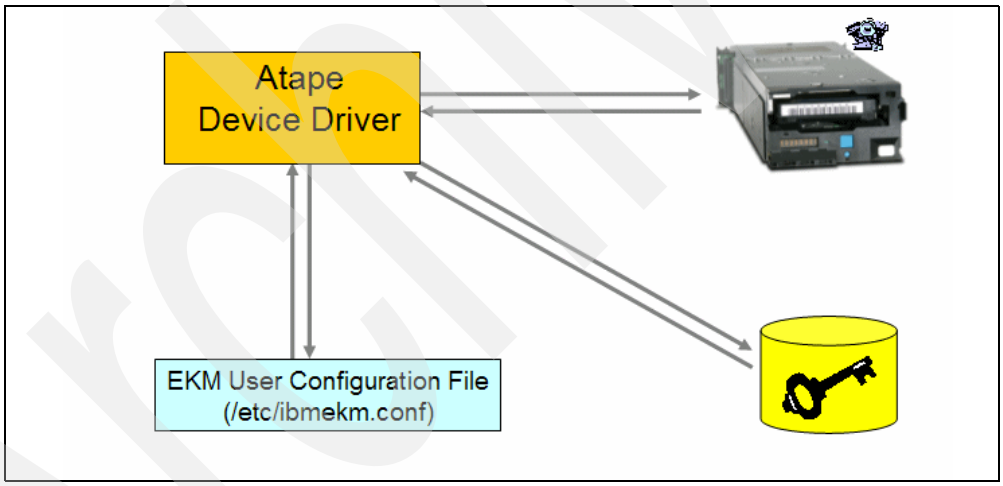


Figure 14-22 Key and policy flow in the SME environment

14.6.1 System-Managed Encryption tasks

Performing SME on the IBM TS1120 or TS1130 Tape Drive requires:

- Ensure that TS1120 or TS1130 tape drives and TS3500, IBM 3494 or TS3400 Tape Library are *encryption-capable*.
  See “Verify library firmware levels” on page 459 for more information.
- Install, verify, and configure:
  - Keystore creation. See “Creating a keystore using keytool” on page 180.
  - IBM EKM component. See 6.2, “Installing EKM on AIX” on page 177.
- Ensure that encryption is enabled on TS1120 (3592-E05) or TS1130 (3592-E06 or 3592-EU6) tape drive.

You may do this by using the TS3500 Web GUI or as an IBM Service task. Refer to“Enable encryption for the logical library” on page 466 to enable encryption using the Web GUI. Also refer to *IBM System Storage TS3500 Operator’s Guide,* GA32-0560, or *IBM System Storage TS3500 Tape Library Operator Guide with ALMS,* GA32-0594 for configuring the TS1120 or TS1130 tape drive on the TS3500 Tape Library.

- Install or update Atape device driver. See “Atape device driver” on page 477:

  Atape Encryption support is included at Version 10.2.8.0 for AIX 5.1, 5.2, or 5.3.
- Update the entry in the Atape EKM proxy configuration file with the EKM or TKLM IP address and port.
- Update entries in the Atape device driver configuration for each tape drive (rmt*x*) that is used for SME:
  - Use System Encryption Fibre Channel Protocol (FCP) Proxy Manager.
  - System Encryption for Write Commands at BOP.
- Use **`tapeutil`** functions to verify EKM or TKLM paths and encryption configuration.

14.6.2 Atape device driver

Example 14-4 shows how to verify that you are at the correct Atape device driver version by issuing the AIX **`lslpp`** command.

Example 14-4 Verify the Atape version that is installed

```
lslpp -l Atape.driver
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  Atape.driver              10.3.2.0  COMMITTED  IBM AIX Enhanced Tape and
                                                 Medium Changer Device Driver
```

As you can see, we are at licensed internal code (LIC) level 10.3.2.0 and the level required for encryption is 10.2.8.0, so we are ready, and we continue with 14.6.3, “Update Atape EKM proxy configuration” on page 478. For those of you who are not at the correct LIC level or who have to install Atape, refer to the next section.

Atape installation procedure

Enter the following command to list the currently installed `Atape.driver` version:

```
lslpp -l Atape.driver
```

If you have the tape device drivers and the SMI-S Agent CD, use the following instructions to install the device driver:

1. Place the CD into the CD-ROM drive on your AIX system.
2. Mount the CD over an empty directory. For example, if your CD-ROM drive is defined at /dev/cd0 and you have an empty directory at /cdrom, issue the following command to mount the CD:

   ```
   mount -frv cdrfs /dev/cd0 /cdrom
   ```

   You can create an empty directory using the **`mkdir`** command, for example:

   ```
   mkdir /cdrom
   ```

   Subsequent instructions assume that you have mounted the CD at mount point /cdrom.
3. Enter the following command:

```
cd Drivers/cdrom/AIX
```

4. Consult the Atape.Readme file for any important information pertaining to the device driver. Information in this file takes precedence over information in the manual.
5. Execute the install_atape script. This script uninstalls any previous versions of Atape, installs and commits the latest version of Atape, and then runs **cfgmgr** to define your devices.
6. Enter the following command:

```
cd/ unmount/cdrom
```

7. Remove the CD from the CD-ROM drive and store it in a safe place.

14.6.3 Update Atape EKM proxy configuration

After installing the device driver, the addresses of the key managers have to be made known (configured) to Atape. The servers are configured in a text file called ibmekm.conf, which is installed in the /etc directory by the device driver.

This step consists of changing the configuration file that Atape uses to communicate with EKM or TKLM. We change an entry in this file that consists of the Encryption Key Manager IP address. Refer to Example 14-5.

Example 14-5 Atape Encryption Key Manager configuration file

```
#  IBM Encryption Key Manager Configuration File
#
#  (C) COPYRIGHT International Business Machines Corp. 2006
#  All Rights Reserved
#  Licensed Materials - Property of IBM
#
#  US Government Users Restricted Rights - Use, duplication or
#  disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#  This file contains the TCP/IP address(es) and port(s) for the Encryption Key
#  Server with a configuration entry in the following formats. The IPv4 address
#  entered as x.x.x.x:port. The IPv6 address entered as x:x:x:x:x:x:x:x port.
#  The server is for information only and is not used. The timeout value is
#  specified in seconds.
#
#  The format for IPv4 address:
#  server timeout address:port
#  for example,
#  ekmtest  10  9.12.123.1234:8050
#
#  The format for IPv6 address:
#  server timeout address port
#  for example,
#  ekmtest  10  fe80::207:30ee:edcb:d05d  8050
#
#  The Encryption Key Server address and port can be a local loop back
#  address 127.0.0.1:port in IPv4 format or ::1 port in IPv6 format if the server
#  is on the same host or a network address and port if external to the host.
#  Up to 16 server address and port entries are supported if there are multiple
#  TCP/IP connections to the same server and/or multiple servers.
#
# Interoperability between IPv4 and IPv6 versions running on dual-stack hosts:
```

```
#    IPv4 Client <--> IPv4/IPv6 Server    using IPv4 address for EKM server
#    IPv6 Client <--> IPv4 Server         using IPv4 address for EKM server
#    IPv6 Client <--> IPv6 Server         using IPv6 address for EKM server
#
#  Sample entry for a local server with a 10 second timeout using port 8050
#  in IPv4 format
ekmtest   10   127.0.0.1:8050
#
#  in IPv6 format
#  ekmtest   10   ::1   8050
ekmonp 10 9.82.26.27:3801 <<<---------
```

In Example 14-5 on page 478, you see the value that we added to the configuration file. The field is defined as:

```
Server - Timeout value - IPv4 address:port (for IPv4)
Server - Timeout value - IPv6 address - port (for IPv6)
```

Because we are using IPv4, the values are:

```
Server = ekmonp
Timeout value = 10 seconds
IPv4 address:port = 9.82.26.27:3801
```

The server value is used only for testing purposes for EKM or TKLM connectivity, which you can see in 14.6.8, “Using tapeutil functions to verify EKM paths” on page 490.

The timeout value in seconds is used when a request is sent to EKM or TKLM. If a response is not received within the time set, Atape fails over to the next EKM or TKLM defined and redrives the request. If EKM or TKLM is on the same server as Atape, set this value low, at perhaps 10 seconds.

However, if EKM or TKLM and Atape are on separate servers and connected using a network that is *perhaps very busy at times*, increase this value to something more appropriate, such as 60 seconds.

The IP address is the IP address where EKM or TKLM is located, and the port number is the port that EKM or TKLM uses to listen for traffic from tape drives. The port value has to be the same as specified in the EKM configuration file on the `TransportListener.tcp.port` or the TKLM GUI. On this port, the key management server listens for requests from tape drives. The values is required in both the Atape and EKM configuration files; TKLM defaults to 3801.

14.6.4 System-Managed Encryption Atape device entries

SME can be set on a specific tape drive using the standard SMIT panels to Change/Show Characteristics of a tape device or the command line **chdev** command.

Two attributes have been added for encryption:

- sys_encryption “yes/no”

  Use System Encryption FCP Proxy Manager. This attribute enables device driver SME for a tape drive by setting the value to yes.

- wrt_encryption “off/on/custom”

  Use System Encryption for Write Commands at BOP. This attribute controls whether the device driver sets the tape drive to encryption-enabled for write commands:

– When set to no, the tape drive uses encryption for read operations, and write operations do not use encryption.
– When set to yes, the tape drive uses encryption for both read and write operations.
– When set to custom, the device driver does not modify the current tape drive setting. The custom setting is intended for applications using SME to control write encryption without device driver intervention.

Our TS3500

Looking at our TS3500, you notice that it is partitioned into several logical libraries as reflected in Example 14-6.

Example 14-6 Our partitioned TS3500

```
lsdev -Cc tape
rmt0  Available 06-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt1  Available 06-08-02 IBM 3592 Tape Drive (FCP)
rmt2  Available 06-08-02 IBM 3592 Tape Drive (FCP)
rmt3  Available 06-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt4  Available 06-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt5  Available 06-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt6  Available 06-08-02 IBM 3592 Tape Drive (FCP)
rmt7  Available 06-08-02 IBM 3592 Tape Drive (FCP)
rmt8  Available 06-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt9  Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt10 Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt11 Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt12 Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt13 Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
rmt14 Available 08-08-02 IBM 3580 Ultrium Tape Drive (FCP)
smc0  Available 06-08-02 IBM 3584 Library Medium Changer (FCP)
smc1  Available 06-08-02 IBM 3584 Library Medium Changer (FCP)
smc2  Available 06-08-02 IBM 3584 Library Medium Changer (FCP)
smc3  Available 06-08-02 IBM 3573 Tape Medium Changer (FCP)
smc4  Available 06-08-02 IBM 3584 Library Medium Changer (FCP)
smc5  Available 06-08-02 IBM 3573 Tape Medium Changer (FCP)
smc6  Available 08-08-02 IBM 3584 Library Medium Changer (FCP)
```

Our logical library consists of the devices shown in Example 14-7.

Example 14-7 Our logical library

```
smc4  Available 06-08-02 IBM 3584 Library Medium Changer (FCP)
rmt6  Available 06-08-02 IBM 3592 Tape Drive (FCP)
rmt7  Available 06-08-02 IBM 3592 Tape Drive (FCP)
```

14.6.5 Updating the Atape device driver configuration

Prior to making any changes to the Atape configuration, we display the encryption attributes using the AIX **lsattr** command as well as **tapeutil**. Refer to Example 14-8 on page 481.

Example 14-8 Using lsattr command and tapeutil to check encryption parameters on the drives

```
> lsattr -El rmt6 | grep encr
drv_encryption  yes            Drive Encryption Support                    False
sys_encryption  no             Use System Encryption FCP Proxy Manager     True
wrt_encryption  off            Encryption for Write Commands at BOP        True

> lsattr -El rmt7 | grep encr
drv_encryption  yes            Drive Encryption Support                    False
sys_encryption  no             Use System Encryption FCP Proxy Manager     True
wrt_encryption  off            Encryption for Write Commands at BOP        True

tapeutil -f/dev/rmt6 encryption
Getting drive encryption settings...
Encryption settings:

  Drive Encryption Capable.... Yes
  Encryption Method........... Library
  Encryption State............ NA

tapeutil -f/dev/rmt7 encryption
Getting drive encryption settings...
Encryption settings:

  Drive Encryption Capable.... Yes
  Encryption Method........... Library
  Encryption State............ NA
```

Our tape drives are in an IBM TS3500 Tape Library and currently have LME turned on. If you are changing your drives to SME from LME and attempt to use **smitty tape** to change the `Use system encryption FCP Proxy manager` setting while in this state, you receive the message in Example 14-9. If the message:

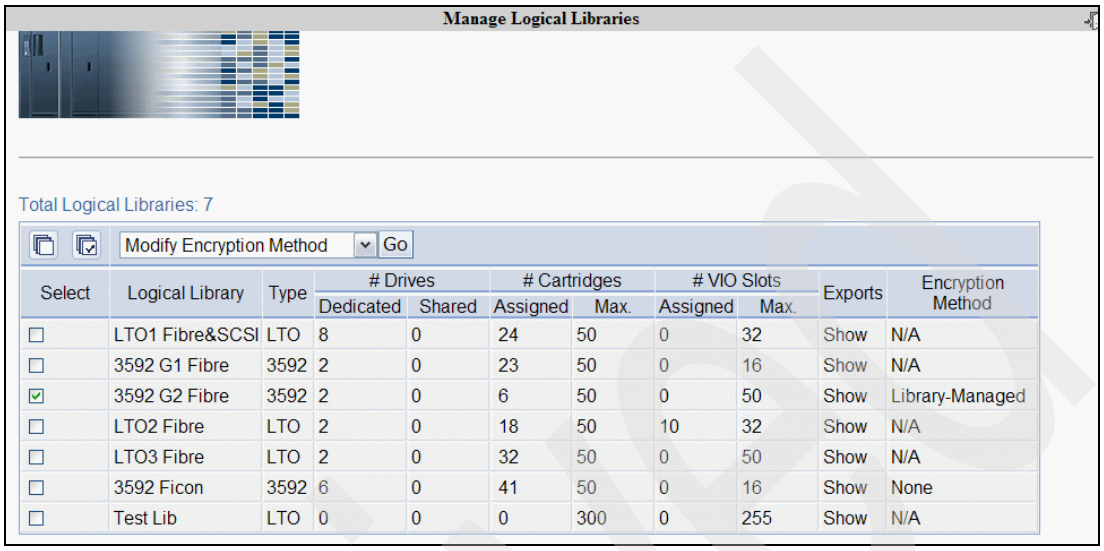
- Applies to your environment, read and follow 14.6.6, “Enabling System-Managed Encryption using the TS3500 Web GUI” on page 482.
- Does not apply to your environment, continue with 14.6.7, “Using SMIT to enable System-Managed Encryption” on page 483.

Example 14-9 Error message 0514-018

```
COMMAND STATUS

Command: failed         stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

Method error (/etc/methods/chgAtape):
        0514-018 The values specified for the following attributes
                 are not valid:

 Drive encryption method 60 not set to system method
```

14.6.6 Enabling System-Managed Encryption using the TS3500 Web GUI

To enable SME using the TS3500 Web GUI, **Select** → **Manage Library** → **By Logical Library** and press Enter. The panel shown in Figure 14-23 opens.



Select	Logical Library	Type	# Drives		# Cartridges		# VIO Slots		Exports	Encryption Method
			Dedicated	Shared	Assigned	Max.	Assigned	Max.		
☐	LTO1 Fibre&SCSI	LTO	8	0	24	50	0	32	Show	N/A
☐	3592 G1 Fibre	3592	2	0	23	50	0	16	Show	N/A
☑	3592 G2 Fibre	3592	2	0	6	50	0	50	Show	Library-Managed
☐	LTO2 Fibre	LTO	2	0	18	50	10	32	Show	N/A
☐	LTO3 Fibre	LTO	2	0	32	50	0	50	Show	N/A
☐	3592 Ficon	3592	6	0	41	50	0	16	Show	None
☐	Test Lib	LTO	0	0	0	300	0	255	Show	N/A

Figure 14-23 Modify encryption method

Select the library to modify and using the drop-down list box, select **Modify Encryption Method** and press Enter. The panel shown in Figure 14-24 opens.

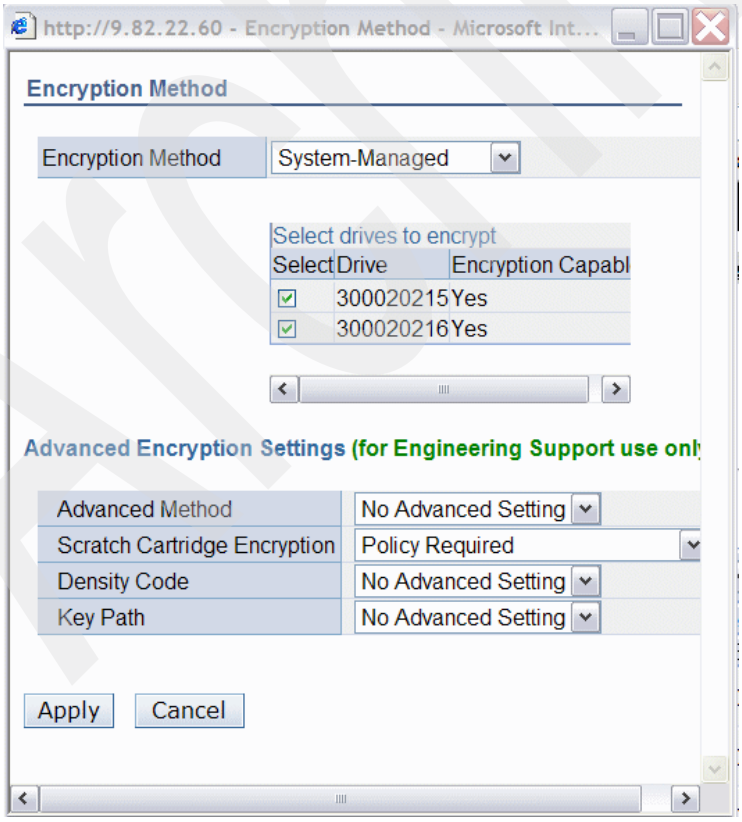


Figure 14-24 Using the TS3500 Web GUI to turn on SME for selected drives

Note: Referencing Figure 14-24 on page 482, note that all of the drives in an SME logical library do not have to use SME. Only this Open Systems encryption environment allows this. When implementing LME, all drives in a logical library must use LME. When implementing AME, all drives in a logical library must use AME. This is not the case with SME.

Now that you have changed the drives to use SME, if you use **tapeutil** to display the encryption settings for each drive, you see the output listed in Example 14-10.

Example 14-10 tapeutil drive display

```
>tapeutil -f/dev/rmt6 encryption
Getting drive encryption settings...
Encryption settings:

  Drive Encryption Capable.... Yes
  Encryption Method........... System
  Encryption State............ NA

>tapeutil -f/dev/rmt7 encryption
Getting drive encryption settings...
Encryption settings:

  Drive Encryption Capable.... Yes
  Encryption Method........... System
  Encryption State............ Off
```

14.6.7 Using SMIT to enable System-Managed Encryption

In Example 14-10, you note that the drives are encryption-capable, the encryption method selected is System, but the encryption state is off. The two parameters that have to be changed for SME are not yet set. We use an AIX tool called System Management Interface Tool (SMIT) to change these parameters. SMIT provides an alternative to the typical method of using complex command syntax, valid parameter values, and custom shell path names for managing and maintaining your operating system configuration.

For more information about SMIT, you can refer to a white paper located at:

http://www.ibm.com/servers/aix/products/aixos/whitepapers/smit.pdf

Enter **smitty tape** and press Enter. The result is shown in Figure 14-25 on page 484.

```
Tape Drive

Move cursor to desired item and press Enter.

  List All Defined Tape Drives
  List All Supported Tape Drives
  Add a Tape Drive
  Change / Show Characteristics of a Tape Drive
  Remove a Tape Drive
  Configure a Defined Tape Drive
  Generate Error Report
  Trace a Tape Drive
```

Figure 14-25 SMITTY Tape

Highlight **Change / Show Characteristics of a Tape Drive** as shown in Figure 14-25 and then press Enter.

The results in the panel are shown in Figure 14-26.

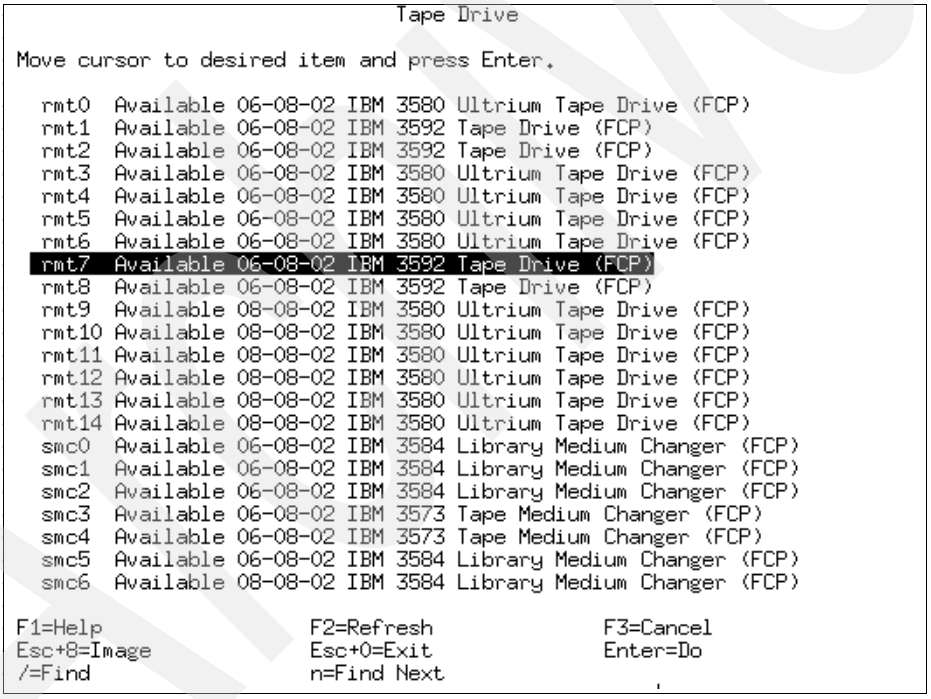


Figure 14-26 SMIT select drive

Select the drive and press Enter. The results are shown Figure 14-27 on page 485.

```
                                  Change / Show Characteristics of a Tape Drive

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Tape Drive                                           rmt7
  Tape Drive type                                      3592
  Tape Drive interface                                 fcp
  Description                                          IBM 3592 Tape Drive (FCP)
  Status                                               Available
  Location                                             06-08-02
  Parent adapter                                       fscsi0
  Connection address                                   13
  SCSI ID                                              0xeb000a
  Logical Unit ID                                      0x0
  World Wide Port Name                                 0x5005076300420215
  World Wide Node Name                                 0x5005076300020215
  New Logical Name                                    []
  Enable Alternate Pathing Support                     no
  Block Size (0=Variable Length)                      [0]
  Use DEVICE BUFFERS during writes                     yes
  Use Hardware Compression on Tape                     yes
  Activate volume information logging                  no
  Maximum size of log file (in # of entries)          [500]
  Backward Space/Forward Space Record Mode             SCSI
  Use Immediate Bit in Rewind Commands                 no
  Trailer Label Processing                             no
  Use System Encryption FCP Proxy Manager              no
  System Encryption for Write Commands at BOP          off
```

Figure 14-27 Drive parameters prior to SME turned on

We have to change the last two fields:

- Use System Encryption FCP Proxy Manager

  This attribute enables device driver SME for a tape drive by setting the value to `yes`.

- System Encryption for Write Commands at BOP.

  This attribute controls if the device driver sets the tape drive to encryption-enabled for write commands. When set to no, the tape drive uses encryption for read operations, and write operations do not use encryption. When set to yes, the tape drive uses encryption for both read/write operations. When set to custom, the device driver does not modify the current tape drive setting.

The custom setting is intended for applications using SME to control write encryption without device driver intervention.

Using SMIT, highlight each field one at a time and change the value; then, press Enter to start the process to change all fields. Highlighting the field and pressing F4 results in the options that you can select as shown in Figure 14-28.

```
                    Use System Encryption FCP Proxy Manager

Move cursor to desired item and press Enter.

  no
  yes

F1=Help                 F2=Refresh              F3=Cancel
Esc+8=Image             Esc+0=Exit              Enter=Do
/=Find                  n=Find Next
```

Figure 14-28 Results of F4 list

The SMIT tool is careful to ensure that even after you select yes, you still have a chance to change your mind. So even after you select yes, press Enter, and see the display shown in Figure 14-29 on page 486. You still have to confirm that you want to make the change, so you must press Enter one more time.

```
                                  Change / Show Characteristics of a Tape Drive

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Tape Drive                                           rmt7
  Tape Drive type                                      3592
  Tape Drive interface                                 fcp
  Description                                          IBM 3592 Tape Drive (FCP)
  Status                                               Available
  Location                                             06-08-02
  Parent adapter                                       fscsi0
  Connection address                                   13
  SCSI ID                                              0xeb000a
  Logical Unit ID                                      0x0
  World Wide Port Name                                 0x5005076300420215
  World Wide Node Name                                 0x5005076300020215
  New Logical Name                                    []
  Enable Alternate Pathing Support                     no
  Block Size (0=Variable Length)                      [0]
  Use DEVICE BUFFERS during writes                     yes
  Use Hardware Compression on Tape                     yes
  Activate volume information logging                  no
  Maximum size of log file (in # of entries)          [500]
  Backward Space/Forward Space Record Mode             SCSI
  Use Immediate Bit in Rewind Commands                 no
  Trailer Label Processing                             no
  Use System Encryption FCP Proxy Manager              yes
  System Encryption for Write Commands at BOP          off
```

Figure 14-29 Selecting the Encryption mode

The change has not taken place unless you see the panel shown in Figure 14-30, which confirms that the change has been made.

```
                                                        COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

rmt7 changed
```

Figure 14-30 Pressing Enter until rmt reflects the change

Important: After highlighting `yes`, ensure that you press Enter until you see Figure 14-30, which reflects that `rmt7 changed`. If not, the change does not take effect.

The `Use System Encryption FCP Proxy Manager` attribute, which enables device driver SME for a tape drive, has been set to `yes` as shown in Figure 14-31 on page 487.

```
                                  Change / Show Characteristics of a Tape Drive

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Tape Drive                                           rmt7
  Tape Drive type                                      3592
  Tape Drive interface                                 fcp
  Description                                          IBM 3592 Tape Drive (FCP)
  Status                                               Available
  Location                                             06-08-02
  Parent adapter                                       fscsi0
  Connection address                                   13
  SCSI ID                                              0xeb000a
  Logical Unit ID                                      0x0
  World Wide Port Name                                 0x5005076300420215
  World Wide Node Name                                 0x5005076300020215
  New Logical Name                                    []
  Enable Alternate Pathing Support                     no
  Block Size (0=Variable Length)                      [0]
  Use DEVICE BUFFERS during writes                     yes
  Use Hardware Compression on Tape                     yes
  Activate volume information logging                  no
  Maximum size of log file (in # of entries)          [500]
  Backward Space/Forward Space Record Mode             SCSI
  Use Immediate Bit in Rewind Commands                 no
  Trailer Label Processing                             no
  Use System Encryption FCP Proxy Manager              yes
  System Encryption for Write Commands at BOP          off
```

Figure 14-31 SME device driver parameter

The next and last parameter to change is the `System Encryption for Write Commands at BOP` parameter shown in Figure 14-32. This attribute controls if the device driver sets the tape drive to encryption-enabled for write commands. When set to no, the tape drive uses encryption for read operations, and write operations do not use encryption. When set to yes, the tape drive uses encryption for both read and write operations. When set to custom, the device driver does not modify the current tape drive setting.

```
                                  Change / Show Characteristics of a Tape Drive

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Tape Drive                                           rmt7
  Tape Drive type                                      3592
  Tape Drive interface                                 fcp
  Description                                          IBM 3592 Tape Drive (FCP)
  Status                                               Available
  Location                                             06-08-02
  Parent adapter                                       fscsi0
  Connection address                                   13
  SCSI ID                                              0xeb000a
  Logical Unit ID                                      0x0
  World Wide Port Name                                 0x5005076300420215
  World Wide Node Name                                 0x5005076300020215
  New Logical Name                                    []
  Enable Alternate Pathing Support                     no
  Block Size (0=Variable Length)                      [0]
  Use DEVICE BUFFERS during writes                     yes
  Use Hardware Compression on Tape                     yes
  Activate volume information logging                  no
  Maximum size of log file (in # of entries)          [500]
  Backward Space/Forward Space Record Mode             SCSI
  Use Immediate Bit in Rewind Commands                 no
  Trailer Label Processing                             no
  Use System Encryption FCP Proxy Manager              yes
  System Encryption for Write Commands at BOP          off
```

Figure 14-32 SME for Write Commands

As indicated previously, selecting F4 lists the options as shown in Figure 14-33 on page 488. We set this value to yes, so that the tape drive uses encryption for both read and write operations.

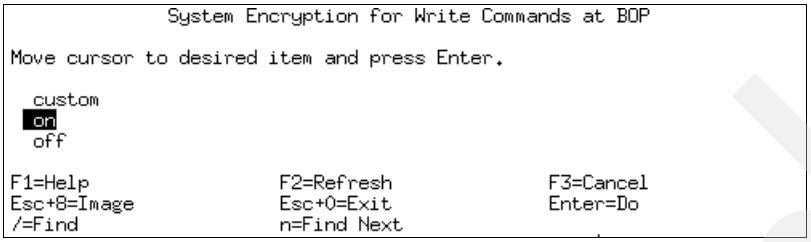


Figure 14-33 Select F4 for list

Again, you see the confirmation panel as shown in Figure 14-34 and the instruction to press Enter.

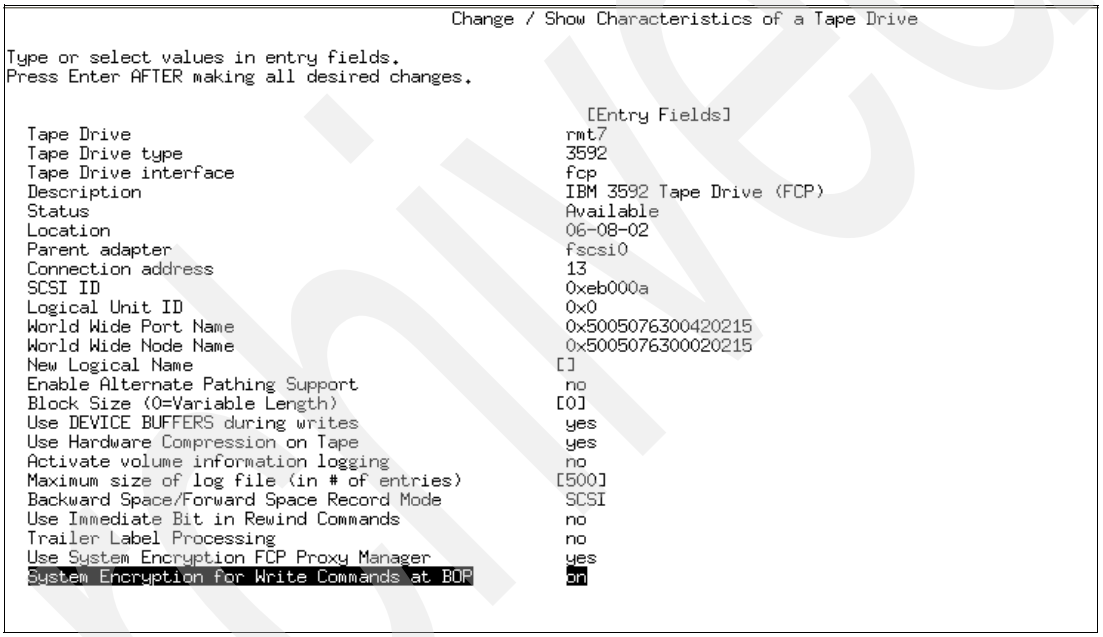


Figure 14-34 Confirmation still required

Just a reminder that you have to press Enter one more time to confirm the change and that you have to see the panel shown in Figure 14-35.

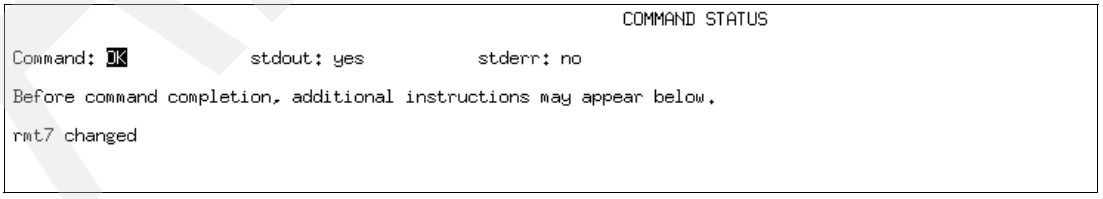


Figure 14-35 Change has been accepted

The display shown in Figure 14-36 on page 489 reflects that both parameters have been changed for rmt7. This drive is almost ready to write encrypted data.

```
                                        Change / Show Characteristics of a Tape Drive

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
  Tape Drive                                           rmt7
  Tape Drive type                                      3592
  Tape Drive interface                                 fcp
  Description                                          IBM 3592 Tape Drive (FCP)
  Status                                               Available
  Location                                             06-08-02
  Parent adapter                                       fscsi0
  Connection address                                   13
  SCSI ID                                              0xeb000a
  Logical Unit ID                                      0x0
  World Wide Port Name                                 0x5005076300420215
  World Wide Node Name                                 0x5005076300020215
  New Logical Name                                    []
  Enable Alternate Pathing Support                     no
  Block Size (0=Variable Length)                      [0]
  Use DEVICE BUFFERS during writes                     yes
  Use Hardware Compression on Tape                     yes
  Activate volume information logging                  no
  Maximum size of log file (in # of entries)          [500]
  Backward Space/Forward Space Record Mode             SCSI
  Use Immediate Bit in Rewind Commands                 no
  Trailer Label Processing                             no
  Use System Encryption FCP Proxy Manager              yes
  System Encryption for Write Commands at BOP          on
```

Figure 14-36 All changes for rmt7 have been made

You can use the AIX command **lsattr -El rmt*n*** | **grep encryp** and then **tapeutil** to verify that the drive is SME ready. Refer to Example 14-11.

Example 14-11 lsattr and tapeutil display

```
lsattr -El rmt8 | grep encryp
drv_encryption  yes                 Drive Encryption Support False
sys_encryption  yes                 Use System Encryption FCP Proxy Manager True
wrt_encryption  on                  System Encryption for Write Commands at BOP
lsattr -El rmt7 | grep encryp
drv_encryption  yes                 Drive Encryption Support False
sys_encryption  yes                 Use System Encryption FCP Proxy Manager True
wrt_encryption  on                  System Encryption for Write Commands at BOP

> tapeutil -f/dev/rmt7 encryption
Getting drive encryption settings...
Encryption settings:
Drive Encryption Capable.... Yes
  Encryption Method........... System
  Encryption State............ On
> tapeutil -f/dev/rmt8 encryption
Getting drive encryption settings...
Encryption settings:

  Drive Encryption Capable.... Yes
  Encryption Method........... System
  Encryption State............ On
```

Figure 14-37 Drive encryption characteristics

14.6.8 Using tapeutil functions to verify EKM paths

A tapeutil command is used to validate the `ibmekm.conf` server entries and test connectivity operations from the tape drive to the server. This test can be run using the tapeutil menu option 40 Data Encryption Test or the command line **`tapeutil –f/dev/name ekmtest`**.

Note: If a cartridge is mounted on the tape drive, unload this cartridge before running the **`ekmtest`** command.

The first test checks the server configuration defined in the `ibmekm.conf` file and then communication to the configured servers. This test reports the number of servers available. The second test runs a basic diagnostic test that checks the tape drive to server communication and reports success or failure. The third test runs an enhanced diagnostic test that checks a key operation between the tape drive and server, and then it reports success or failure.

The output of the **`tapeutil`** command is shown in Example 14-12.

Example 14-12 tapeutil ekm validation

```
Enter Selection for /dev/rmt7: 40
Testing server configuration and connections...
Testing complete, servers available 1
Running basic drive to server encryption test...
Testing complete, completion code 0
Running full drive to server encryption test...
Testing complete, completion code 0
```

After all Atape changes have been made and the EKM **`tapeutil`** validation has been run to a successful completion, you are ready to start encrypting data. Your next concern is how to manage this process, which we describe in the next section.

14.6.9 Managing System-Managed Encryption and business partner exchange

Managing SME is different from managing LME. Whether to encrypt in LME is managed at the VOLSER-level by using the Scratch Encryption Policies (SEPs). SME is managed at the drive-level, because not every drive in a logical library must have SME implemented. Implementing your decision to encrypt a tape using SME is solved simply by directing allocations to one drive instead of another.

Let us quickly review label usage. Labels inform EKM which public key to use when encrypting the data key (DK). This encrypted DK is stored within an EEDK along with the label and written to the cartridge. There are two EEDKs for each cartridge. The EKM at your business partner's site unwraps the EEDKs, pulls out the certificate label, and uses it to do a lookup within your business partner's keystore to obtain the correct private key to decrypt the DK.

Referring to Figure 14-38 on page 491, the EKM responds to a write request in the following order (these numbers correspond to the numbers in the figure):

1. The tape drive requests the key to encrypt the tape.
2. The EKM verifies that the tape device is in the Drive Table.
3. The EKM fetches the keys and certificates for the tape device from the keystore.
4. The EKM generates a random DK and wraps it with public keys to create an EEDK.

5. The EKM sends the EEDKs and (separately wrapped) DK to the tape drive.
6. The tape drive unwraps the DK and writes the EEDKs on the tape leader.
7. The tape drive encrypts data using the DK and writes encrypted data to tape.

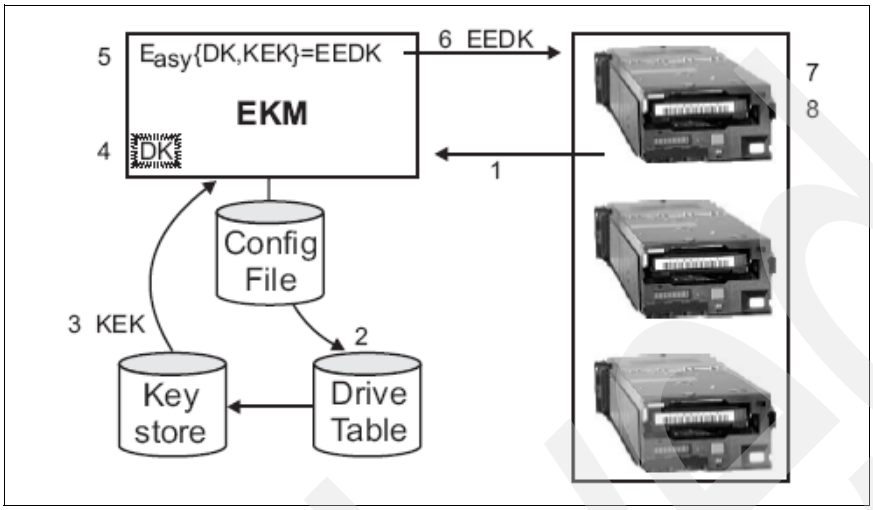


Figure 14-38 EKM response to a write request

Referring to Figure 14-39, the following events occur when you read an encrypted cartridge at your business partner's site:

1. The tape drive receives a read request and sends the EEDKs to the EKM.
2. The EKM verifies the tape device in the Drive Table.
3. The EKM fetches the keys required to process the EEDKs from the keystore.
4. The EKM unwraps the EEDK using the private key of the KEK pair to recover the DK.
5. The EKM wraps the DK with a key that the drive can decrypt and sends the wrapped DK to the tape drive.
6. The tape drive unwraps the DK and uses it to decrypt the data.

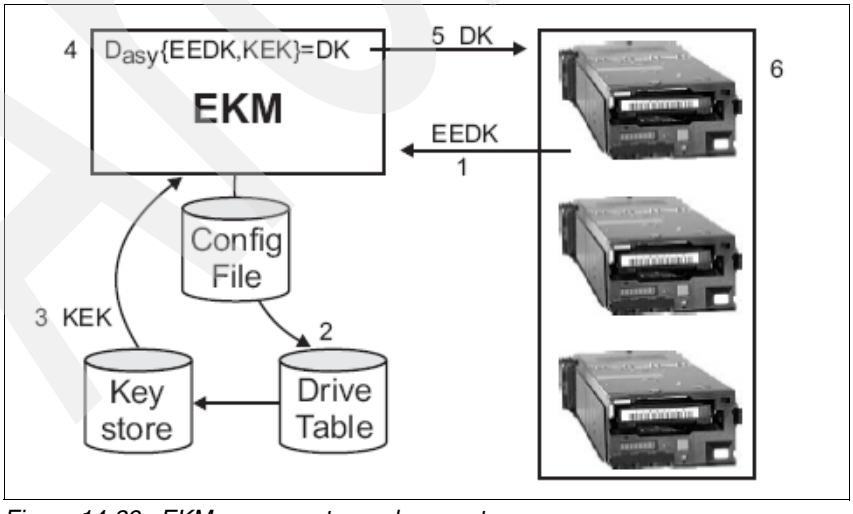


Figure 14-39 EKM response to read request

The success of the read operation at your business partner's site assumes two things:

- You have selected the correct public key (using label specification) to encrypt the data key when the cartridge was created.
- The label you use to encrypt also exists in your business partner's keystore.

LME within the BEP has a Hash option so that certificate labels do not have to match. SME does *not* have this option, so you must ensure that label usage between you and your business partners is carefully managed. it is necessary that you and your business partner agree on a naming convention for the business partner's certificate, or that you exchange this information with your business partner so that the correct certificate can be found within the business partner's keystore.

Sending a cartridge to a business partner entails selecting the correct public key to encrypt the data key, and you accomplish this by specifying the right certificate label to use. How do you do this within SME? Business partner exchange for LME is managed using labels specified within Barcode Encryption Policies. SME business partner exchange is managed using label information kept in the drive table or EKM configuration file.

These values can be specified in the drive table or if you have `drive.acceptUnknownDrives` set to `yes` in your EKM configuration, they are taken from the EKM configuration file global `default.drive.alias1` and `default.drive.alias2` parameters.

EKM provides an option to set an EKM-wide default for the primary alias and secondary alias. Refer to Table 14-6.

Table 14-6 EKM label alias configuration parameters

Parameters	Definition	Required
drive.default.alias1	Provides a default alias, to a drive, for use if an alias is not specified in the drive table.	Yes
drive.default.alias2	Provides a default alias, to a drive, for use if an alias is not specified in the drive table.	Yes

These variables are used to wrap the data encrypting key when writing to a tape drive that has not been configured with specific keys and certificates to use. The values of these variables are used when a tape device in the drive table does not have a specific definition as to which alias to use when writing a tape. When EKM encounters a device that does not have a specific entry in the drive table specifying an alias (or key label) and an alternate alias, EKM then uses the `drive.default.alias1` and `drive.default.alias2` variables if they are set.

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. In other words, you can have EKM fully configure the device with associated certificates when the device contacts EKM. 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.

Changing which encryption label to use is as easy as issuing the **`moddrive`** command prior to allocation. You can use the **`moddrive`** command to modify drive information in the drive table. The equivalent command is **`modifydrive`**. See Example 14-13 on page 493 for the syntax of the **`moddrive`** command.

Example 14-13 moddrive command

```
moddrive -drivename drivename [ -rec1 alias -rec2 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.

Example: moddrive -drivename 000123456789 -rec1 newalias1
```

For more information about these configuration variables, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide*, GA76-0418.

14.7 Application-Managed Encryption

Application-Managed Encryption (AME) is currently supported with ITSM. In this section, we describe how to implement encryption in the ITSM environment. We contrast IBM Tivoli Storage Manager encryption with LME and SME and give best practices regarding when to use each method. ITSM uses the AME method, as shown in Figure 14-40.

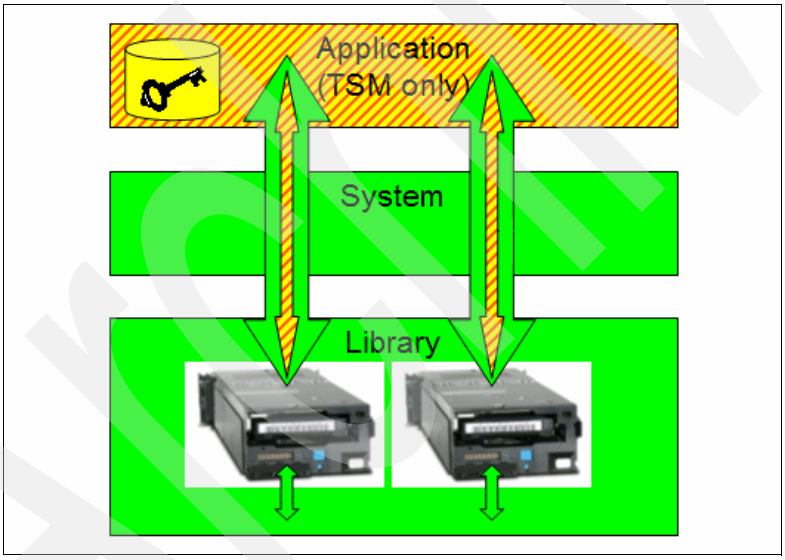


Figure 14-40 Application-Managed Encryption architecture

By design, when implementing AME, the application is very much aware of encryption. ITSM creates and manages the policies and the keys and passes this information between the application layer and the 3592-E05 drive.

14.7.1 IBM Tivoli Storage Manager overview

IBM Tivoli Storage Manager (ITSM) is an enterprise-wide storage management application. It provides automated storage management services to workstations, personal computers, and file servers from a variety of vendors, with a variety of operating systems.

How ITSM stores client data

IBM Tivoli Storage Manager stores data using policies. Policies are rules that determine how the client data is stored and managed. The rules include where the data is initially stored, how many backup versions are kept, how long archive copies are kept, and so on. You can have multiple policies and assign different policies as required to specific clients, or even to specific files. A *policy* assigns a location in server storage where data is initially stored. Server storage is divided into *storage pools* that are groups of storage volumes. Server storage can include hard disk, optical, and tape volumes.

Policies are rules that you set at the ITSM server to help you manage client data. Policies control how and when client data is stored, for example:

- How and when files are backed up and archived to server storage
- How space-managed files are migrated to server storage
- The number of copies of a file and the length of time that copies are kept in server storage

Referring to Figure 14-41, when a IBM Tivoli Storage Manager client is registered, it is associated with a *policy domain*. Within the policy domain are the policy set, management class, and copy groups. When a client backs up, archives, or migrates a file, it is bound to a management class. A *management class* and the backup and archive copy groups within it specify where files are stored and how they are managed when they are backed up, archived, or migrated from the client. *Storage pools* are the destinations for backed-up, archived, or space-managed files. *Copy groups* specify storage pools for backed-up or archived files. Management classes specify storage pools for space-managed files. Storage pools are mapped to *device classes*, which represent devices. The storage pool contains volumes of the type indicated by the associated device class. For example, a storage pool that is mapped to a device class with a device type of 8 MM contains only 8 mm tapes. Files that are initially stored on disk storage pools can migrate to tape or optical disk storage pools if the pools are set up in a storage hierarchy.

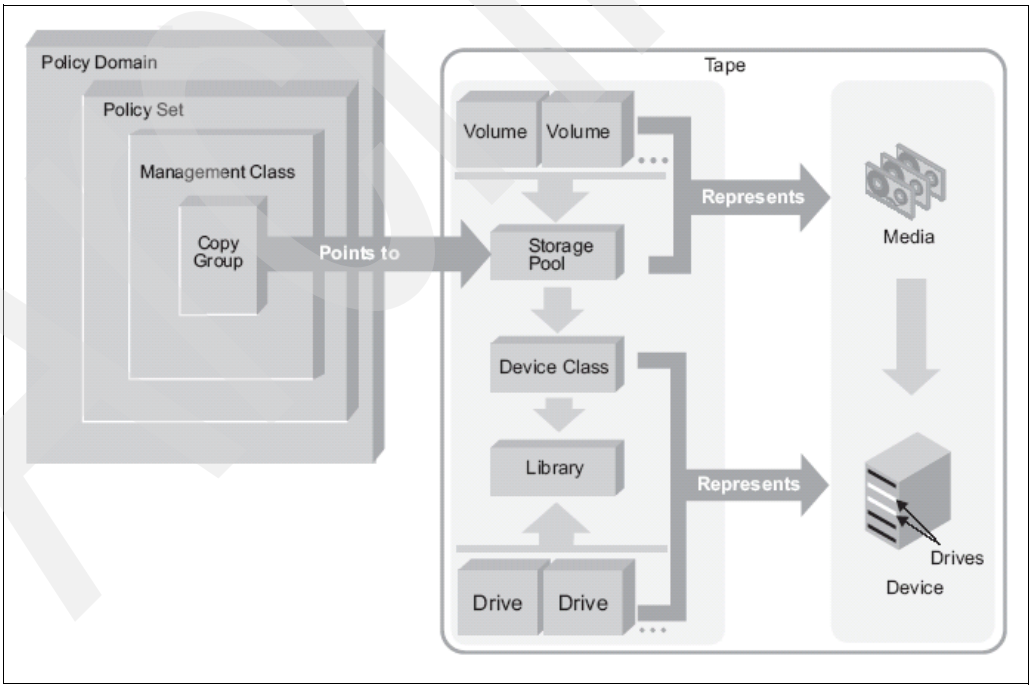


Figure 14-41 How clients, server storage, and policies work together

14.7.2 ITSM support for 3592 drive encryption

IBM tape device encryption is now supported for 3592-E05, 3592-E06 and 3592-EU6 drives. When enabled, ITSM handles encrypting and decrypting data on tapes, according to specifications set when defining the device class. When using the TS1130 with ITSM you must be running TSM 5.4.3 or 5.5.1 or later.

Encryption keys are managed by the IBM Tivoli Storage Manager and not by the Encryption Key Manager. ITSM generates and stores the keys in the IBM Tivoli Storage Manager server database. The data keys required to decrypt the cartridge data are *not* kept on the data cartridge as they are with LME and SME. Data is encrypted during WRITE operations, when the encryption key is passed from the server to the drive. Data is decrypted on READ operations. The AME method is only supported for storage pool volumes.

Data encrypted using ITSM is *incompatible* with data encrypted using LME or SME.

14.7.3 Implementing Application-Managed Encryption

To utilize drive encryption, your IBM Tivoli Storage Manager environment must be set up so that all drives in a library support the new encryption format. In addition, all drives within a logical library *must* use the same method of encryption. An environment is not supported that has some drives using AME, and some drives using LME or SME methods.

When using encryption-capable drives with a supported encryption method, a new format is used to write encrypted data to tapes. If volumes are written to use the new format and then returned to scratch, they contain labels that are only readable by encryption-enabled drives. To use these scratch volumes in a drive that is not enabled for encryption, either because the hardware is incapable of encryption or because the encryption method is set to NONE, you must relabel them.

Encryption is enabled using the DRIVEENCRYPTION parameter within the device class and can affect several storage pools, as shown in the example in Figure 14-42.

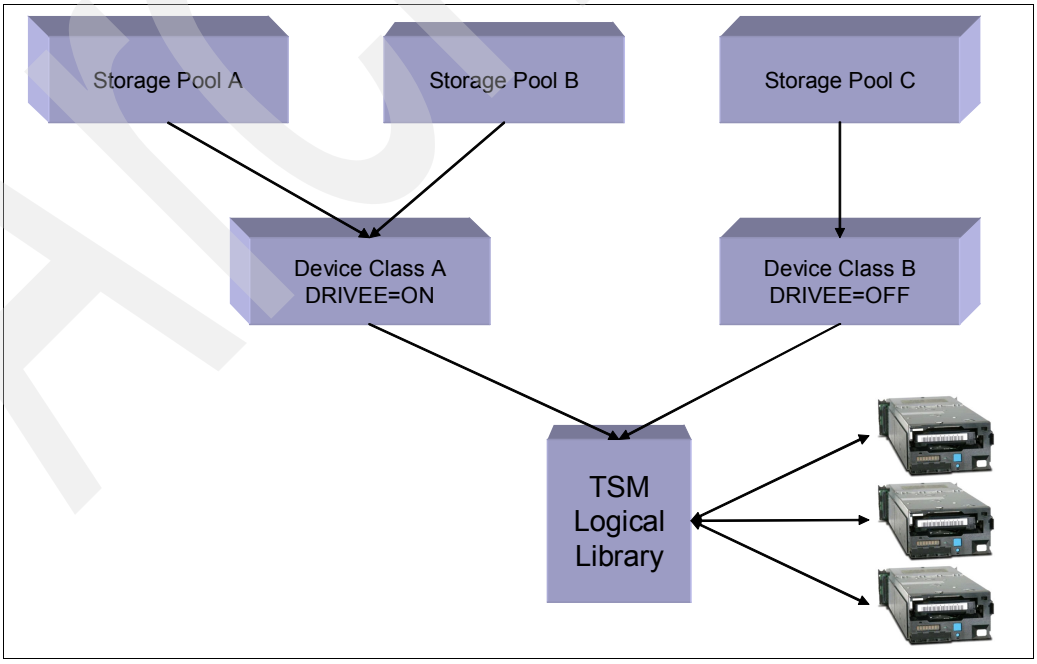


Figure 14-42 Device class impact

Setting the device class DRIVEEncryption parameter

The device class DRIVEEncryption parameter (abbreviated as **drivee**) is supported on the 3592-E05 also known as the TS1120 (3592-2 and 3592-2C) and 3592-E06 also known as the TS1130 (3592-3 and 3592-3C) formats. It specifies whether drive encryption is enabled or can be enabled. The three types of drive encryption available with the 3592-E05 and 3592-E06 drives are AME, SME, and LME. These methods are defined through the hardware. ITSM supports all three encryption methods with the DRIVEEncryption parameter. This parameter is used to ensure IBM Tivoli Storage Manager compatibility with hardware encryption settings for empty volumes. Refer to Figure 14-43.

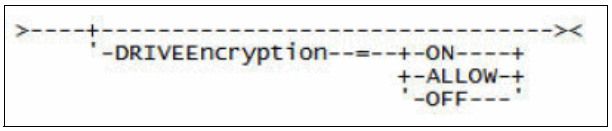


Figure 14-43 DRIVEEncryption parameter

Note the following information:

- To utilize the AME method, in which ITSM generates and manages encryption keys, the parameter must be set to `ON`. This permits the encryption of data for empty volumes. When the parameter is set to `ON`, backup operations fail if the hardware is configured for another encryption method.
- To utilize the LME or SME method, the parameter must be set to `ALLOW`. This specifies that IBM Tivoli Storage Manager is not the key manager for drive encryption, but allows the hardware to encrypt the volume's data through one of the other methods. Data is only encrypted by setting the `ALLOW` parameter and configuring the hardware to use one of these methods. Volumes are not automatically encrypted by specifying this parameter.
- To disable any method of encryption on new volumes, the parameter must be set to `OFF`. If the hardware is configured to encrypt data through the LME or SME method, and DRIVEEncryption is set to `OFF`, backup operations fail.

The DRIVEEncryption parameter is optional. The default value allows the LME or SME methods.

Defining an encrypted storage pool

Assume you want to encrypt storage pool volumes and use ITSM to manage encryption. This method is defined through the device class.

Complete the following steps:

1. Define your library: define library 3584 libtype=SCSI
2. Define a device class 3592_ENCRYPT so that storage pool volumes are encrypted: define devclass 3592_encrypt library=3584 driveencrypt=on
3. Define a storage pool named 3592_ENCRYPT_POOL with a MAXSCRATCH value of 10: define stgpool 3592_encrypt_pool 3592_encrypt maxscr=10

Displaying encryption status

To display the status of encryption for a particular devclass, issue the **`Query devclass`** command. To display a detailed report of the 3592 device class as shown in Figure 14-44 on page 497, enter the following command:

```
query devclass 3592 format=detailed
```

Note: In the figure, `Drive Encryption` is set to `On`.

```
            Device Class Name: 3592
      Device Access Strategy: Sequential
          Storage Pool Count: 1
                 Device Type: 3592
                      Format: 3592
       Est/Max Capacity (MB):
                 Mount Limit: DRIVES
           Mount Wait (min): 60
       Mount Retention (min): 60
                Label Prefix: ADSM
                     Library: MANLIB
                   Directory:
                 Server Name:
                Retry Period:
              Retry Interval:
                      Shared:
                      HLAddr:
                        WORM: No
             Scaled Capacity: 90
            Drive Encryption: On
Last Update by (administrator): SERVER_CONSOLE
        Last Update Date/Time: 08/04/03 14:28:31
```

Figure 14-44 Detailed Devclass query report

To check the status of a volume that results in a detailed report as shown in Figure 14-45, use the following command:

```
query volume 000642 format=detailed
```

```
                  Volume Name: 000642
             Storage Pool Name: 3592POOL
             Device Class Name: 3592CLASS
       Estimated Capacity (MB): 2,048.0
       Scaled Capacity Applied:
                       Pct Util: 0.0
                  Volume Status: Filling
                         Access: Read/Write
        Pct. Reclaimable Space: 0.0
                  Scratch Volume?: Yes
                  In Error State?: No
      Number of Writable Sides: 1
       Number of Times Mounted: 1
            Write Pass Number: 1
     Approx. Date Last Written: 03/22/2004 15:23:46
        Approx. Date Last Read: 03/22/2004 15:23:46
          Date Became Pending:
        Number of Write Errors: 0
         Number of Read Errors: 0
               Volume Location:
  Drive Encryption Key Manager: Tivoli Storage Manager
Volume is MVS Lanfree Capable : No
Last Update by (administrator):
         Last Update Date/Time: 03/22/2004 15:23:46
          Begin Reclaim Period: 03/22/2005
            End Reclaim Period: 04/22/2005
```

Figure 14-45 Detailed query volume command report

In this example, the `Drive Encryption Key Manager` field indicates `Tivoli Storage Manager` (ITSM). This means that the volume is encrypted and ITSM performed the key management.

Changing your encryption method and hardware configuration

If you want to update the DRIVEEncryption parameter for a given set of volumes, the volumes have to be returned to scratch status. Updating the parameter only affects empty volumes.

For example, if you currently have AME enabled, and you want to update DRIVEEncryption to OFF, only empty volumes are impacted by the change. Filling volumes continue to be encrypted, although new volumes are not. If you do not want currently filling volumes to

continue being encrypted, the volume status must be changed to READONLY. This ensures that Tivoli Storage Manager (ITSM) does not append any more encrypted data to the volumes. You can use the MOVE DATA command to transfer the data to a new volume following the update of the DRIVEEncryption parameter. The data then is available in an unencrypted format.

When migrating from one hardware configuration to another hardware configuration, you have to move your data from the old volumes to new volumes with new encryption keys and key managers. You can do this by setting up two logical libraries and storage pools (each with a different encryption method) and migrating the data from the old volumes to the new volumes. This eliminates volumes that were encrypted using the original method.

One scenario could you could have is that volumes that were encrypted using the LME method and you want to migrate to the AME method. ITSM is unable to determine which encryption keys are required for data on these volumes, because the library's Encryption Key Manager stores these keys and ITSM does not have access to them.

Table 14-7 illustrates considerations when changing your hardware encryption method.

Table 14-7 Hardware and volume compatibility

<table>
<tr><th></th><th>Volumes with no encryption</th><th>Volumes with AME</th><th>Volumes with LME</th><th>Volumes with SME</th></tr>
<tr><td>Desired hardware method=None</td><td rowspan="3">No special consideration</td><td colspan="3">Incompatible scratch tape labels will be unreadable and will have to be relabeled</td></tr>
<tr><td>Desired hardware method=Application</td><td>No special consideration</td><td colspan="2">Incompatible</td></tr>
<tr><td>Desired hardware method=Library</td><td>Incompatible</td><td>No special consideration

Ensure the same key bank/server is still used</td><td>Ensure the same key bank/server is still used

No special consideration</td></tr>
</table>

14.7.4 ITSM Encryption considerations

This section explores considerations unique to AME using ITSM.

Safeguarding encryption keys

When implementing AME using ITSM, take extra care to secure backups of the database. The keys to encrypt and decrypt data using these methods are stored in the ITSM server database. To restore your data, you must have the correct database backup and corresponding encryption keys to access your information. Ensure that you back up the database frequently and safeguard the backups to prevent data loss or theft. Anyone who has access to both the database backup and encryption keys has access to your data.

Business partner exchange

LME and SME include on each encrypted cartridge two EEDKs, which contain the DK that was used to encrypt the data on the cartridge. AME using ITSM does not. The DK is kept within the ITSM database.

Exchanging AME encrypted cartridges with business partners is a challenge as compared to LME or SME, because LME and SME include a copy of the DK on the cartridge. If AME cartridges are to be exchanged, a copy of the ITSM database also has to be shared, because the DK used to encrypt the data is only available using the ITSM database.

If your business dictates that you regularly have to share data with business partners, outside vendors, and so forth, we recommend that you continue to create this data as you do today, which is probably outside of ITSM's control. If you want to encrypt this data, we recommend that you encrypt it using LME or SME. This encryption environment is referred to as a *hybrid*.

Backing up the ITSM database

Implementing encryption does not affect or cause any changes to the process that you currently use to back up your ITSM databases.

14.8 IBM 3494 with TS1120 or TS1130 Encryption

In this section, we review the 3494 Web GUI panels and values that are required to implement LME using BEPs on a 3494 using TS1120 drives. The same principles apply when using TS1130 drives. We review the Web GUI panels for LME ILEP, AME, and SME.

14.8.1 Review the 3494 encryption-capable drives

As shown in Figure 14-46, we have four drives in our 3494 library that are encryption-capable. The drives are devices 100, 101, 300, and 301. Select **Administer Library Manager** → **Manage 3592 Drives.** In our examples, we work with drives 100 and 101.

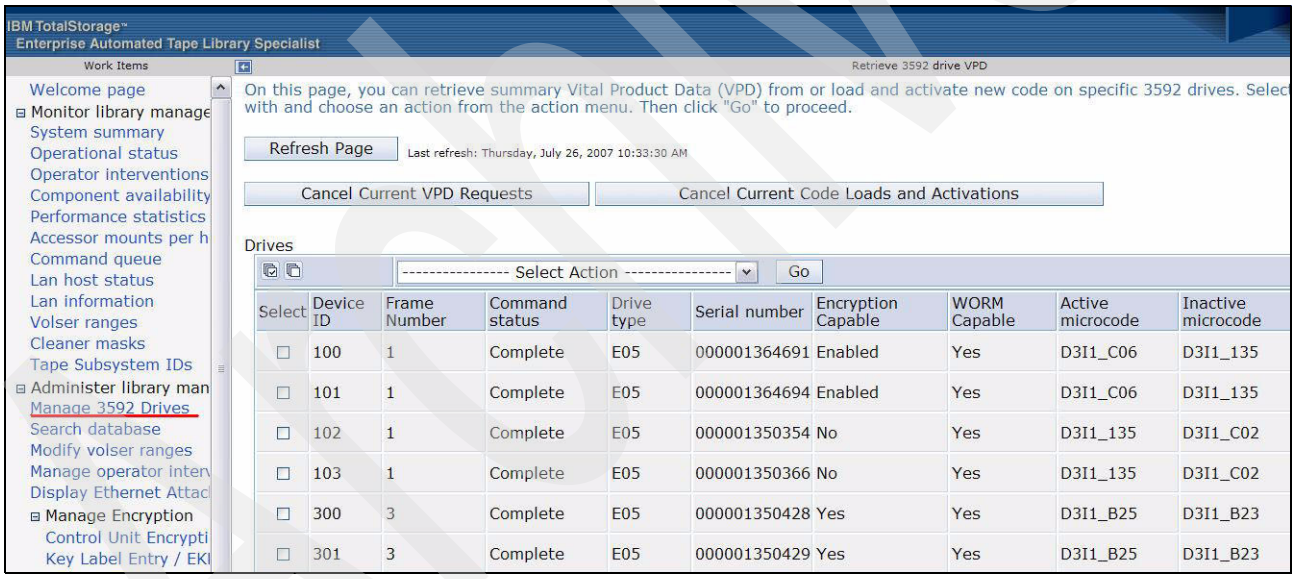


Figure 14-46 3494 encryption-capable drives

Select **Administer Library Manager** → **Manage Encryption** → **Drive Encryption Settings**. The results are shown in Figure 14-47 on page 500.

Note that drives 100 and 101 are already set for Library-Managed Encryption; however, they both use **Internal Label - selective encryption** scratch encryption policies.

Update the panel with current Vital Product Drive (VPD) information by selecting **Get VPD** (not Set VPD) and then clicking **Go**.

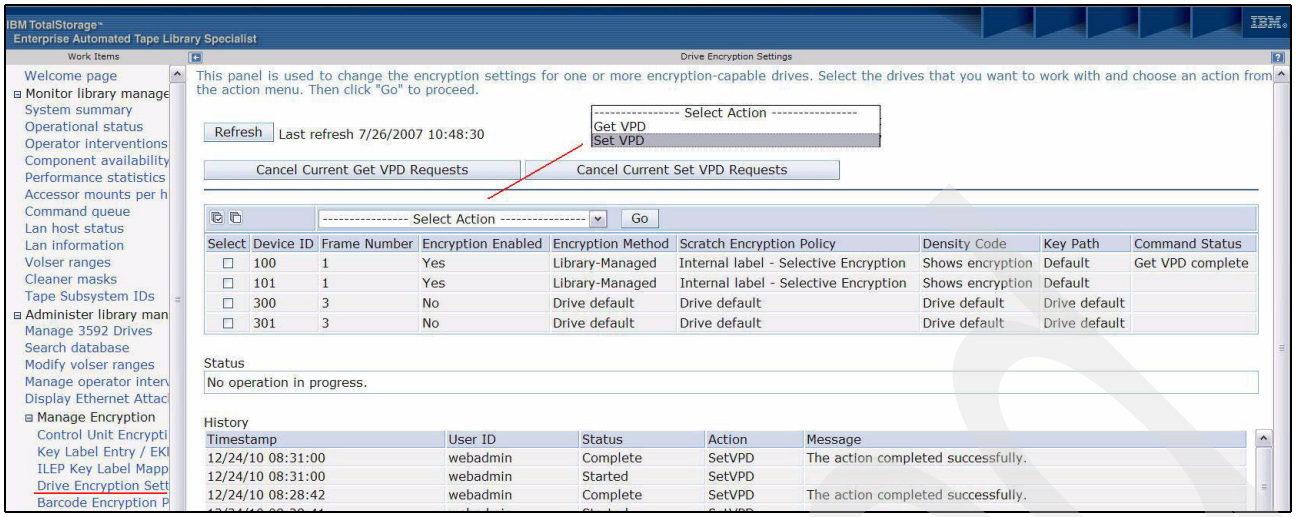


Figure 14-47 Drive Encryption Settings

The 3494, like the TS3500 Web GUI, has help panels to guide you. For example, if you have any questions about the Drive Encryption Settings panel, select the question mark character **(?)** in the top right corner of the panel. A User Assistance panel opens, as shown in Figure 14-48.

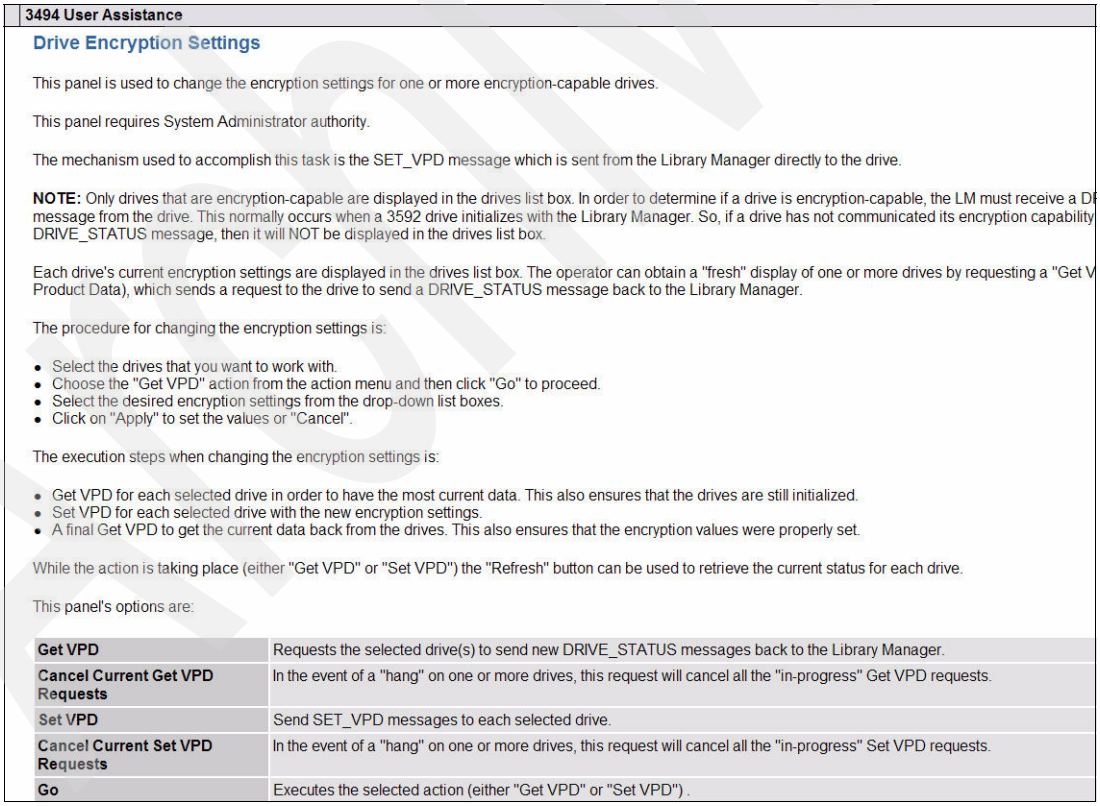


Figure 14-48 3494 Web GUI help panel

To set the drive encryption method, select the drives that you want to change, select **Set VPD,** and click **Go** as shown in Figure 14-47. For our example, we select drive **100** only, which results in the display as shown in Figure 14-49 on page 501.

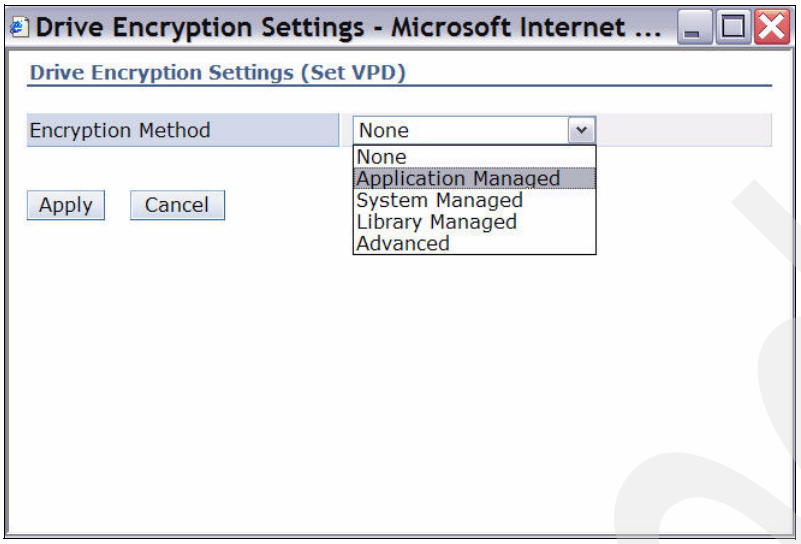


Figure 14-49 Drive Encryption Settings (Set VPD)

For our example, we select **Library Managed** and click **Apply**. (Note that on this panel, you may also that you select Application Managed or System Managed. If you select either one of those encryption methods, no other panel displays and you are finished with the 3494 Web GUI.

Selecting **Library Managed** results in the panel shown in Figure 14-50.This panel is where you select either **Barcode,** or if you are implementing Symantec NetBackup, you may select either the **Internal label - Selective Encryption** or **Internal Label - Encrypt All** policy method. Internal Label Encryption Policies are referred to as ILEP.

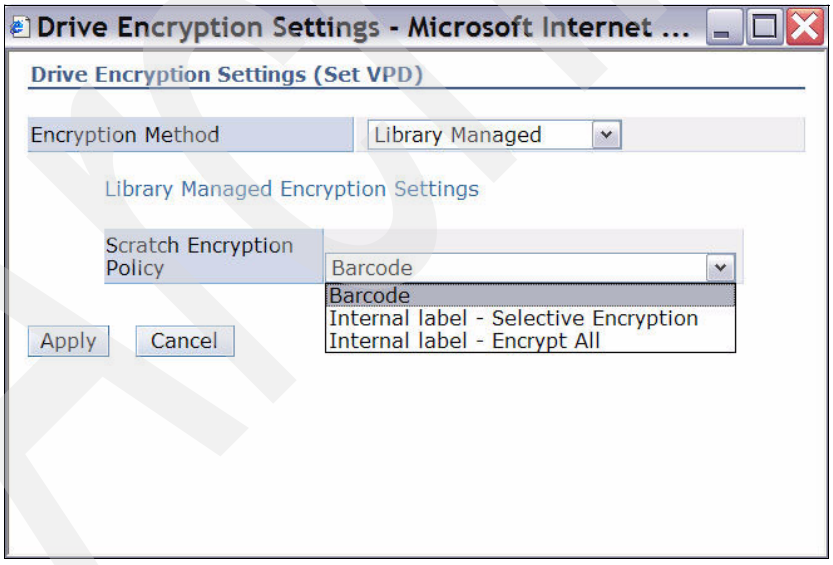


Figure 14-50 Library-Managed Scratch Encryption Policy

For our example, select **Barcode** and then click **Apply**. This results in the panel shown in Figure 14-51 on page 502. Note that for drive 100 in the **Scratch Encryption Policy** column, the value has changed from **Internal Label-selective Encryption** to **Barcode**. After you ensure that the correct BEPs are set, you are ready to encrypt data. We now look at specifying a Barcode Encryption Policy (BEP).

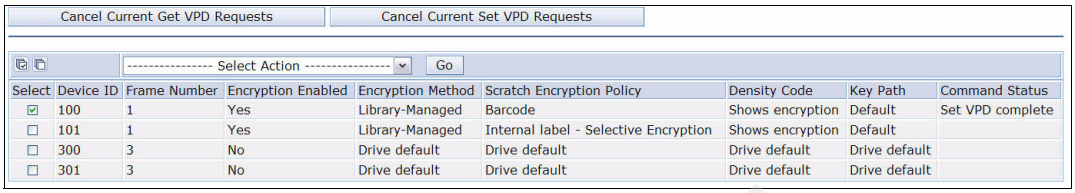


Figure 14-51 Drive 100 after LME and Barcode Encryption Policy changes

14.8.2 Specifying a Barcode Encryption Policy

Select **Administer Library Manager** → **Manage Encryption** → **Barcode Encryption Policy**. The panel shown in Figure 14-52 opens. The panel reflects three barcode encryption policies that are already created for three VOLSER ranges.

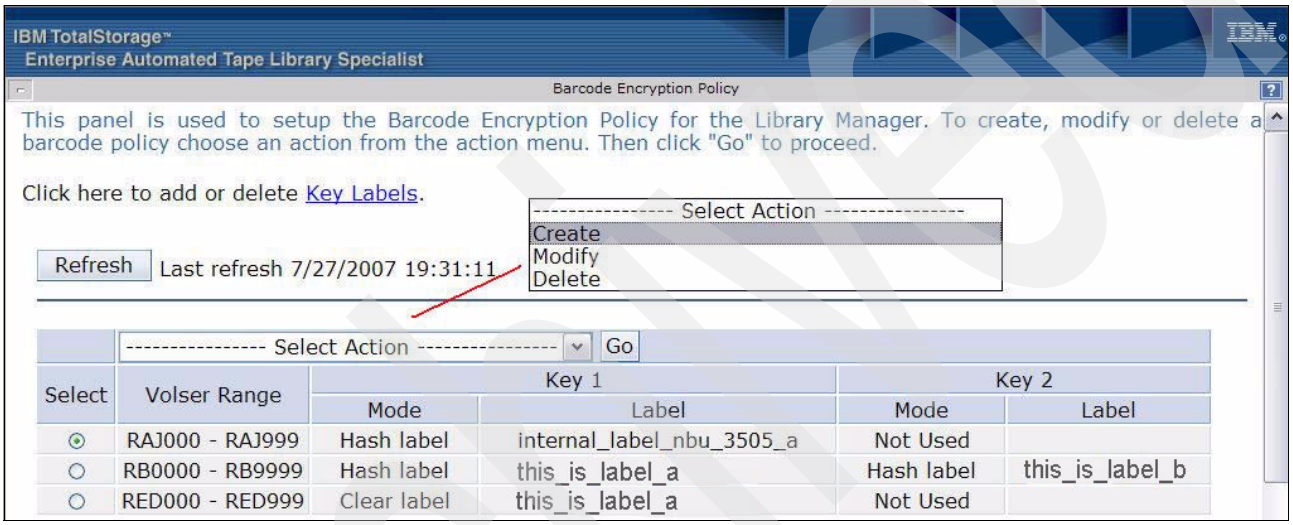


Figure 14-52 Barcode Encryption Policy

To create a new Barcode Encryption Policy, select the **Create** action, and then click **Go**, which results in the panel shown in Figure 14-53 on page 503.

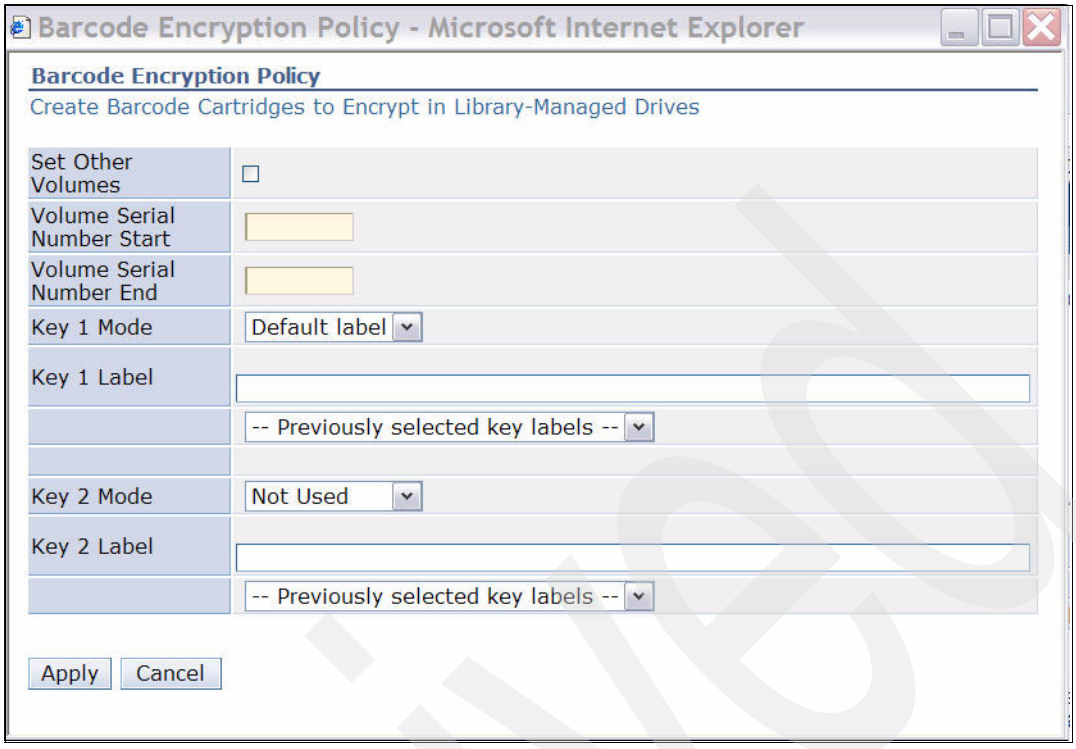


Figure 14-53 Blank specify barcode encryption policy panel

As described previously, a *Barcode Encryption Policy* is a range of cartridge VOLSERs and specifies which scratch cartridges to encrypt on the next attempt to write from the beginning of the tape; it does not indicate which cartridges are currently encrypted. When used with LME, a Barcode Encryption Policy controls cartridge encryption by VOLSER ranges in *all* logical libraries that have LME with Barcode Encryption Policy selected. Stated another way, if you have defined multiple policies (overlap is not allowed), they all are effective simultaneously, and they affect which cartridges are encrypted in every defined library-managed logical library in a TS3500 that is enabled to encrypt using Barcode Encryption Policies.

When implementing a Library-Managed BEP, you not only specify which cartridges to encrypt but also the public keys that are used to encrypt the data key. You manage this by directing future allocations to certain Barcode Encryption Policies through VOLSER ranges. The BEP that you select depends on the cartridge destination.

For our example, which is shown in Figure 14-54 on page 504, we create a BEP to encrypt all cartridge VOLSERs from `ATS000` thorough `ATS999`. Each cartridge that falls within that sequence will be encrypted using a public key referenced by *label a* for key label 1 and a public key referenced by *label b* for key label 2. When a certificate is imported into a keystore, a label is assigned to it. You enter this label on this panel, which in our example is `label b` and `label a`. When an encrypted cartridge is created, two EEDKs are also created and stored on the cartridge. In addition to the encrypted Data Key, each EEDK has a pointer to the public key that was used to encrypt the Data Key. The pointer can either be the label of the certificate or a Hash value based on the public key. Regardless of which method is used, EKM uses the value to identify the correct private key to use to decrypt the encrypted data key.

Because we do not plan to send this cartridge off-site, we create each label in clear mode. For an explanation of clear label mode compared to the Hash label, refer to “Creating a BEP using Hash values” on page 472.

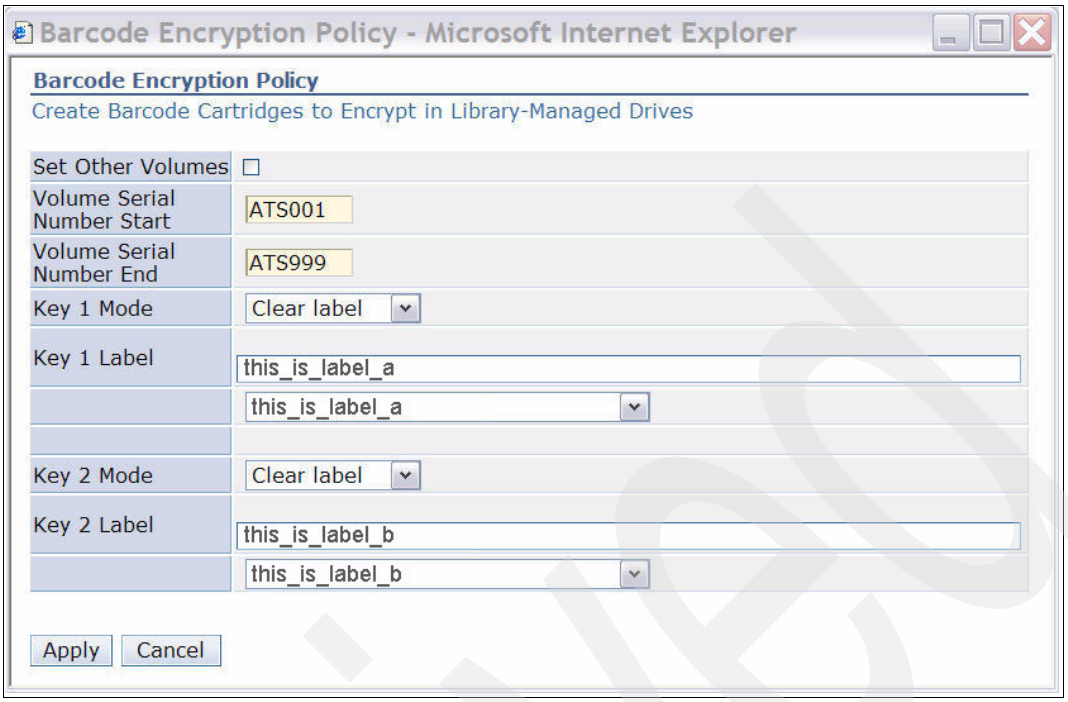


Figure 14-54 Barcode policy create

Figure 14-55 reflects the new barcode encryption policy that we just created.

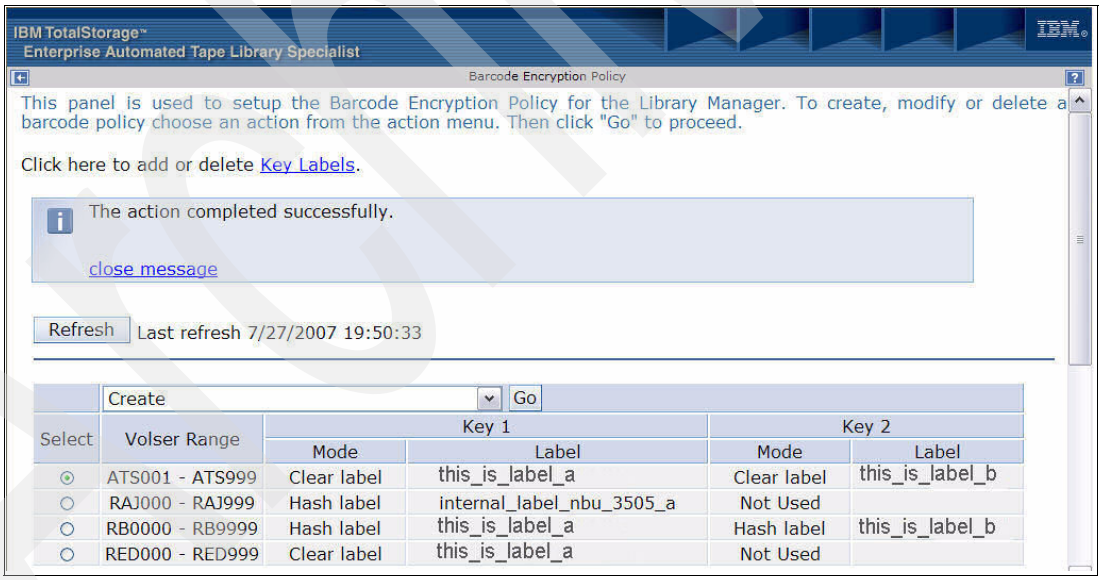


Figure 14-55 Newly created barcode encryption policy

With the following Barcode Encryption Policies in place, imagine this scenario. What if a cartridge with the VOLSER of RAJ001 is mounted on a drive that has LME with the Barcode Encryption Policy specified? Will it be encrypted? What keys will be used to encrypt the data key (DK)? Remember that on the TS1120, the DK is enclosed within a data structure called EEDK1 and EEDK2. Each EEDK includes the DK encrypted by a public key, which is pointed to by the key label specified in the Barcode Encryption Policy.

Therefore, in our scenario, yes, the cartridge will be encrypted, because it falls within the VOLSER range of the RAJ000 - RAJ999 BEP. The first EEDK (EEDK1) on the cartridge will contain the DK encrypted with the key pointed to by the following label and in HASH format:

`internal_label_nbu_3505_a`

Key 2 is not specified, so a default label value will be used to encrypt the data key and create EEDK 2. The order of default key labels for the TS1120 to use are:

1. Drive table, if specified
2. EKM `configuration.properties` file, if no key label is specified in the drive table

14.8.3 Entering the EKM IP address and key labels

Before you can specify a key label as part of a BEP or within the key label remapping function, identify the key label to the 3494, as shown in the panel in Figure 14-56.

To open this panel from the 3494 Web GUI, select **Administer Library Manager** → **Manage Encryption** → **Key Label Entry / EKM IP Info**. The current maximum number of labels that can be specified is 64. This panel is also where you specify the IP address for the Encryption Key Manager (EKM) that is used for LME.

Important: The IP address specified in this panel is for LME-enabled drives only. If using SME on Open Systems, the pointer for the EKM is kept within the device driver configuration file. AME does not use an EKM. The z/OS uses SME only.

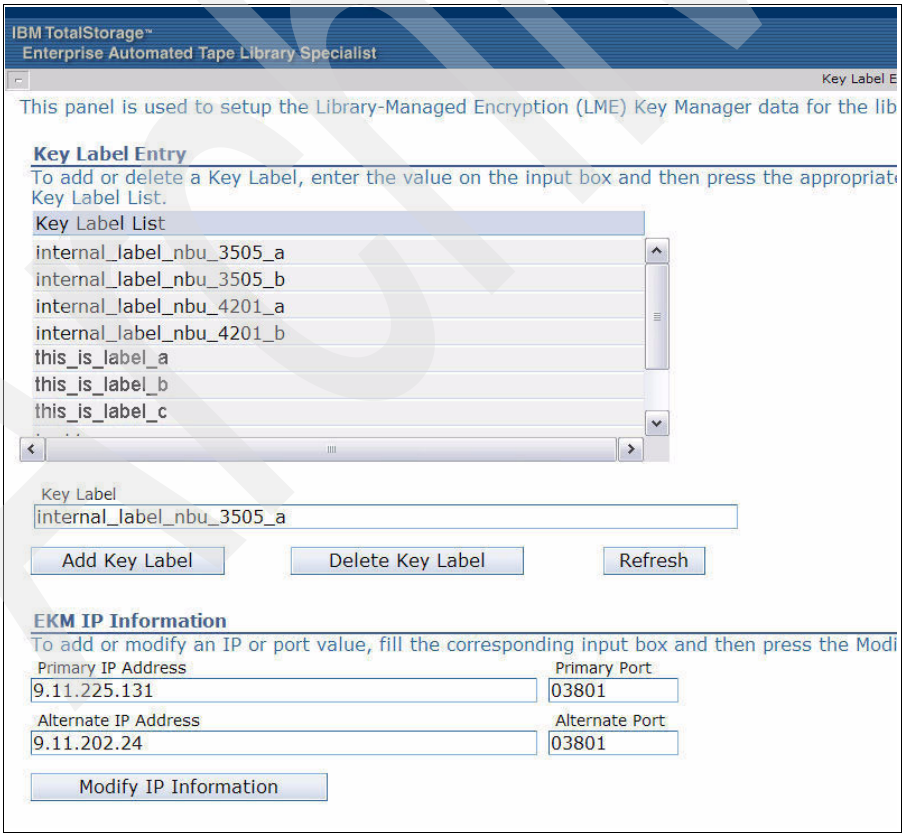


Figure 14-56 Key label entry

14.8.4 ILEP key label mapping

If implementing Internal Label Encryption Policies (ILEP), a function within the 3494 Web GUI allows you to map the ILEP-generated label into a more user-friendly label. ILEP-generated labels can get quite long and are not easy to read.

To access the panel shown in Figure 14-57, select **Administer Library Manager** → **Manage Encryption** → **ILEP Key Label Mapping**. In the example, you can see that for testing purposes we remapped the label `internal_label_nbu_3505_a` to a key label named `test1`.

Also, be aware that both labels must already exist in the key label table, which we discussed in 14.8.3, “Entering the EKM IP address and key labels” on page 505. If the key labels do not already exist in the table, you cannot remap them using this function.

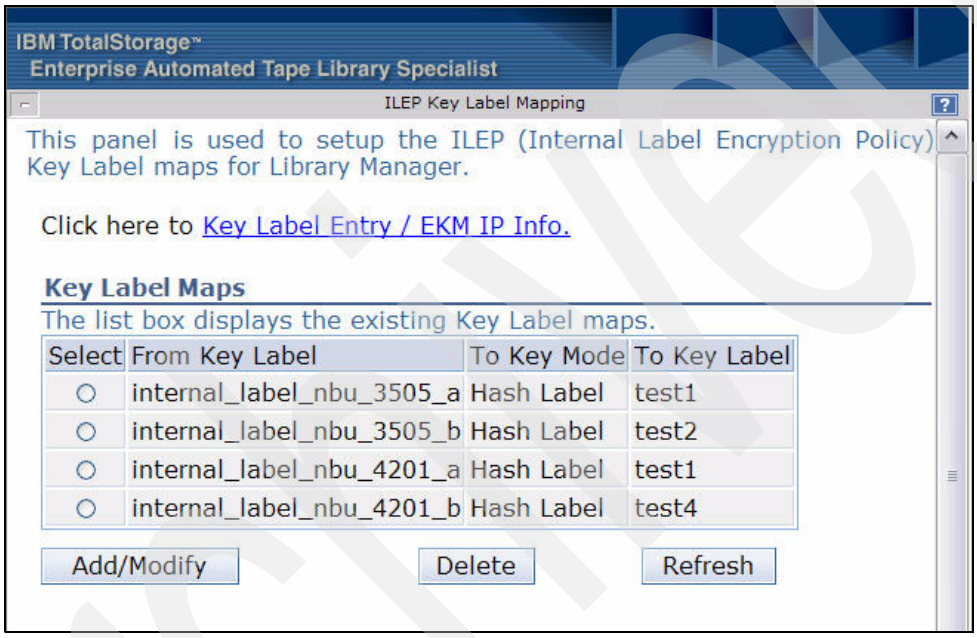


Figure 14-57 ILEP Key label Mapping

This completes our discussion of the 3494 and TS1120 on Open Systems.

15

Tape data encryption with i5/OS

In this chapter, we describe Library-Managed Encryption (LME) with the IBM TS1120 Enterprise Tape Drive Model 3592-E05 and the IBM LTO4 tape drive TS1040 for i5/OS environments.

We discuss the following topics:

- Planning for tape encryption with i5/OS and clarification of the software and hardware prerequisites together with considerations for disaster recovery and sharing tape cartridges with partners
- Detailed implementation procedures that provide guidance for configuring LME for TS1120 and LTO4 drives

15.1 Planning for tape data encryption with i5/OS

In this section, we provide planning information about the hardware and software prerequisites for using tape encryption with i5/OS and the IBM TS1120 enterprise tape drive and the IBM TS1040 LTO4 tape drive. Important considerations pertaining to the selection of the EKM keystore, the encryption policies, the EKM setup for disaster recovery, and sharing tape cartridges with partners are discussed. Then, we provide an overview of the implementation steps required for tape encryption with i5/OS.

Note: Within this chapter, we focus on Library-Managed Encryption (LME) with the Encryption Key Manager (EKM) running on i5/OS. There is no requirement to have EKM installed on i5/OS to use tape encryption for i5/OS, because with LME, the EKM can be installed on any supported platform.

15.1.1 Hardware prerequisites

Currently, the following IBM tape drives support hardware tape encryption:

- IBM TS1120 Enterprise Tape Drive encryption-capable with feature code (FC) 9592 (earlier TS1120 models can be upgraded to be encryption-capable through the chargeable feature FC5592)
- IBM TS1040 LTO4 Fibre Channel or serial SCSI (SAS) tape drive (encryption support for LTO4 tape cartridges only)

To enable these encryption-capable IBM tape drives for LME, which is required for i5/OS, they must reside in one of the following supported IBM tape libraries:

- TS1120 supported in:
  - IBM TS3400 library
  - TS3500 library frame models L23 or D23
- TS1040 LTO4 supported in:
  - IBM TS3100 and TS3200 libraries (AAS orders only) Release 4 or later with:
    - Transparent LTO Encryption feature (FC5900)
  - IBM TS3310 library Release 4 or later with:
    - Transparent LTO Encryption feature (FC5900)
    - Encryption Configuration (FC9000)
  - IBM TS3500 library frame models L53 or D53 Release 7A' or later with:
    - Transparent LTO Encryption feature (FC1604)
    - TS1040 Fibre Channel model 3588-F4A only

The minimum TS1120 drive firmware level to support LME is 1942, but the best practice is that you have the latest PFE recommended-level applied. Refer to the RMSS PFE Communications Web site for 3592 at:

http://snjlnt02.sanjose.ibm.com/tape/tapetec.nsf/pages/3592page00

For the IBM TS3500, use FC1690, Advanced Library Management System (ALMS), to flexibly set encryption methods at a logical library level and allow the intermixing of encryption-capable drives with non-encryption-capable drives within a logical library.

Without ALMS, all logical libraries of the same technology (either TS1120 or LTO) must be set to the same encryption mode, which has the following implications:

- Encryption-capable drives added to a non-ALMS library with older non-encryption-capable drives cannot be encryption-enabled.
- Older non-encryption-capable drives added to a non-ALMS encryption library will be set to “restricted mode” and not be available for use.

15.1.2 Software prerequisites

To use tape encryption with the IBM Encryption Key Manager component for the Java™ platform (EKM) installed on i5/OS, the following software prerequisites must be met:

- i5/OS V5R3 or later
- Crypto Access Provider 128-bit, product number 5722-AC3 (*V5R3 only*)
- IBM HTTP Server for i5/OS, product number 5722-DG1 (*TS1120 tape encryption with IBMi5OSKeyStore only*)
- Digital Certificate Manager, product number 5722-SS1, option 34 (*TS1120 tape encryption with IBMi5OSKeyStore only*)

> **Note:** The i5/OS Digital Certificate Manager is used to administer an *IBMi5OSKeyStore* for TS1120 tape encryption and does not support the symmetric keys required for LTO4 tape encryption.

- Java Developer Kit 1.5, product number 5722-JV1, *BASE and option 7 (*V5R3 only*)
- J2SE 5.0 32 bit, product number 5722-JV1, *BASE and option 8 (*V5R4 only*)
- The latest Java Group PTF (SF99269 for *V5R3* and SF99291 for *V5R4*)
- 5722-JV1 PTF SI26811 providing IBM Java 5.0 Service Release 4 containing the EKM Release 1 code (*V5R4 only*)
- 5722-SS1 PTF SI25094 providing the EKM default configuration file and strEKM script (*V5R4 only*)
- 5722-SS1 PTF SI26705 providing the IBM EKM Release 1 code, default configuration file and strEKM script (*V5R3 only*)
- 5722-BR1 PTF SI24934 providing a new media density FMT3592A2E for Backup, Recovery and Media Services (BRMS) to help identify encrypted tape cartridges (*V5R4 only*)
- 5722-BR1 PTF SI24933 providing a new media density FMT3592A2E for BRMS to help identify encrypted tape cartridges (*V5R3 only*)

> **Note:** LME for LTO4 tape drives requires the *EKM Release 2 code* and *IBM Java 5.0 Service Release 5,* which adds support for handling symmetric data keys.

The plan is to release EKM Release 2 code (build 20070503) for i5/OS with IBM Java 5.0 Service Release 5, including the IBM Java keytool that is required for the support of generating and storing symmetric keys, which will be made available as a 5722-JV1 PTF for i5/OS. To download a more recent version of the EKM code, refer to the IBM support Web site at:

`http://www-1.ibm.com/support/docview.wss?rs=1139&context=STCXRGL&dc=D400&uid=ssg1S4000504`

15.1.3 Disaster recovery considerations

The main reason against using a single EKM server is that if this EKM server fails, there is no way to read from or write to newly mounted encrypted tapes before you manually recover the failed EKM server.

Note: For availability reasons, we suggest that you set up two redundant EKM servers on different host systems that are ideally at different locations.

Another reason to set up a secondary EKM server is that an EKM server that was started in a batch job *currently* cannot be dynamically reconfigured using the EKM Admin Console. Figure 15-1 shows a redundant two EKM server setup. If one EKM server fails, the IBM tape library, which has been configured with two redundant EKM paths, that is, EKM server TCP/IP addresses, will automatically attempt to fail over to the second EKM server to retrieve the data keys required for the encryption and decryption of the tape cartridges. Having two redundant EKM servers requires keystore, configuration file, and drive table synchronization, which you can do automatically at regular intervals through defined parameters in the EKM configuration file.

For additional details about setting up a secondary EKM server and synchronizing EKM servers, refer to “Additional information” on page 204.

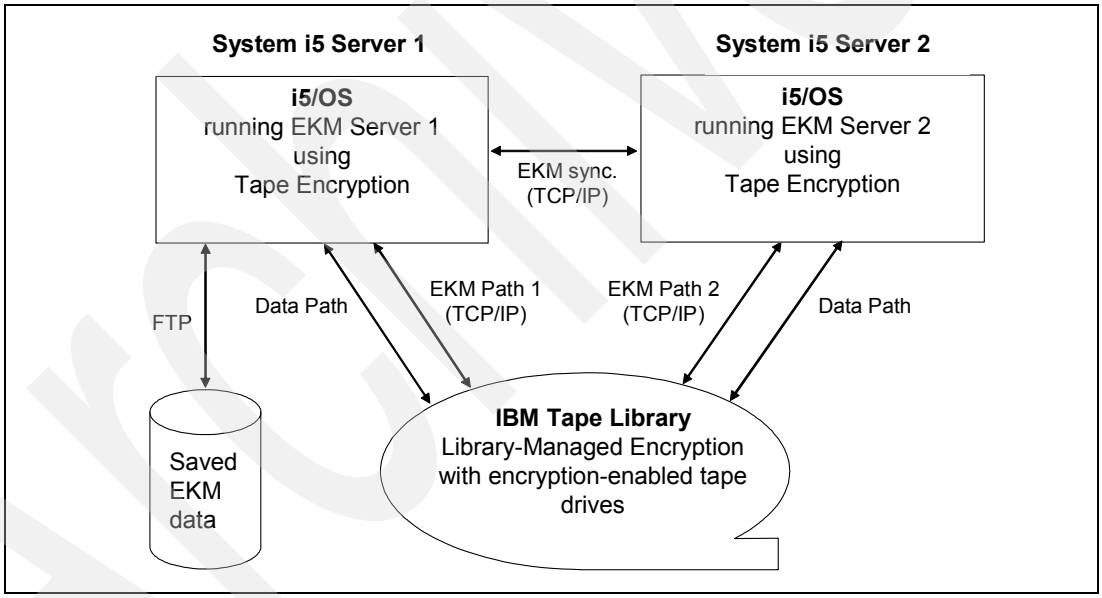


Figure 15-1 Redundant EKM server configuration using two systems

Important: Taking regular *unencrypted* backups of the EKM keystore `*.KDB` file, drive table, and configuration file `KeyManagerConfig.properties`, for example, through prior transfer (using FTP) to another system, is crucial for restoring a working EKM server configuration after a disaster before you can regain access to the encrypted tape data.

If EKM server availability and recovery time is not an issue and you want a single EKM server configuration, be aware of disaster recovery implications. For example, install the EKM server on another logical partition (LPAR) or on a system other than the system from which you take system backups with tape encryption. Otherwise, you will not have a way to quickly perform a

system-restore from the encrypted tapes without a lengthy and error-prone configuration of a new EKM server from unencrypted saved EKM data.

For this reason, this distributed EKM server configuration, which is shown in Figure 15-2, is the only *single* EKM server configuration that is supported by IBM Rochester System i development.

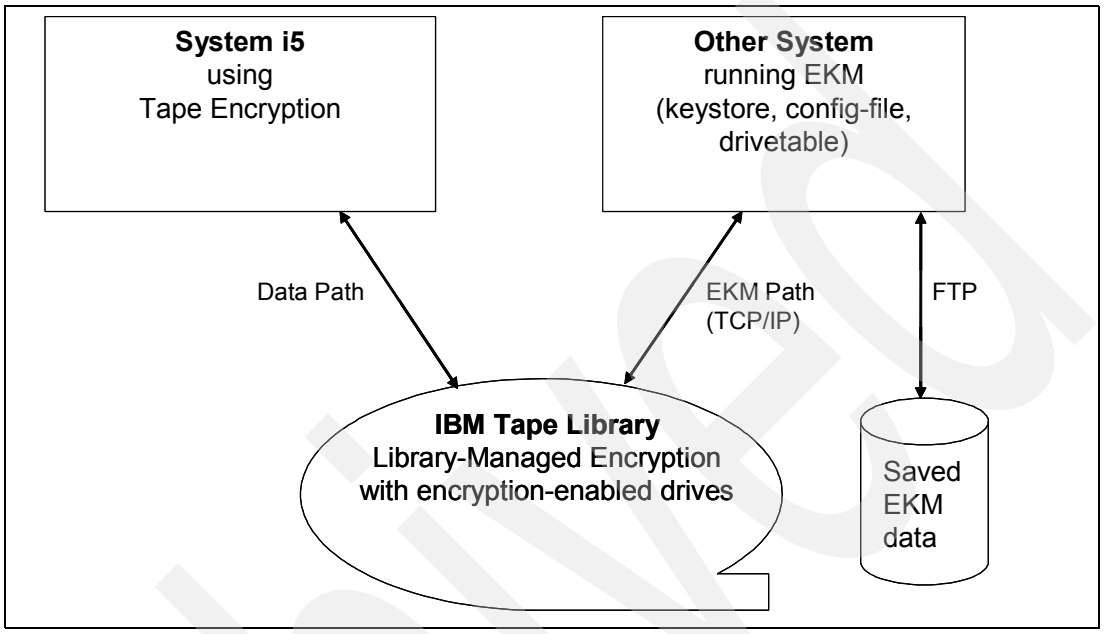


Figure 15-2 Single distributed EKM server configuration

15.1.4 EKM keystore considerations

The two types of supported keystores for installing the Encryption Key Manager (EKM) on i5/OS are:

- An *IBMi5OSKeyStore* that is managed through i5/OS Digital Certificate Manager, which is i5/OS specific and supports no storage of symmetric keys so that it can be used for TS1120 tape encryption only
- A Java Cryptography Extension Keystore (*JCEKS*) that is supported on all platforms and supports both LTO4 encryption with storage of symmetric keys and TS1120 tape encryption with storage of asymmetric keys.

Use the *IBMi5OSKeyStore* only for TS1120 tape encryption environments where there is no LTO4 encryption used and where EKM (both a primary and an optional secondary EKM server for high availability) is installed on i5/OS only.

In all other situations, a *JCEKS* keystore is either required when LTO4 encryption is used or highly recommended when the EKM primary server and secondary server reside on different platforms in order to ease synchronization of both of the EKM servers. For information about automatic synchronization between two EKM servers, refer to “Additional information” on page 204.

Note: Each EKM server only supports *one* keystore so you must decide whether to use an *IBMi5OSKeyStore* or *JCEKS* keystore.

A comparison between the supported features of a JCEKS keystore and IBMi5OSKeyStore is shown in Table 15-1.

Table 15-1 Comparison between the IBMi5OSKeyStore and the JCEKS keystore

Supported feature	IBMi5OSKeyStore	JCEKS
TS1120 (asymmetric keys)	Yes	Yes
LTO4 (symmetric keys)	No	Yes
Supported platforms	i5/OS only	All
Graphical user interface	Yes	No

15.1.5 TS1120 Tape Encryption policy considerations

Three methods of defining encryption policies are available to determine which encryption keys from the EKM keystore (referenced by key labels or aliases) are used for TS1120 tape encryption. These methods in the order of priority that they are used by EKM are:

- Default global key aliases defined in the EKM configuration file
- Default drive key aliases defined in the EKM drive table
- Barcode Encryption Policies defined in the IBM TS3500 Tape Library

Default global key aliases are defined in the EKM configuration file through the parameters drive.default.alias1 and drive.default.alias2, which can be set to the default key labels to be used for the EEDK1 and EEDK2. We recommend that you define default global key aliases when using the parameter setting **drive.acceptUnknownDrives = true** to make sure that certificates for generating or decrypting EEDK1 and EEDK2 are associated with new drives that are automatically added to the drive table. To implement default global key aliases, refer to “Customizing the EKM Configuration File” on page 202.

Default drive key aliases are used to associate certificates with 3592 drive serial numbers. They are defined by using the rec1 and rec2 parameters when you manually create or modify an entry in the EKM drive table using the EKM Admin Console commands **`adddrive`** and **`moddrive`**. Defining default drive key aliases is useful if different (logical) tape libraries communicating with the same EKM server use different encryption keys, for example, because only a subset of libraries is used for sharing tape cartridges with business partners. For reference information about the EKM Admin Console commands, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide*, GA76-0418.

Barcode Encryption Policies (BEP) are supported with the IBM System Storage TS3500 Tape Library only. They are used to specify which VOLSER ranges are to be encrypted (and which are not) and which certificates referred to by the specified *key labels* are to be used for the encryption process. A *key mode* specified for the two key labels determines the method by which EKM identifies the public-private keys to be used for generating and decrypting the EEDKs. Choices for key mode are:

Default label The key label, default drive key alias or default global key alias, which is configured at the encryption key manager, will be used.

Clear label The key is referenced by the specified key label.

Hash label The key is referenced by a computed value from the public key that is referenced by the specified key label.

Using a Hash label is especially useful for sharing tapes with a business partner, because the certificate can be imported into the keystore with another label than the label with which it was

exported. That way, both partners do not have to agree on a common key label to use. For additional information about sharing tape cartridges, refer to 15.1.6, “Considerations for sharing tapes with partners” on page 513.

Figure 15-3 shows an example from the IBM System Storage TS3500 Tape Library’s IBM UltraScalable Specialist **Cartridges** → **Barcode Encryption Policy** view with a user-defined BEP for the VOLSER range J10000 - JZZZZZ using a Hash label to refer to the public key certificate from the business partner.

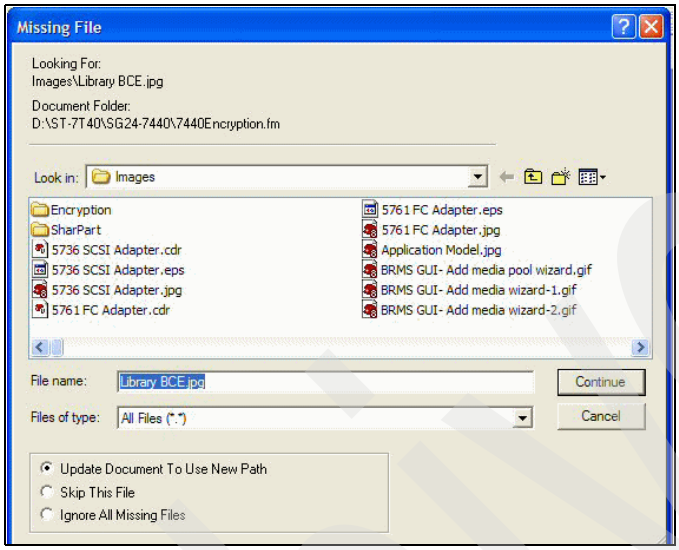


Figure 15-3 IBM UltraScalable Specialist: TS3500 Tape Library Barcode Encryption Policy panel

15.1.6 Considerations for sharing tapes with partners

This section describes approaches that might apply for sharing encrypted 3592 tape cartridges and encrypted LTO4 tape cartridges.

Sharing encrypted 3592 tape cartridges

If encrypted 3592 tape cartridges are to be shared with partners, we do not recommend that you provide the partners with the private part of the key encryption key (KEK) that is used for decryption for security reasons. A *private key* is intended to be safely kept and never provided to other organizations. Sharing a private key is a security risk, because anyone else with access to your private key can read your encrypted tape data.

Note: A best practice is to share encrypted 3592 tapes with partners by importing your business partner’s certificate, including only the *public key* but not the private key, into your EKM keystore.

On i5/OS with the Digital Certificate Manager (DCM) or on other platforms with the IBM Key Management tool as part of the IBM Java RTE, the digital certificate used for tape encryption can be exported into a file that can be imported into a partner’s EKM keystore. You must be careful not to export the full certificate, which holds the public key and corresponding private key, but only the public key part of the certificate. Section 15.2.3, “Importing and exporting encryption keys” on page 538 describes the detailed procedures to export and import public-private key certificates as *PKCS12* binary files and public key only certificates as *Base64_encoded* text files.

Two sets of EEDKs can be stored on a 3592 tape cartridge (specify your partner's public key certificate as the second key label), depending on your encryption policy either in your EKM configuration file if using default key labels, in the drive table entries, or in your TS3500 library Barcode Encryption Policy. Enable only the selected business partner, who owns the corresponding private key, to read your 3592 cartridges that are newly written with encryption from beginning of tape (BOT).

Note: Existing encrypted 3592 tape cartridges will continue to use their EEDK1 and EEDK2 that were originally stored with the first write on the cartridge, even if the EKM encryption policy is changed.

To share already encrypted 3592 cartridges with existing data, you may *rekey* the cartridges with the IBM Tape Library Specialist Web GUI. Refer to "Rekeying encrypted 3592 cartridges" on page 552.

For exporting certificates, including the public key and the private key from an i5/OS keystore, to an EKM keystore on a platform other than i5/OS for redundancy or disaster recovery, consider that the i5/OS DCM exports certificates in the PKCS 12 version 3 file format. Therefore, the target keystore must support the same format, or the exported certificate file must be converted, for example, by using the OpenSSL open source utility.

Sharing encrypted LTO4 tape cartridges

The possibilities for sharing encrypted LTO4 cartridges are very limited compared to the inherent Business-to-Business model of the TS1120 two-layer encryption algorithm. TS1120 Encryption supports sharing of encrypted cartridges by securing the data key with two different certificates. LTO Tape Drive Encryption use a single-layer symmetric encryption algorithm.

What always works is to provide the partner the symmetric key certificate that was used for a specific encrypted LTO4 cartridge. The EKM audit metadata XML file specified in the EKM configuration file provides the information regarding which symmetric key was used for a specific cartridge volume serial number. Refer to "Example of the EKM audit metadata XML file" on page 553.

However, sharing your symmetric data key implies that the security risk that anyone else who obtains your symmetric data key can read your LTO4 cartridges, which were encrypted with this same data key. You can create a *set* of symmetric keys in the EKM keystore to be used across the pool of LTO4 cartridges. However, this serves for increased security than sharing cartridges with partners, because there is no control over which key alias from this set will be used for a specific LTO4 cartridge serial number. With the IBM TS3500 tape library, a feasible workaround to prevent sharing your own symmetric data keys might be to define a dedicated data key within a Barcode Encryption Policy for a LTO4 cartridge serial number range to be shared with the partner.

15.1.7 Steps for implementing tape encryption with i5/OS

This section provides an overview of the steps required for implementing tape encryption with i5/OS. We also summarize the steps required for installing the EKM in i5/OS that we described in 6.4, "Installing the EKM in i5/OS" on page 196.

Prerequisites and considerations include:

- Ensure that hardware and software prerequisites are met. Refer to 15.1.1, "Hardware prerequisites" on page 508 and 15.1.2, "Software prerequisites" on page 509.

- Consider the EKM backup and disaster recovery concept. Refer to 15.1.3, “Disaster recovery considerations” on page 510.
- Determine the type of EKM keystore to be used. Refer to 15.1.4, “EKM keystore considerations” on page 511.
- Determine the encryption policies to be used. Refer to 15.1.5, “TS1120 Tape Encryption policy considerations” on page 512.
- Determine requirements for key and certificate, especially for sharing cartridges. Refer to 15.1.6, “Considerations for sharing tapes with partners” on page 513.

Implementing tape encryption with i5/OS requires the following major steps:

1. Install a primary or single EKM server on i5/OS. Refer to 6.4.1, “New installation of the Encryption Key Manager” on page 196.
2. Create an EKM keystore and encryption keys. Refer to 15.2.1, “Creating an EKM keystore and certificate” on page 515.
3. Configure EKM for TS1120 or LTO4 tape encryption. Refer to 6.4.3, “Configuring EKM for tape data encryption” on page 201.
4. Configure the IBM tape library for LME. Refer to 15.2.2, “Configuring the TS3500 library for Library-Managed Encryption” on page 528.
5. Back up the EKM data (keystore, configuration file, and drive table file).
6. Optionally install a secondary EKM server on another i5/OS system. Refer to 6.4.1, “New installation of the Encryption Key Manager” on page 196.

15.2 Setup and usage of tape data encryption with i5/OS

In this section, we describe the EKM configuration and setup of LME in the IBM tape library for TS1120 tape encryption or LTO4 tape encryption. We assume that the IBM Encryption Key Manager component for the Java™ platform (EKM) has already been installed as described in steps 1 to 5 in 6.4.1, “New installation of the Encryption Key Manager” on page 196.

15.2.1 Creating an EKM keystore and certificate

Refer to the planning section, 15.1.4, “EKM keystore considerations” on page 511, to decide which type of the required keystore to use for EKM on i5/OS before going to the corresponding following sections to create an EKM keystore:

- To create an *IBMi5OSKeyStore,* see “Creating an IBMi5OSKeyStore keystore and certificate” on page 515.
- To create a *JCEKS* keystore, see “Creating a JCEKS keystore and certificate” on page 526.

Creating an IBMi5OSKeyStore keystore and certificate

To create an IBMi5OSKeyStore and certificate:

1. Use a Web browser to connect to the i5/OS HTTP admin server at the address `http://ipaddress:2001` and click **Digital Certificate Manager** as shown in Figure 15-4 on page 516. You access the i5/OS Digital Certificate Manager that is used for creating and managing an IBMi5OSKeyStore.

Figure 15-4 HTTP admin server i5/OS Tasks entry panel

Note: If you experience connection problems from the Web browser to the i5/OS HTTP admin server, use WRKACTJOB on i5/OS to check that the HTTP admin server QHTTPSVR/ADMIN is running. If not, start it by entering the following command:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
```

2. Select **Create New Certificate Store**. Choose **Other System Certificate Store**, and then click **Continue**, as shown in Figure 15-5 on page 517.

Figure 15-5 DCM: Create New Certificate Store panel

The next panel opens, as shown in Figure 15-6.

Figure 15-6 DCM: Create a Certificate in New Certificate Store panel

3. Select **No - Do not create a certificate in the certificate store** and then click **Continue**. The panel, shown in Figure 15-7 on page 518, opens.

Figure 15-7 DCM: Certificate Store Name and Password panel

4. Enter the certificate store path and filename, for example, `/EKM/EKM.KDB`, making sure to use the path for the IFS directory created for EKM. Specify a certificate store password. Both will be required later for configuring EKM. Click **Continue**.

   The message "`The certificate store has been created`" indicates the successful creation of the certificate store as shown in Figure 15-8.

Figure 15-8 DCM: Certificate Store Created panel

5. Select **Create a Certificate Authority (CA)**. Choose a key size of `1024` bits, enter the required local CA certificate information as in the example shown in Figure 15-9 on page 519, and click **Continue** to proceed.

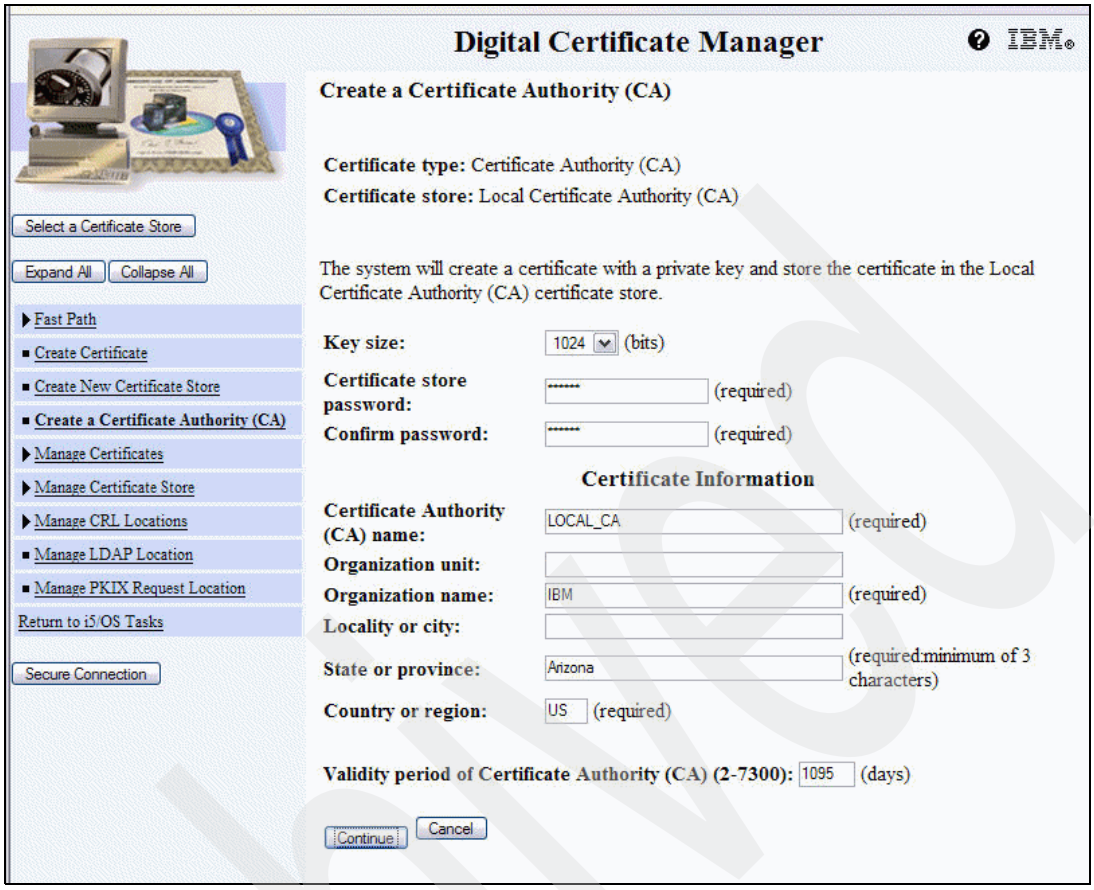


Figure 15-9 DCM: Create a Certificate Authority panel

Note: The **Create a Certificate Authority (CA)** menu option displays if no local CA has been created yet. If a local CA already exists, proceed directly to step 10.

The the Install Local CA Certificate panel shown in Figure 15-10 on page 520 opens.

Figure 15-10 Install Local CA Certificate panel

6. Click **Continue** to create the local CA certificate. Installing the local CA certificate in your browser is not necessary because the local CA is not meant to be used for Secure Sockets Layer (SSL) Web connections.

   The panel shown in Figure 15-11 on page 521 opens.

7. For **Allow creation of user certificates**, select **Yes** and specify the validity period (in days) for the certificates, which are issued by your local CA, that you will use for tape encryption. This validity period must be shorter than the validity period of the CA certificate itself.

   Note: EKM does *not* care about the validity period or expiration of certificates. It also works with the keys from expired certificates as long as they remain in the keystore.

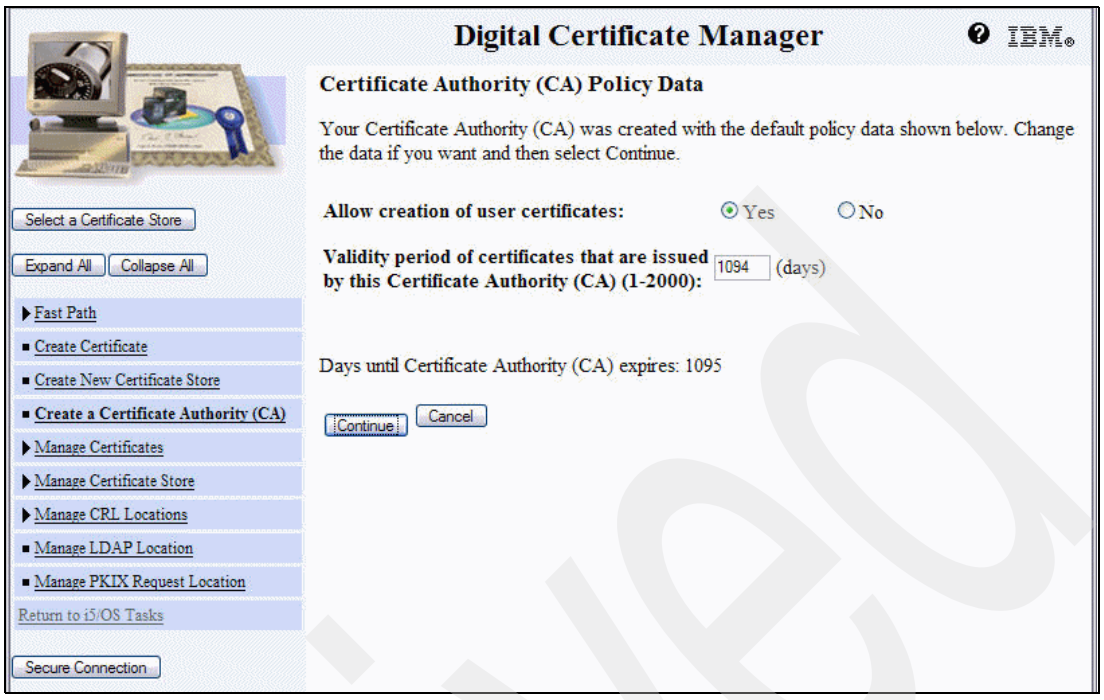


Figure 15-11 Certificate Authority (CA) Policy Data panel

8. Click **Continue.**

   The successful modification of the local CA policy data is indicated by the message `The policy data for the Certificate Authority (CA) was successfully changed`, as shown in Figure 15-12.

   Because the local CA certificate will be used to issue self-signed certificates used for tape encryption, click **Cancel** so a server certificate store is not created.

Figure 15-12 DCM: Policy Data Accepted panel

9. After creating the local CA keystore and certificate, select the EKM keystore (created in a previous step) by using **Select a Certificate Store**. Select **Other System Certificate Store** as shown in Figure 15-13. Then, click **Continue**.

Figure 15-13 DCM: Select a Certificate Store panel

10. Enter the certificate store path, file name, and certificate store password of the previously created EKM keystore. Then, click **Continue**. Refer to Figure 15-14.

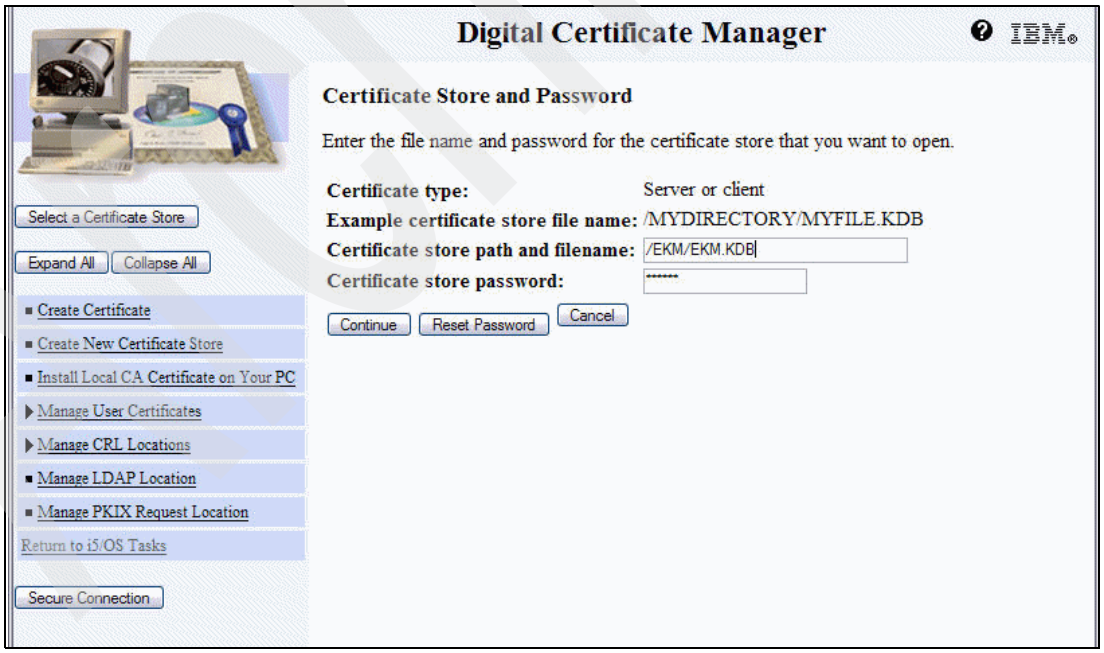


Figure 15-14 DCM: Certificate Store and Password panel

Successful selection of the certificate store is indicated as shown in Figure 15-15 on page 523.

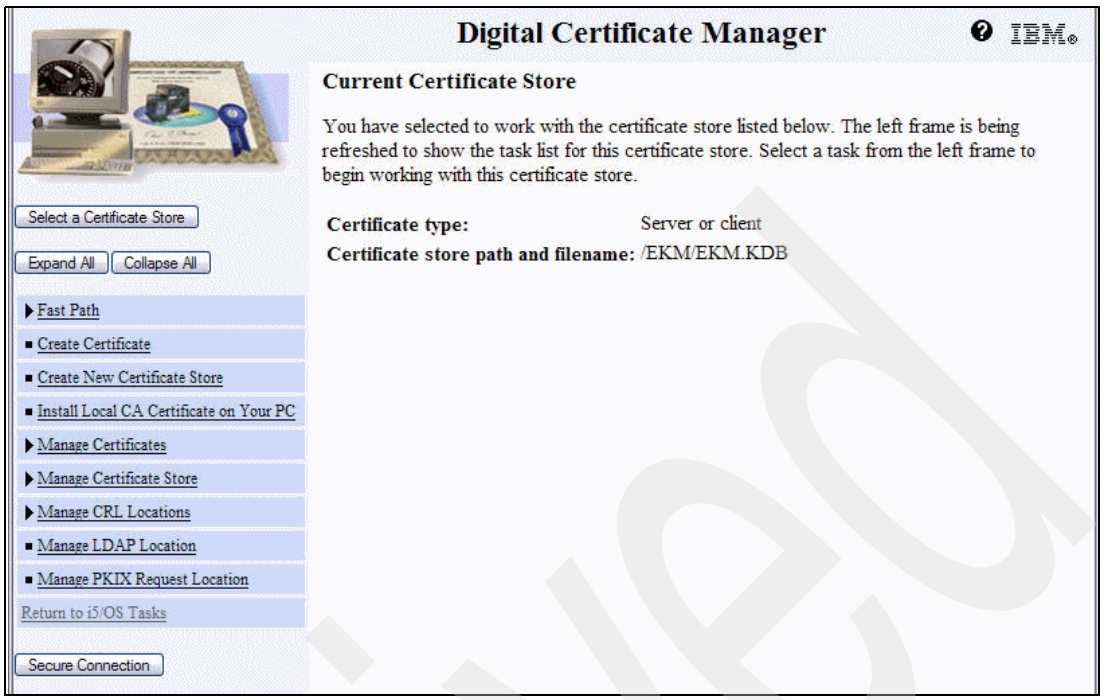


Figure 15-15 DCM: Current Certificate Store panel

11. Select **Fast Path** → **Work with server and client certificates** and click **Create** as shown in Figure 15-16.

Figure 15-16 DCM: Work with Server and Client Certificates panel

12. Select **Local Certificate Authority** for creation of a self-signed certificate to be used for tape encryption and click **Continue** as shown in Figure 15-17 on page 524.

Figure 15-17 DCM: Select a Certificate Authority panel

13. Select 1024 bits for key size, specify a **Certificate label**, which you must record to use later when configuring the EKM, and complete the required **Certificate Information** fields with your identity information before clicking **Continue** to proceed as shown in Figure 15-18 on page 525.

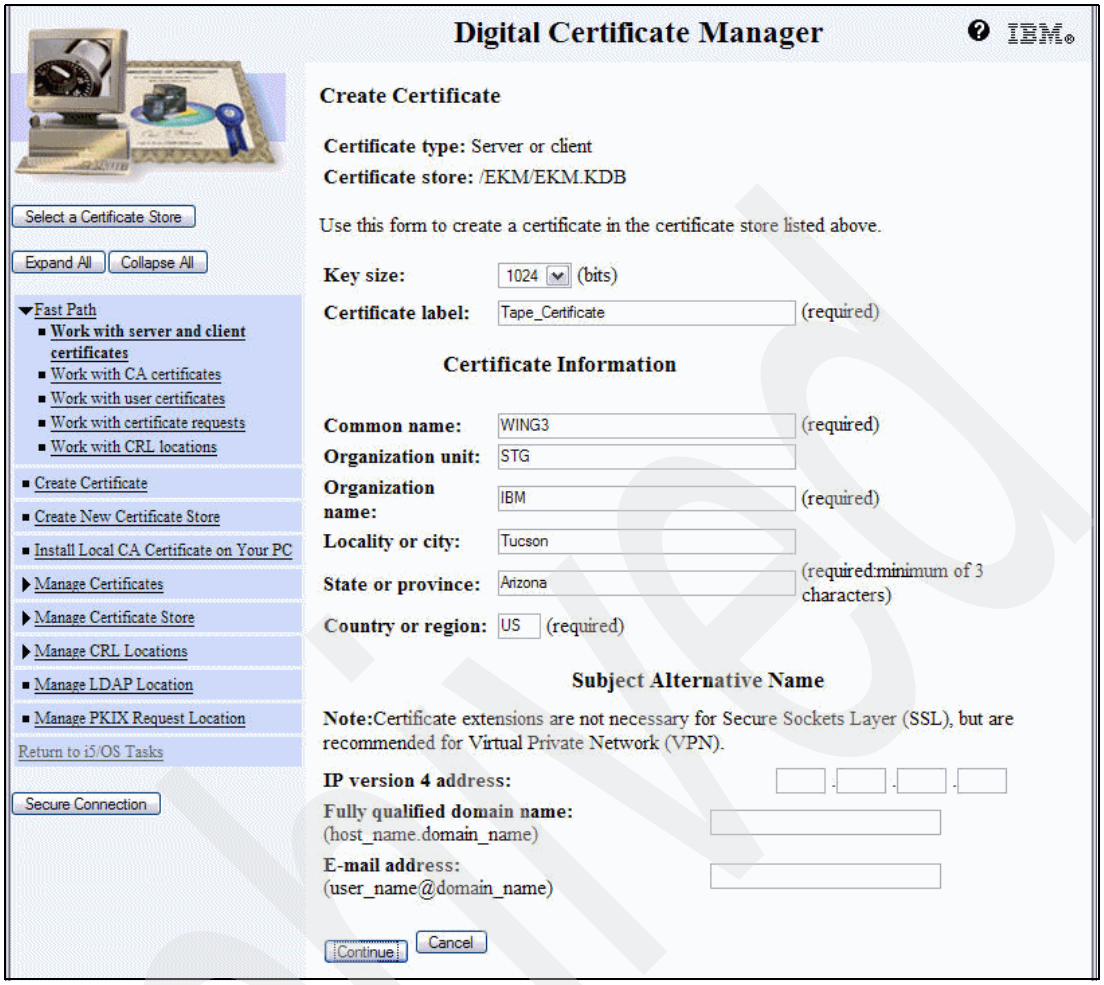


Figure 15-18 DCM: Create Certificate panel

Successful creation of your self-signed certificate in your EKM keystore is indicated by the message `Your certificate was created and placed in the certificate store listed below`, as shown in Figure 15-19.

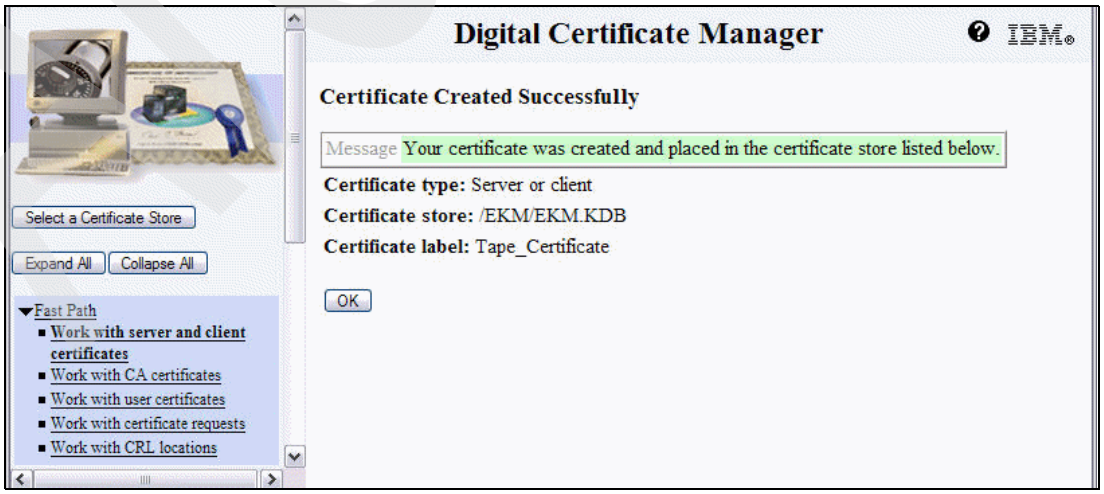


Figure 15-19 DCM: Certificate Created Successfully panel

Now, you have created an EKM keystore with a self-signed public-private key certificate to be used for TS1120 tape encryption. To create a second certificate to be shared with a business partner as discussed in 15.1.6, “Considerations for sharing tapes with partners” on page 513, repeat steps 11 on page 523- 13 on page 524, and specify a different certificate label. Return to 6.4.1, “New installation of the Encryption Key Manager” on page 196 and proceed with step 7 on page 197 to continue with configuring EKM for tape encryption.

Creating a JCEKS keystore and certificate

To create a public-private key certificate for TS1120 tape encryption in a JCEKS keystore, refer to “Public-private key generation for TS1120 tape encryption” on page 526. To create a symmetric key certificate for LTO4 tape encryption in a JCEKS keystore, refer to “Symmetric key generation for LTO4 encryption” on page 526.

Note: When using solely LTO4 tape encryption, we recommend that you also create one public-private key as described in “Public-private key generation for TS1120 tape encryption” on page 526. It is required for the SSL communication that is used for synchronization of two EKM servers. A missing public-private key will not cause the EKM server to fail, but it might result in the Java exception message “`No available certificate corresponds to the SSL cipher suites which are enabled.`”

The JCEKS keystore will automatically be created with the corresponding key generation commands if it does not exist yet.

Note: The *keypass* and *storepass* values used in the following key generation procedures must be the same, because EKM allows you to specify only one password in its KeyManagerConfig.properties configuration file. This file is used for both the keystore and each associated key label.

Public-private key generation for TS1120 tape encryption

Use the following command syntax from i5/OS Qshell to create a public-key certificate for TS1120 tape encryption:

```
keytool -genkey -alias Tape_Certificate -dname "CN=WING3" -keystore /EKM/EKM.jck
-keyalg RSA -keysize 1024 -keypass password -storepass password -storetype JCEKS
```

This example creates a new self-signed *public-private key certificate* to be used for TS1120 tape encryption that is generated from the RSA-1024 encryption key algorithm. It is labeled *Tape_Certificate* with the common name *WING3* in the JCEKS keystore */EKM/EKM.jck*, and both the keystore and the certificate are protected by the same specified *password* and valid for 90 days by default.

Note: The validity period of a digital certificate is irrelevant, because certificate expiration does *not* matter for EKM.

Symmetric key generation for LTO4 encryption

The *keytool* script in /usr/bin uses the default Java version configured on your i5/OS system. On i5/OS systems with multiple Java versions, that is, several 5722-JV1 options installed, this might not be the version required for any *symmetric* key management.

Check your current default Java version by running the command:

```
STRQSH CMD('java -version')
```

If your current default Java version is different than J2SE 5.0 32 bit for V5R4 and Java 1.5 for V5R3, set the required version by defining an i5/OS job-level environment variable as shown:

- For V5R4:

```
ADDENVVAR ENVVAR(JAVA_HOME) VALUE('/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit')
LEVEL(*JOB)
```

- For V5R3:

```
STRQSH CMD('print "java.version=1.5" > /EKM/EKMjava.properties')
ADDENVVAR ENVVAR(QIBM_JAVA_PROPERTIES_FILE) VALUE('/EKM/EKMjava.properties')
```

> **Note:** The job-level environment variable previously defined is only valid for the current interactive i5/OS user session. However, we do not recommend using a system-level environment variable for the default Java version, because it might result in incompatibilities with other programs, such as the i5/OS HTTP admin server, which does not work with J2SE 5.0.

Use the following command in Qshell to generate symmetric keys for LTO4 tape encryption:

```
keytool -genseckey -alias LTO_Key -keypass password -keyalg AES -keysize 256
-keystore /EKM/EKM.jck -storepass password -storetype JCEKS
```

This example creates a new symmetric (or secret) key certificate to be used for LTO4 tape encryption, which was generated from the AES-256 encryption key algorithm, labeled *LTO_Key* in the JCEKS keystore */EKM/EKM.jck*. Both the keystore and the certificate are protected by the same specified *password*.

> **Note:** The *alias* is specified with up to 12 characters. If you want increased security by using a set of symmetric LTO4 encryption keys across the pool of encrypted LTO4 cartridges to limit the work in case a key gets compromised, the keytool also allows the creation of multiple symmetric keys in one step by using the *aliasrange* parameter instead. An aliasrange is specified with a three-character prefix followed by lower and upper limits of up to 16 hex digits, for example, `AES00-0F`, which creates 16 symmetric key aliases at one time. Refer to the online help accessible by the **`keytool -ekmhelp`** command for additional information.

Viewing the newly created certificate in the EKM JCEKS keystore

List the contents of the keystore with the newly created certificates by using the command:

```
keytool -list -keystore /EKM/EKM.jck -storetype JCEKS -storepass password
```

Example 15-1 on page 528 shows the listing for a sample JCEKS keystore with two public-private key certificates labeled `tape_certificate` and `tape_certificate2` and sixteen symmetric keys labeled `aes0...0` through `aes0...f`.

Example 15-1 Sample JCEKS keystore

```
>  keytool -list -keystore /EKM/EKM.jck -storetype JCEKS -storepass TS1120
   Keystore type: JCEKS
   Keystore provider: IBMJCE
   Your keystore contains 18 entries
   aes000000000000000009, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000008, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000007, Apr 14, 2007, SecretKeyEntry,
   aes00000000000000000f, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000006, Apr 14, 2007, SecretKeyEntry,
   aes00000000000000000e, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000005, Apr 14, 2007, SecretKeyEntry,
   aes00000000000000000d, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000004, Apr 14, 2007, SecretKeyEntry,
   aes00000000000000000c, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000003, Apr 14, 2007, SecretKeyEntry,
   aes00000000000000000b, Apr 14, 2007, SecretKeyEntry,
   tape_certificate2, Apr 14, 2007, keyEntry, Certificate fingerprint (MD5):
   41:24:F7:87:7E:6F:C4:B6:DD:17:7E:76:5A:A3:C6:AB
   aes00000000000000000a, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000002, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000001, Apr 14, 2007, SecretKeyEntry,
   aes000000000000000000, Apr 14, 2007, SecretKeyEntry,
   tape_certificate, Apr 14, 2007, keyEntry, Certificate fingerprint (MD5):
   3F:6C:34:D1:2D:01:44:83:29:8F:4D:8A:2A:26:8C:F5
```

15.2.2 Configuring the TS3500 library for Library-Managed Encryption

Using the example of an IBM TS3500 Tape Library, we describe in this section the procedures for setting up the Encryption Key Manager (EKM) addresses in the tape library and enabling it for Library-Managed Encryption.

For information about configuring encryption on the IBM TS3100, TS3200, TS3310, or TS3400 tape libraries, refer to the following corresponding operator guides:

- *IBM System Storage TS3100 Tape Library and TS3200 Tape Library: Setup, Operator and Service Guide*, GA32-0545
- *IBM System Storage TS3310 Tape Library Setup and Operator Guide*, GA32-0477
- *IBM System Storage TS3400 Tape Library Planning and Operator Guide*, GC27-2107

Setting up Encryption Key Manager addresses

To set up the EKM server TCP/IP addresses used by the IBM TS3500 tape library for communication with the EKM servers:

1. Use a Web browser and enter the IBM TS3500 tape library IP address to connect to the *TS3500 Tape Library Specialist*.
2. Select **Access** → **Key Manager Addresses** and select **Create** from the drop-down menu as shown in Figure 15-20 on page 529.

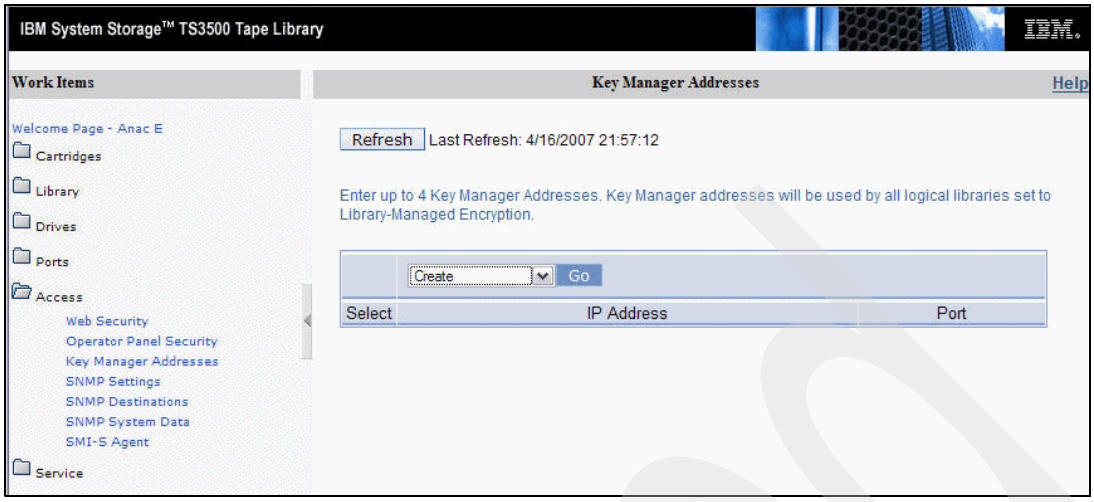


Figure 15-20 IBM TS3500 Tape Library: Key Manager Addresses window

3. Click **Go**. The panel, shown in Figure 15-21 opens.
4. Enter the IP address of the i5 server where EKM has been installed in the newly opened Create Key Manager Address window.

> **Note:** If another TCP/IP port than the default port 3801 is used, because port 3801 is already used for other TCP/IP services or multiple EKM servers will be installed on the same host system, remember to change the parameter `TransportListener.tcp.port` in your corresponding EKM configuration file `/EKM/KeyManagerConfig.properties` too.

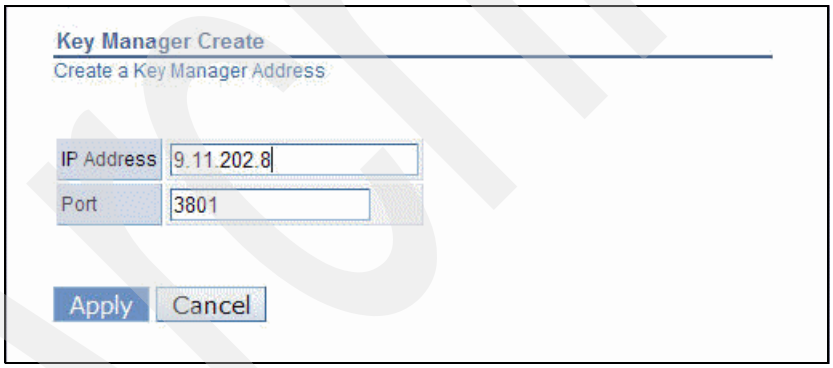


Figure 15-21 IBM TS3500 Tape Library: Create Key Manager Address window

5. Click **Apply** to proceed, then **close** the window that contains the message `The Key Manager Address Change is complete.`

   The IBM UltraScalable Specialist shows the newly created EKM address in the Key Manager Addresses window shown in Figure 15-22 on page 530.

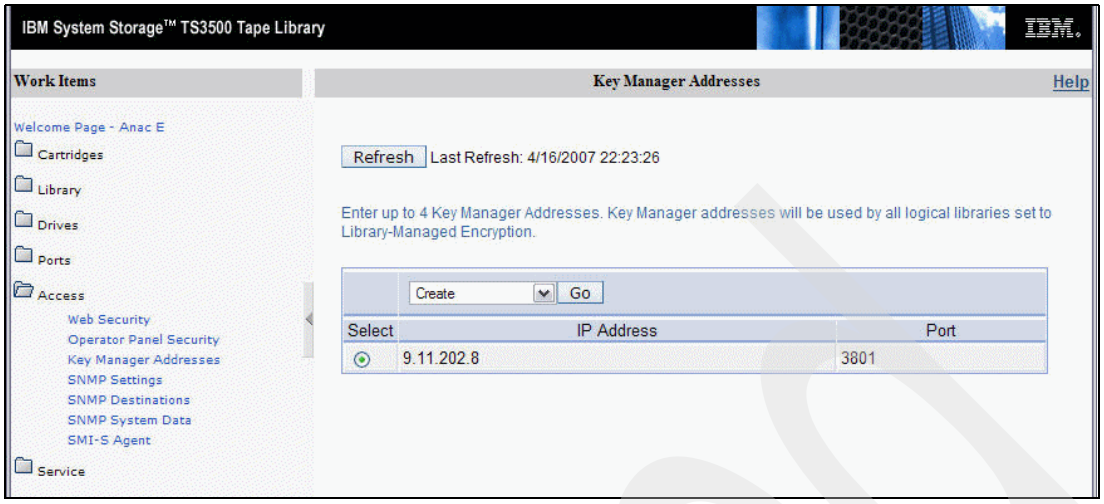


Figure 15-22 IBM TS3500 Tape Library: Key Manager Addresses window

6. Repeat steps 2 on page 528 to 5 on page 529 for any secondary EKM servers you will use.

> **Note:** Up to four EKM server addresses can be configured in the IBM TS3500 tape library for automatic EKM server failover. The library attempts to talk to the first configured EKM server, and if this fails, the library proceeds with trying to talk to the next EKM server in the list. At the end of the list, the library starts over again trying the first EKM server.

Enabling Library-Managed Encryption

Follow this procedure to configure logical libraries in the IBM TS3500 Tape Library for Library-Managed Encryption:

1. Use a Web browser to enter the IBM TS3500 tape library IP address to connect to the *TS3500 Tape Library Specialist*.
2. Select **Library** → **Logical Libraries**. Select the logical libraries from the list with the *same media type* (either TS1120 or LTO4) for which Library-Managed Encryption will be enabled, and choose **Modify Encryption Method** as shown in Figure 15-23 on page 531.

> **Notes:** Logical libraries in the list that show an Encryption Method of N/A have non-encryption-capable drives and cannot be enabled for encryption.

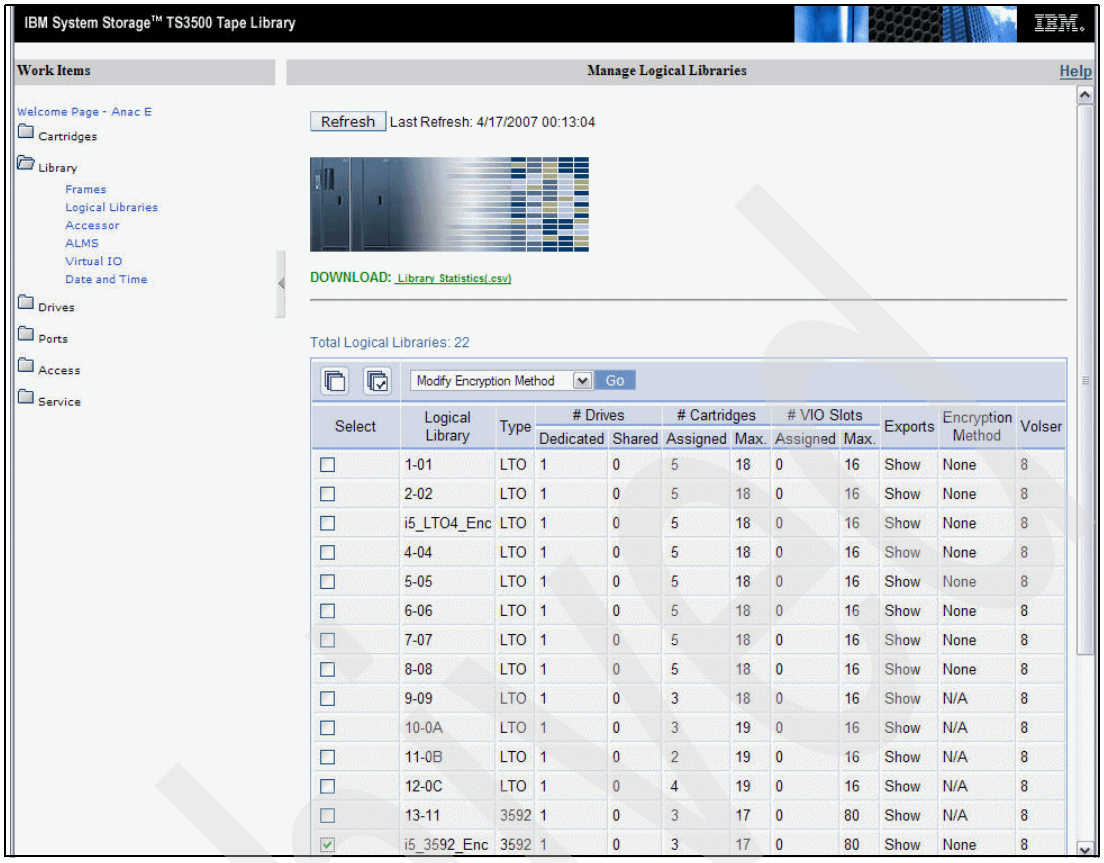


Figure 15-23 IBM Tape Library Specialist: Manage Logical Libraries window

3. After clicking **Go**, select **Library-Managed** for Encryption Method and leave the other settings at their default values as shown in Figure 15-24.

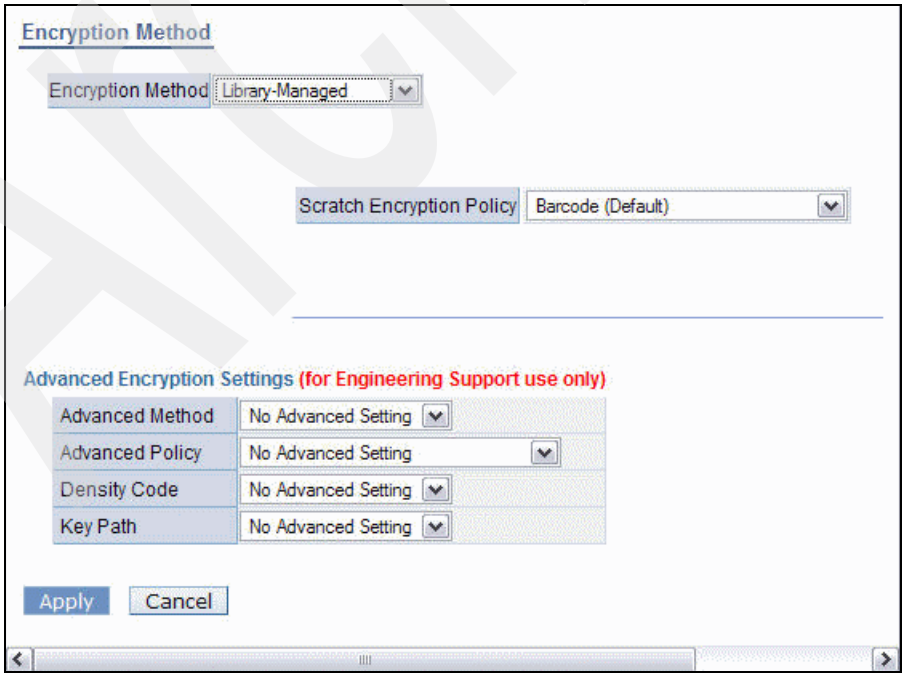


Figure 15-24 IBM UltraScalable Specialist: Encryption Method window

4. After clicking **Apply,** select **OK** for the confirmation window shown in Figure 15-25.

Figure 15-25 IBM UltraScalable Specialist: Encryption Method change confirmation window

Note: i5/OS supports no dynamic device reconfiguration. An IOP reset is required to recognize the changed encryption setting, which we account for later as the last step of this procedure.

5. A new progress window about modification of the encryption method is opened. It completes as indicated by the message `Drive Encryption change request has completed`. Select **Close** to close the Success window shown in Figure 15-26.

Figure 15-26 IBM UltraScalable Specialist: Success window

6. The automatically refreshed Manage Logical Libraries window shows the enabled *Library-Managed* encryption method for the changed logical library as shown in Figure 15-27 on page 533.

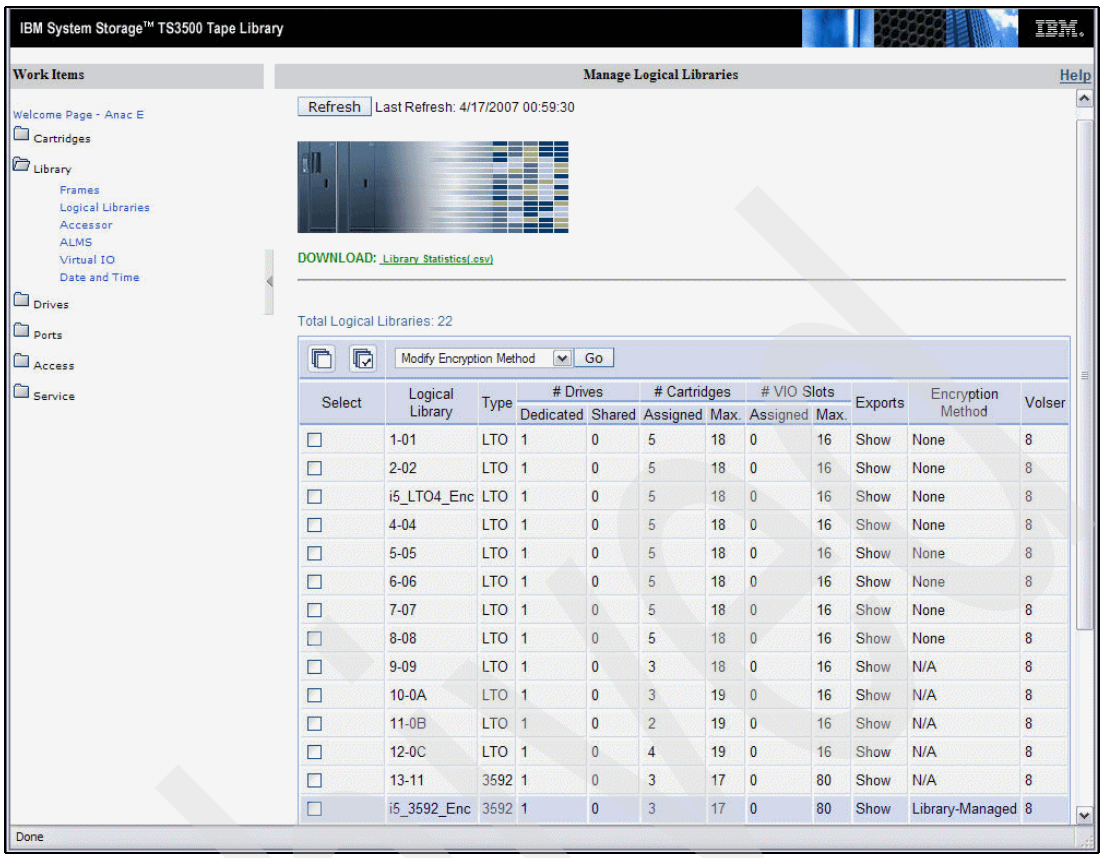


Figure 15-27 IBM UltraScalable Specialist: Manage Logical Libraries window

7. Repeat steps 2 to 6 to enable Library-Managed Encryption for other media types if required.
8. Use the following procedure on i5/OS to make it recognize the configuration change after enabling encryption with two newly added media densities, FMT3592A1E and FMT3592A2E, for encrypted 3592 cartridges:
   a. Vary off the the corresponding tape library device by running the command:

      ```
      VRYCFG CFGOBJ(TAPMLB73) CFGTYPE(*DEV) STATUS(*OFF)
      ```

   b. Obtain the *IOP resource name* of the newly attached encryption-enabled tape drive from the output of the following command:

      ```
      WRKHDWRSC *STG
      ```

      Figure 15-28 on page 534 shows an example with the `TAPMLB73` library being attached through the #2844 IOP with the resource name `CMB09`.

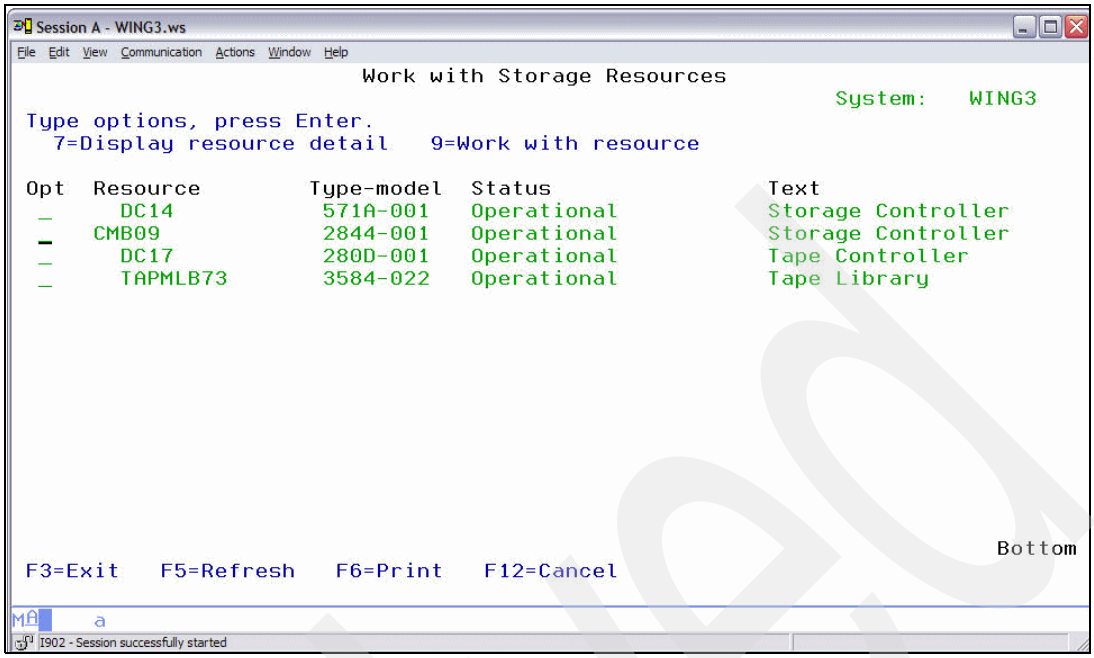


Figure 15-28 i5/OS Work with Storage Resources window

c. Access System Service Tools (SST) to reset and re-IPL the IOP associated with the attached encryption-enabled tape drive:

   `STRSST`

d. Log in to SST with your SST password and user ID and select the following menu options:

   i. **Start a service tool** from the System Service Tools (SST) window

   ii. **Select Hardware service manager** from the Start a Service Tool window

   iii. **Locate resource by resource name** from the Hardware Service Manager window.

   iv. In the Locate Resource By Resource Name window, enter your IOP resource name from step b (in the given example, it is `CMB09`).

   v. Select option **6=I/O debug** for the IOP in the Logical Hardware Resources window.

   vi. Select **3. Reset I/O processor** in the Select IOP Debug Function window and confirm the action by pressing Enter.

   vii. After receiving the completion message "`Reset of IOP was successful`", select **IPL I/O processor** from the Select IOP Debug Function window and confirm the action by pressing Enter.

   viii. After receiving the completion message `Re-IPL of IOP was successful`, press F3 three times and press Enter to exit SST.

e. Vary on the corresponding tape library device again by running the command:

   ```
   VRYCFG CFGOBJ(TAPMLB73) CFGTYPE(*DEV) STATUS(*ON)
   ```

Testing the EKM IBM TS3500 Library-Managed Encryption setup

After completing the steps in "Setting up Encryption Key Manager addresses" on page 528 and "Enabling Library-Managed Encryption" on page 530, use the procedure in this section to test the IBM TS3500 library communication with the configured EKM server addresses.

To test the setup:

1. Start the EKM Admin Consoles for all configured EKM server addresses:
   - For EKM servers installed on i5/OS, run the following command from i5/OS Qshell:

     `strEKM -propfile /EKM/KeyManagerConfig.properties`

     Note: Always use the strEKM start script on i5/OS for starting the EKM Admin Console, which helps to ensure that the correct Java version is used for EKM.

   - For EKM servers installed on other Java platforms, run the following Java command:

     `java com.ibm.keymanager.KMSAdminCmd KeyManagerConfig_full_file_path_name`

   Example 15-2 shows the EKM admin command prompt # displayed after starting the EKM Admin Console. The output from the **listdrives** EKM command shows that there are no drives yet listed in the EKM drive table.

   Example 15-2 EKM admin command prompt

```
> strEKM -propfile /EKM/KeyManagerConfig.properties
Apr 16, 2007 4:33:21 PM Thread[main,5,main] com.ibm.keymanger.config.ConfigImpl
get
FINER: ENTRY
Apr 16, 2007 4:33:21 PM Thread[main,5,main] com.ibm.keymanger.config.ConfigImpl
get ALL: debug.output = simple_file
Apr 16, 2007 4:33:21 PM Thread[main,5,main] com.ibm.keymanger.config.ConfigImpl
get
FINER: RETURN
#
> listdrives
Drive entries: 0
#
```

2. If you use the EKM configuration file `KeyManagerConfig.properties` parameter setting drive.acceptUnknownDrivesBefore = false to not automatically add new drives, you must manually add each tape drive to be used for encryption to the EKM drive table by using the EKM **adddrive** command before you can use the tape drives with the EKM server:

   `adddrive -drivename drivename -rec1 alias1 -rec2 alias2`

   Example:

   `adddrive -drivename 000123456789 -rec1 Tape_Certificate -rec2 Tape_Certificate2`

   Note: The *drivename* parameter value has to be specified as a 12-digit drive serial number (with leading zeroes). The *rec1* and *rec2* parameters apply for TS1120 tape encryption only and allow you to use an encryption policy with specified default key aliases for a selected drive serial number.

   The drive table can dynamically be changed without restarting the EKM server, however, *currently* only if the EKM server was started interactively from the EKM Admin Console.

3. Start the EKM server for all configured EKM server addresses by running the command **startekm** from the EKM Admin Console.

   Example 15-3 on page 536 shows the output from the **startEKM** command.

Example 15-3 startEKM output

```
> startekm
Loaded drive key store successfully
No symmetric keys in symmetricKeySet, LTO drives can not be supported.
Starting the Encryption Key Manager 2.0-20070328
Processing Arguments
Processing
Server is started
#
```

Note: The message "`No symmetric keys in symmetricKeySet, LTO drives cannot be supported`" is posted by EKM to inform you that there was no *symmetricKeySet* parameter in the EKM configuration file and, thus, it cannot support LTO4 encryption.

4. Test the encryption setup by using the IBM TS3500 tape library Operator Panel function **MENU** → **Service** → **Tests/Tools** → **Diagnostics** → **Test Encryption Key Path/Setup**. Press Enter and use the Up and Down arrow keys to select the drive from the list of encryption-enabled drives as shown in Figure 15-29.

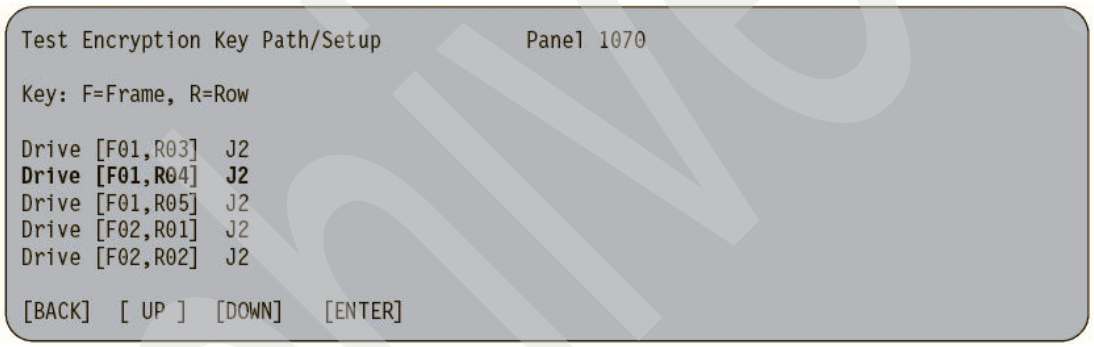


Figure 15-29 IBM TS3500 Tape Library Operator Panel Test Encryption Key Path/Setup select panel

5. Press Enter. The library performs the following tests for the encryption setup:
   - Ethernet

     The library performs a ping to the EKM's IP addresses where the selected drive is registered. It can ping up to four host server IP addresses. If successful, the panel displays the message `Passed`, which means that the target host system can be reached by the library over the network. If unsuccessful, the panel displays the message `Failed`. If at least one **`ping`** test passes, the testing continues with the *EKM Path* and *EKM Configuration* tests; otherwise, it stops and `ENTER` is displayed.
   - EKM Path

     The EKM Path test tries to establish communication to an Encryption Key Manager. It ensures that the communication path between the drive and the EKM, including the library's proxy server, is working. If successful, the panel displays the message `Passed`; if unsuccessful, it displays "`Failed`", the following *EKM Configuration* test does not run, and `ENTER` is displayed.
   - EKM Configuration

     The EKM Configuration test is a diagnostic test that establishes a link to an Encryption Key Manager and requests a default key, which ensures that the drive has been correctly installed and is able to service key requests.

Note: The library's three encryption diagnostics, that is, ping, EKM path, and EKM configuration test, *all* have to pass successfully as shown in Figure 15-30 as a prerequisite to being able to use tape encryption for the selected drive.

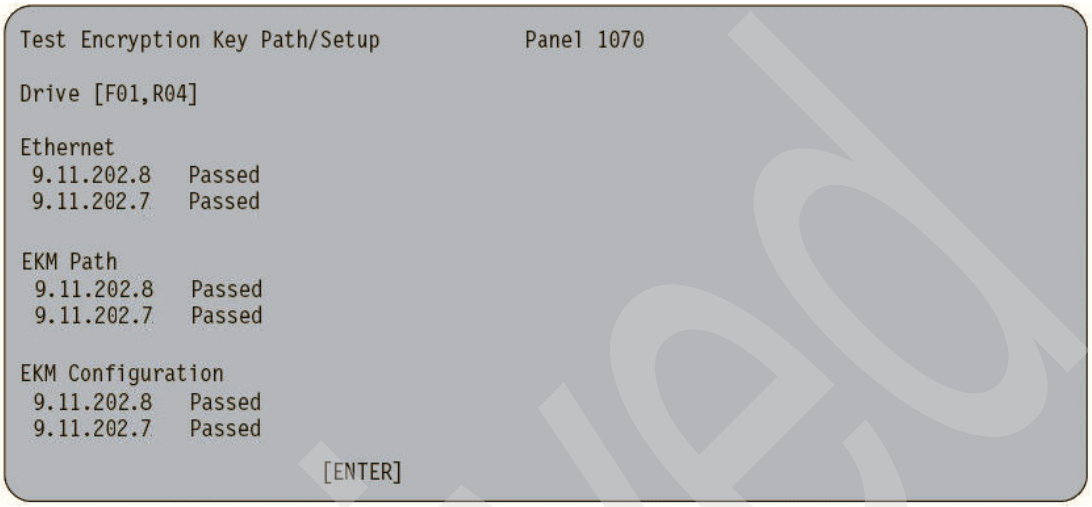


Figure 15-30 IBM TS3500 Tape Library Operator Panel Test Encryption Key Path/Setup result panel

6. Ensure *all* three diagnostic tests are passed successfully and repeat steps 4 to 5 for each encryption-enabled drive.

   A *failed Ethernet test* normally points to a network problem of the library not reaching the host server where the EKM is installed, which might be further isolated by trying to ping the EKM host server and library from another server in the network.

   For a *failed EKM Path test*, the first steps in failure isolation are to verify if the EKM server was started and is running properly by using the EKM Admin Console **`status`** command and to check that the IP port settings for the EKM servers that are defined in the library match with those IP port settings in the EKM configuration file and cause no port conflicts on the host.

   A *failed EKM configuration test* typically points to a problem with the configured key alias or keystore. For additional guidance to isolate and resolve the problem, refer to "Chapter 6. Problem Determination" of the *IBM System Storage TS3500 Tape Library Operator Guide*, GA32-0560. Contact your IBM service support representative (SSR) for additional assistance if necessary.

   Note: When using the `drive.acceptUnknownDrives = true` EKM configuration file parameter setting, new drives are automatically added to the EKM drive table after their successful completion of *EKM Path* diagnostic tests, even if the subsequent EKM configuration test fails.

7. End the interactive EKM server sessions again by entering the command **`exit`** from each EKM Admin Console. Return to "New installation of the Encryption Key Manager" on page 196 and proceed with the step to start the EKM servers as a batch job.

15.2.3 Importing and exporting encryption keys

In this section, we describe the procedures to import or export a digital certificate that is used for tape encryption from an EKM keystore. The procedures vary based on whether public-private keys, symmetric keys, or public key only certificates are considered:

- For importing and exporting public-private key certificates and symmetric keys, for example, for transfer to another EKM server on a different host platform, refer to "Procedures for public-private key certificates and symmetric keys".
- For importing and exporting a public key only certificate to provide it to a business partner in order to exchange encrypted TS1120 tape cartridges, refer to ""Procedures for public key only certificates" on page 539".

Procedures for public-private key certificates and symmetric keys

The procedures to import or export either a public-private key certificate used for TS1120 tape encryption or a symmetric key used for LTO4 tape encryption depend on the type of keystore used:

- For *IBMi5OSKeyStore*, use the i5/OS Digital Certificate Manager (DCM) **Manage Certificates** → **Export certificate** or **Import certificate** menu option, and select **Server or client certificate** to export your public-private key certificate used for TS1120 tape encryption.

  Additional information about DCM is available on the i5/OS Information Center:

  `http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp`

  Select **Site map** → **Networking** → **Networking security** → **Digital Certificate Management**.

- For *Java Cryptography Extension Keystores* (*JCEKS*), use the IBM Java *keytool* command:

  – For exporting a public-private key certificate:

  ```
  keytool -export -alias keylabel -file filename -keystore keystore -storepass
  password -storetype JCEKS -keypass password -pkcs12
  ```

  – For importing a public-private key certificate:

  ```
  keytool -import -alias keylabel -file filename -keystore keystore -storepass
  password -storetype JCEKS -keypass password -pkcs12
  ```

  – For exporting a symmetric key wrapped by a public key:

  ```
  keytool -exportseckey -alias keylabel -keyalias publickey -keystore keystore
  -storepass password -storetype JCEKS -exportfile filename
  ```

  – For importing a symmetric key wrapped by a public key:

  ```
  keytool -importseckey -alias keylabel -keyalias publickey -keystore keystore
  -storepass password -storetype JCEKS -importfile filename
  ```

  Additional information about the IBM Java keytool is available at:

  – IBM Java Keytool online help by entering **`keytool -help`** and **`keytool -ekmhelp`**

  – *IBM Java Keytool Users Guide* available online at:

  `http://www-128.ibm.com/developerworks/java/jdk/security/50/secguides/keytoolDocs/KeyToolUserGuide-150.html`

Procedures for public key only certificates

The following procedures to export and import a public key only certificate apply only in the context of providing or receiving a public key certificate to or from a business partner for sharing encrypted 3592 tape cartridges as described in 15.1.6, “Considerations for sharing tapes with partners” on page 513. These procedures depend on the type of keystore used.

Exporting a public key only from an i5OSKeyStore

For IBMi5OSKeyStore `*Other System Certificate Store` type EKM keystores, the i5/OS Digital Certificate Manager (DCM) provides no option to directly export only the public key of a digital certificate that is used for TS1120 tape encryption. However, by using the workaround, which we describe next, create an **OBJECTSIGNING certificate store* first and then export the EKM keystore certificate used for tape encryption into the *OBJECTSIGNING certificate store; the public key only can be exported from the *OBJECTSIGNING certificate store as a *signature verification certificate* without the private key.

To export:

1. Connect to the HTTP Admin Server from the i5/OS system with the EKM keystore by entering the URL `http://`*`ipaddress`*`:2001` in a Web browser.
2. Access the i5/OS Digital Certificate Manager by selecting **Digital Certificate Manager** from the i5/OS Tasks HTTP Admin Server entry panel shown in Figure 15-31.

Figure 15-31 HTTP Admin Server i5/OS Tasks entry panel

3. Select ***OBJECTSIGNING** from the Select a certificate store menu shown in Figure 15-32 on page 540.

Figure 15-32 DCM: Create New Certificate Store panel

Note: If the *OBJECTSIGNING option does not show in the DCM Create New Certificate Store panel, directly proceed to step 7, because an *OBJECTSIGNING certificate store already exists.

4. Click **Continue**. Then, select the option **No** (do not create a certificate in the certificate store) as shown in Figure 15-33.

Figure 15-33 DCM: Create a Certificate in New Certificate Store panel

5. Click **Continue**. Then, specify a password for the new *OBJECTSIGNING certificate store as shown Figure 15-34.

Figure 15-34 DCM: Certificate Store Name and Password panel

6. Click **Continue**. The confirmation message in Figure 15-35 shows the successful creation of the new *OBJECTSIGNING certificate store.

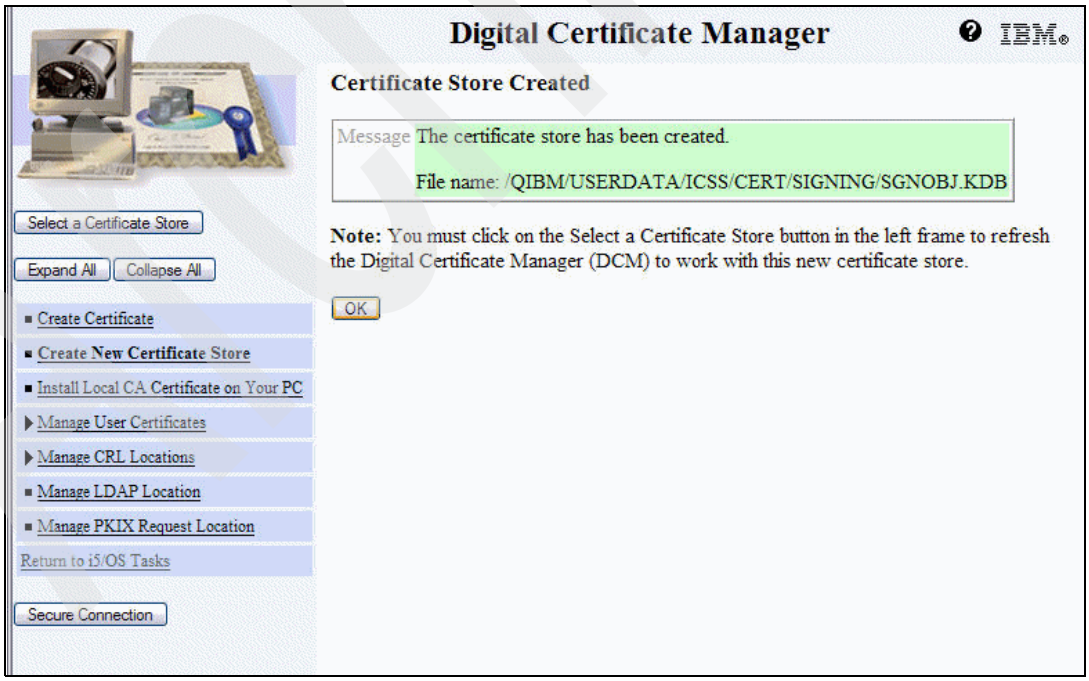


Figure 15-35 DCM: Certificate Store Created panel

7. After creating the *OBJECTSIGNING certificate store, select the EKM keystore by clicking **Select a Certificate Store** → **Other System Certificate Store**, which contains the

certificate to be exported first into the *OBJECTSIGNING certificate store. Finally, the *public key only* is exported into a file to provide to the business partner. The business partner then can use the *public key only* as a second key label for encryption of 3592 tape cartridges to shared with you, because you have the private key that is required to decrypt them.

8. Export the EKM keystore certificate into the *OBJECTSIGNING certificate store by selecting **Manage Certificates** → **Export certificate** and choosing **Server or client** as shown in Figure 15-36.

Figure 15-36 DCM: Export Certificate panel

9. After clicking **Continue**, select the certificate that is used for TS1120 tape encryption for which the public key must be exported in order to provide it to the business partner as shown in Figure 15-37 on page 543.

Figure 15-37 DCM: Export Server or Client Certificate panel

10. Click **Export**. Then, select the option **Certificate store** for the export destination to export the certificate into the previously created *OBJECTSIGNING certificate store as shown in Figure 15-38.

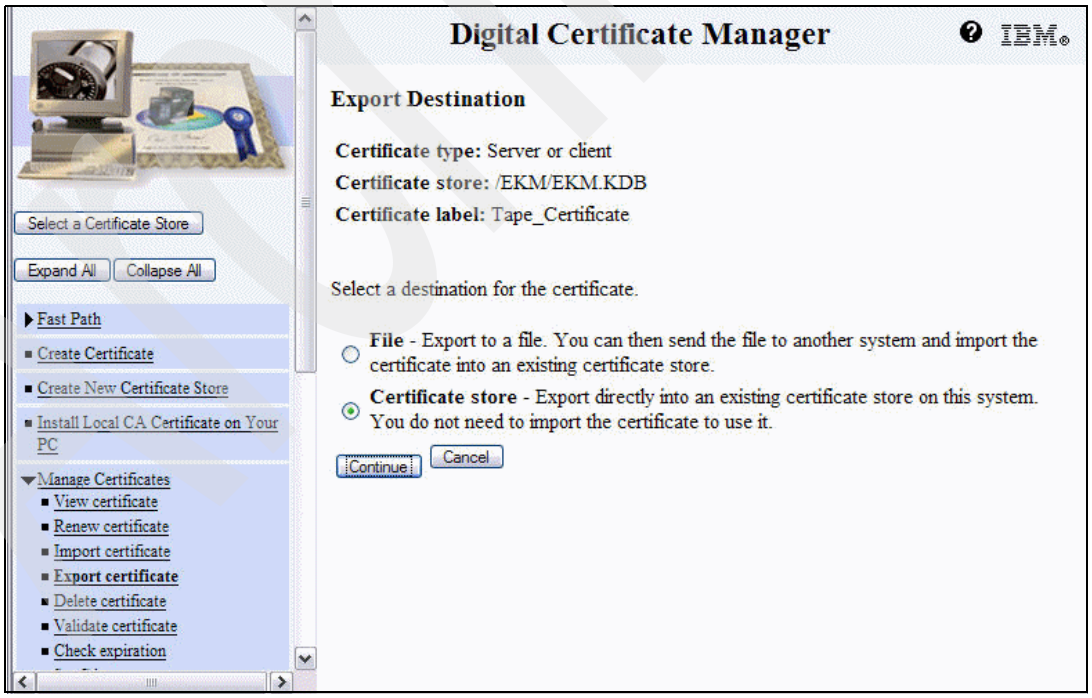


Figure 15-38 DCM: Export Destination panel

11. Click **Continue**. Then, specify the target certificate store by entering `*OBJECTSIGNING` and your password as shown in Figure 15-39 on page 544. DCM automatically replaces *OBJECTSIGNING by its full IFS file path name.

Figure 15-39 DCM: Specify Target Certificate Store panel

12. Click **Continue**. The message `The certificate has been exported to the certificate store`, which is shown in Figure 15-40, confirms the successful completion of exporting the public-private key certificate into the *OBJECTSIGNING certificate store.

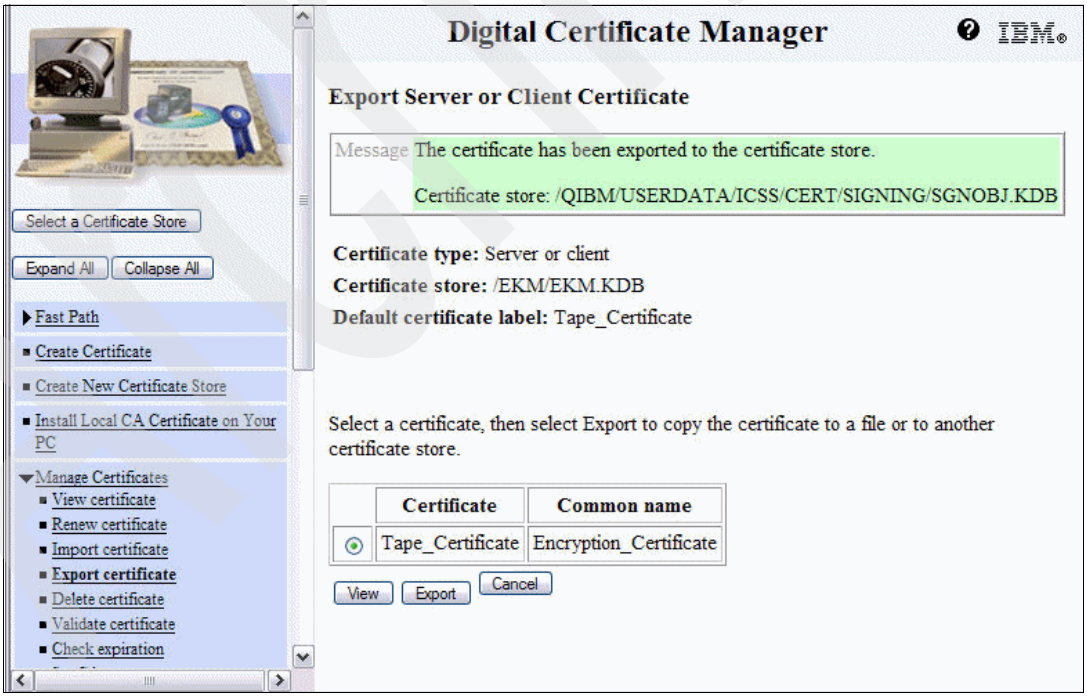


Figure 15-40 DCM: Export Server or Client Certificate panel

13. For the remaining tasks, switch to the *OBJECTSIGNING certificate store again by clicking **Select a Certificate Store** → ***OBJECTSIGNING**, clicking **Continue**, entering your password, and clicking **Continue**.

14. Export the public key only certificate by clicking **Manage Certificates** → **Export certificate** and choosing **Object signing** as shown in Figure 15-41.

Figure 15-41 DCM: Export Certificate panel

15. Click **Continue**. Then, select your certificate that is used for TS1120 tape encryption for which the public key will be exported, which is shown in Figure 15-42.

Figure 15-42 DCM: Export Object Signing Certificate panel

16. Click **Export**. Then, select **File, as a signature verification certificate** for exporting the public key only as shown in Figure 15-43.

Figure 15-43 DCM: Object Signing Certificate Export Destination panel

17. Click **Continue**. Then, specify the file name for exporting the certificate as shown in Figure 15-44.

Figure 15-44 DCM: Export Object Signing Certificate panel

18. Click **Continue**. A successful export of the public key certificate is indicated by the message `The certificate has been exported to the file` as shown in Figure 15-45.

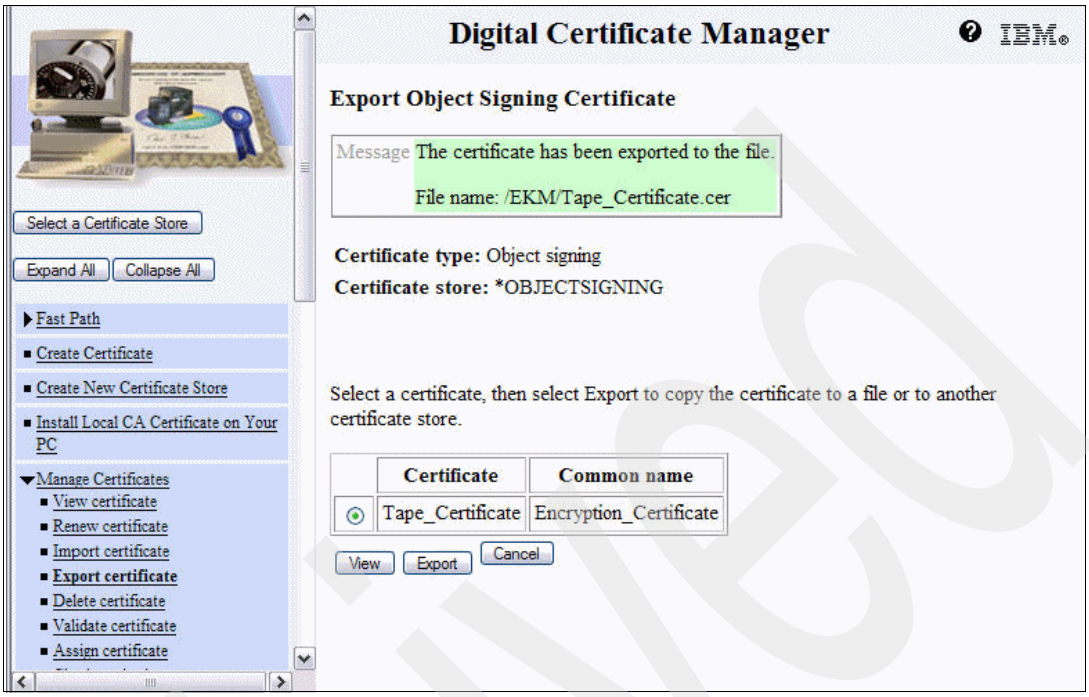


Figure 15-45 DCM: Export Object Signing Certificate confirmation panel

Note: To import this exported public key certificate on another system, use FTP ASCII mode to transfer the file, because the certificate has been exported in the *Base64_encoded* text format. Copying the file through iSeries Navigator to a personal computer destroys the certificate.

Importing a public key only to an IBMi5OSKeyStore

For importing a public key only certificate received in a *Base64_encoded* text file from a business partner to an IBMi5OSKeyStore, use the i5/OS Digital Certificate Manager's **Fast Path** → **Work with CA Certificates** → **Import** function, which is shown in Figure 15-46 on page 548, to import it into your EKM keystore.

Figure 15-46 DCM: Work with CA Certificates panel

Exporting and importing a public key only from a JCEKS keystore

For Java Cryptography Extension Keystores (JCEKS), use the IBM Java **keytool** command to export a public key only certificate:

- For *exporting* a public key only certificate:

  ```
  keytool -export -alias keylabel -file filename -keystore keystore -storepass
  password -storetype JCEKS
  ```

- For *importing* a public key only certificate:

  ```
  keytool -import -alias keylabel -file filename -keystore keystore -storepass
  password -storetype JCEKS
  ```

15.2.4 Working with encrypted tape cartridges

In this section, we show you basic examples of working with encrypted tape cartridges, such as viewing their encryption status or rekeying 3592 tape cartridges.

Viewing tape cartridge encryption status

This example shows you how to view the tape cartridge encryption status by using the TS3500 Tape Library Specialist on an IBM TS3500 Tape Library. Selecting **Cartridges** → **Data Cartridges** and using the option to select a logical library shows its assigned data cartridges as shown for a 3592 drive logical library in Figure 15-47 and an LTO4 drive logical library in Figure 15-48 on page 550.

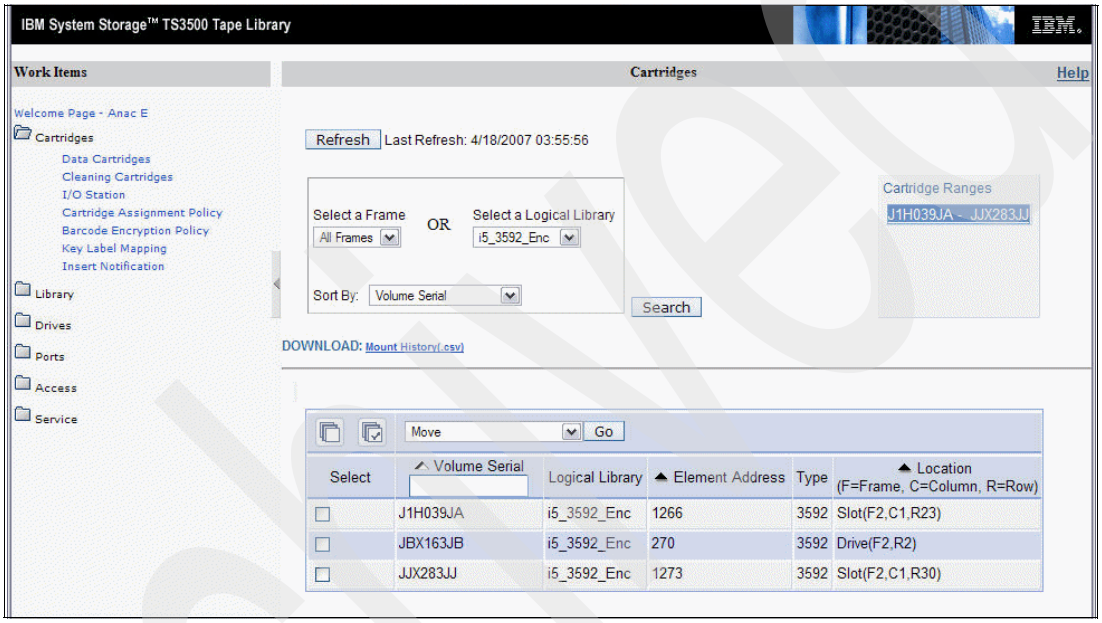


Figure 15-47 IBM UltraScalable Specialist: 3592 logical library cartridges panel

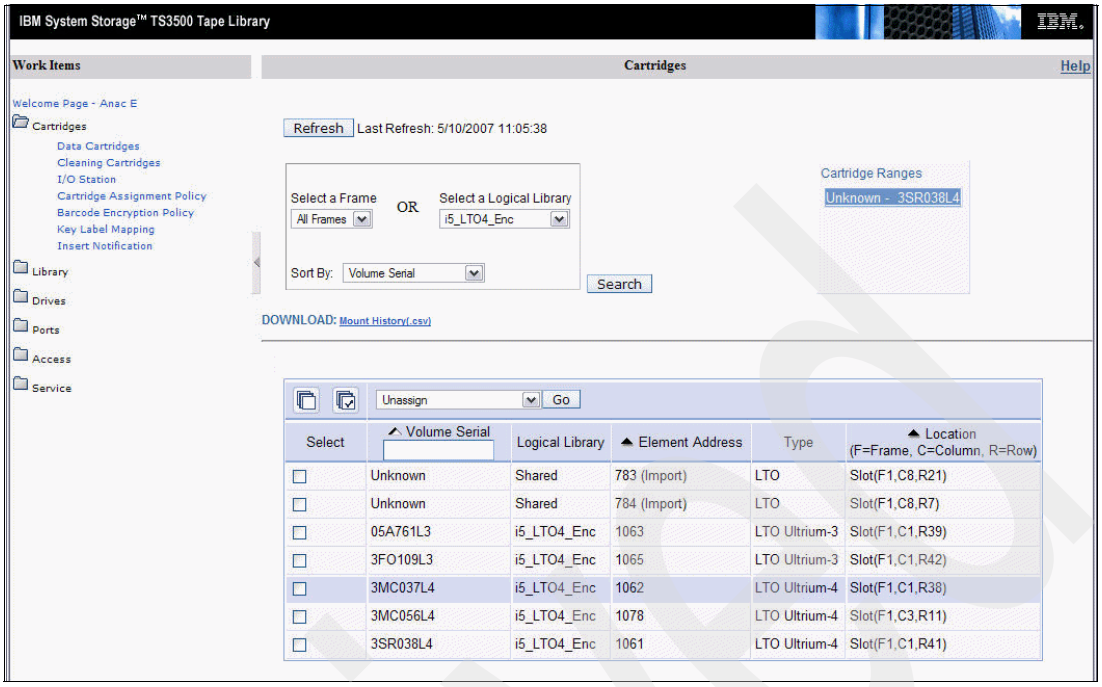


Figure 15-48 IBM UltraScalable Specialist: LTO4 logical library cartridges panel

Note: The Encryption column in the cartridges panel for an encryption-enabled logical library is not shown before the first data cartridge has been written to and unloaded from the drive.

The following commands show adding tape cartridges from the *INSERT category to the *SHARED category to make them available for usage with the TAPMLB73 3592 logical library and the TAPMLB71 LTO4 logical library:

```
ADDTAPCTG DEV(TAPMLB73) CTG(J1H039 JEX163 JJX283)
ADDTAPCTG DEV(TAPMLB71) CTG(3MC037 3MC055 3SR038)
```

The next two commands show initializing a 3592 and LTO4 data cartridges using the new media density FMT3592A2E for 3592 encryption:

```
INZTAP DEV(TAPMLB73) NEWVOL(JBX163) VOL(JBX163) CHECK(*NO) DENSITY(FMT3592A2E)
INZTAP DEV(TAPMLB71) NEWVOL(3MC037) VOL(3MC037) CHECK(*NO)
```

Note: Although two new media densities, FMT3592A1E and FMT3592A2E, are available on i5/OS for encryption-enabled 3592 tape drives, only the format FMT3592A2E is supported. FMT3592A1E was originally intended for use with the 3592-J1A Emulation mode of the 3595-E05 drive, but IBM does not support encryption in the emulation mode.

Unload the initialized cartridges from the drives by using the IBM UltraScalable Specialist Cartridges **Move** option on the IBM UltraScalable Specialist Cartridges panel. You see that the *initialized* cartridges now have the encryption status `Encrypted` as shown for the 3592 cartridge VOLSER `JBX163` in Figure 15-49 on page 551, and for the LTO4 cartridge VOLSER `3MC037` in Figure 15-50 on page 551.

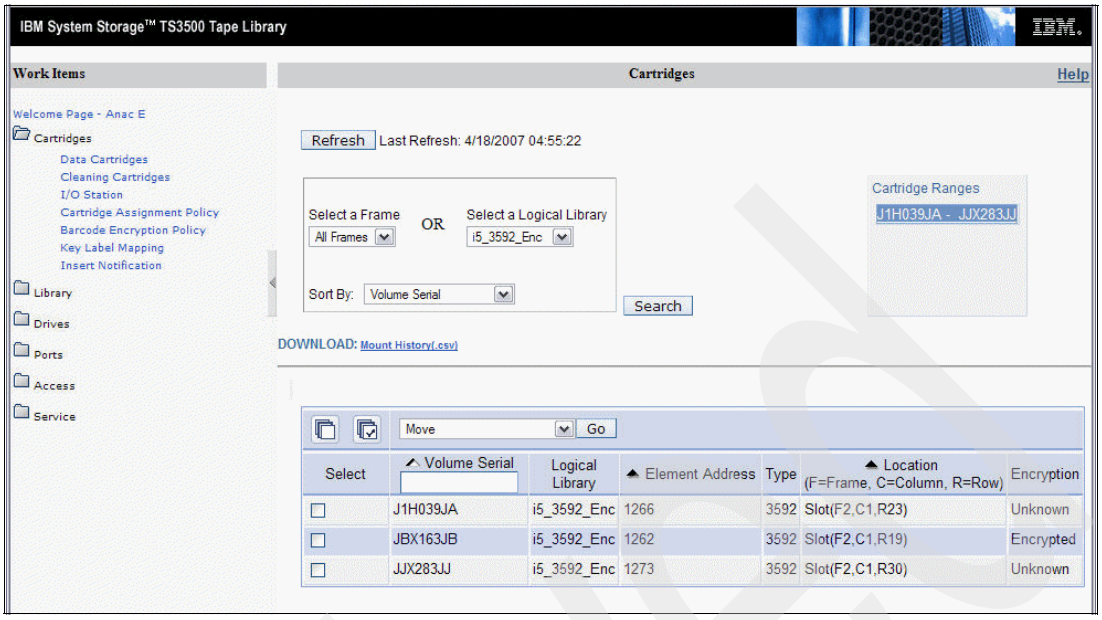


Figure 15-49 IBM UltraScalable Specialist: 3592 Cartridges panel with Encryption information

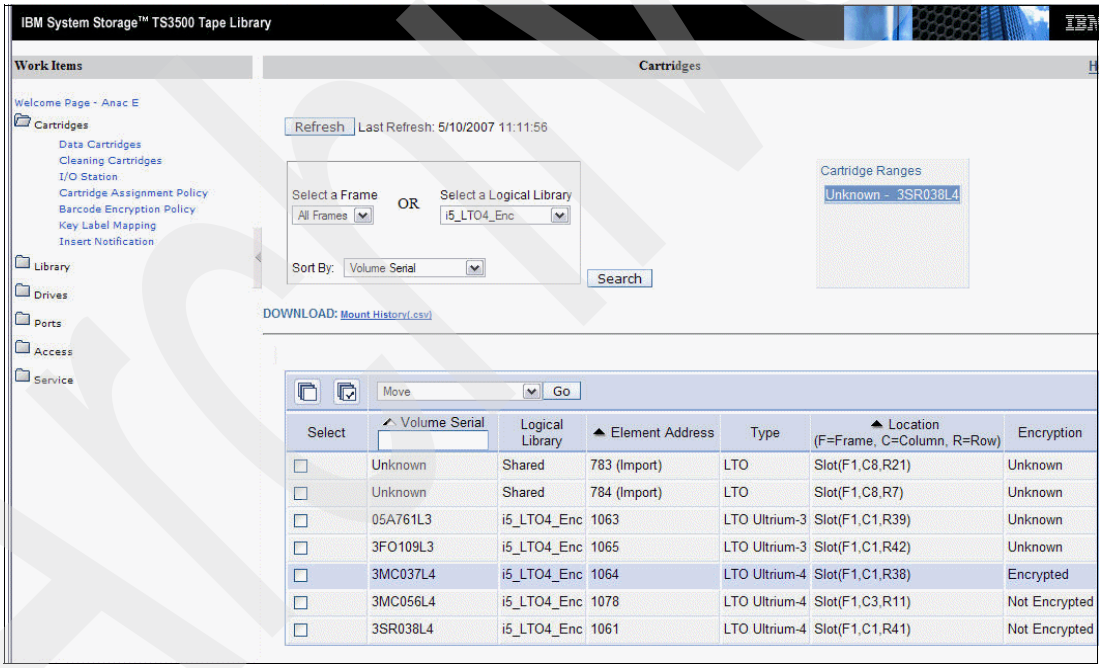


Figure 15-50 IBM TS3500 Specialist: LTO4 Cartridges panel with Encryption information

Note: The cartridge encryption status displayed in the Encryption column is updated only when the cartridge is unloaded from a drive. The status `Unknown` is displayed for cartridges in an encryption-enabled logical library as long as they have not been loaded/unloaded from a drive.

Rekeying encrypted 3592 cartridges

This example shows using the TS3500 Tape Library Specialist for rekeying encrypted 3592 tape cartridges. You can use rekeying after you have imported a public key certificate from a business partner for sharing already encrypted tape cartridges.

Encrypted 3592 tape cartridges loaded into a library-managed encryption-enabled drive are *rekeyed* through the IBM Tape Library Specialist Web GUI **Manage Cartridges** → **Data Cartridges** menu option **Rekey Encryption** shown in Figure 15-51.

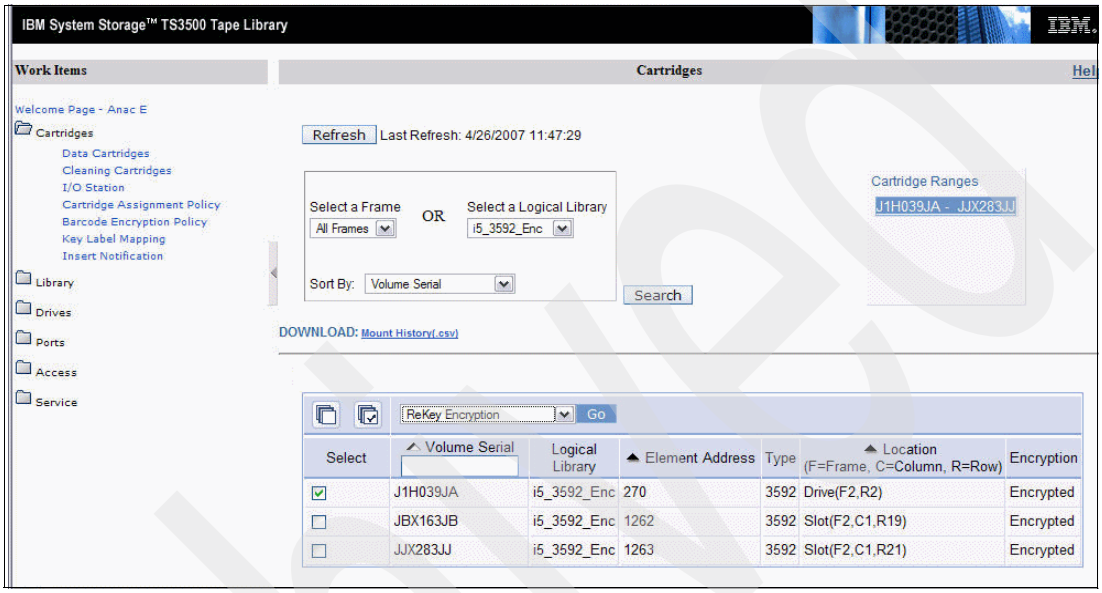


Figure 15-51 IBM TS3500 Specialist: Cartridges panel with ReKey Encryption option selected

After clicking **Go**, you may specify new key settings for the Key Mode and the Key Label for each of the two key labels that are used to refer to EEDK1 and EEDK2 shown in Figure 15-52 on page 553. In the example, a *Hash label* key mode was chosen for a new EEDK2 so that a Hash computed value of the public key part of the specified `tape_certificate2` key label, which refers to the imported public key certificate from the business partner, is stored within the EEDK2 instead of the clear label itself.

Note: Use a Hash label for sharing tape cartridges with business partners, because it eliminates the requirement of using the same key label as the business partner.

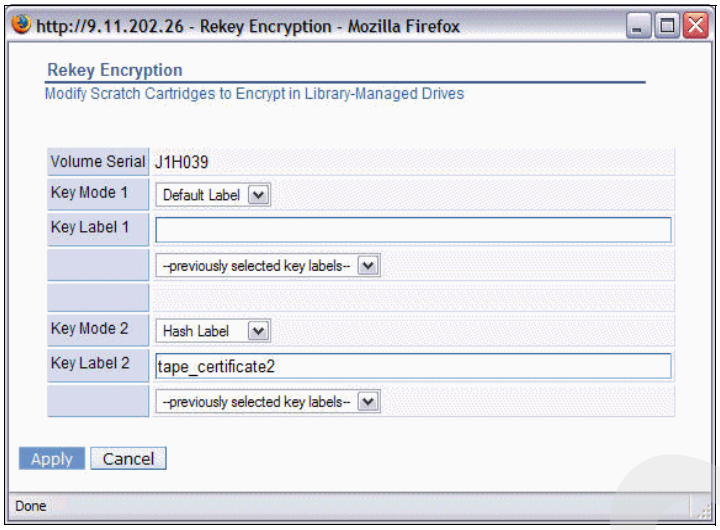


Figure 15-52 IBM UltraScalable Specialist: Rekey Encryption panel

Successful completion of the 3592 cartridge rekeying request is indicated by the message `Cartridge Rekey request has completed` shown in Figure 15-53.

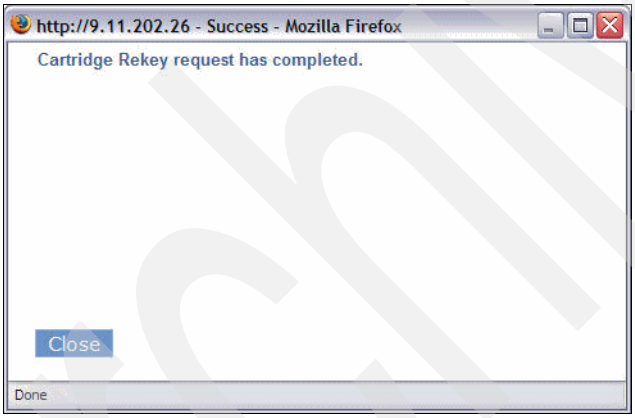


Figure 15-53 IBM UltraScalable Specialist: Rekey Encryption Success window

Example of the EKM audit metadata XML file

With EKM Release 2, an audit metadata XML log file has been introduced for logging key usage events for 3592 and LTO4 cartridge serial numbers. Refer to the section “Customizing the EKM Configuration File” on page 202 item 2 for information about the new *EKMDataParser* Java tool, which can be used for querying this file for a particular VOLSER or keyalias. Figure 15-54 on page 554 shows an example of the audit metadata XML file with key usage events logged for 3592 cartridge VOLSER `JBX163` and LTO4 cartridge VOLSER `3MC037`.

Figure 15-54 Example of EKM audit metadata XML file

15.2.5 Troubleshooting

When you encounter problems with EKM, check the EKM standard outputs, on the EKM Admin Console, especially the files `/EKM/stdout.log` and `/EKM/stderr.log`, if EKM was started as a batch job, and the EKM audit log file, which is typically the file `/EKM/auditlogs/ekm_audit.log`.

For failure isolation, refer to the “Problem Determination” chapter in *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide*, GA76-0418, at:

`http://www-1.ibm.com/support/docview.wss?rs=1139&context=STCXRGL&dc=D400&uid=ssg1S4000504`

For debugging EKM error messages that are displayed on the Admin Console or logged in the standard error outfile when starting EKM or issuing EKM Admin commands, refer specifically to the sections Debugging EKM Server problems and Messages of the Problem Determination chapter in the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide*, GA76-0418.

EKM runtime errors show up as `errorcode = Exyz` in the EKM audit log. For debugging EKM runtime errors, refer to the section, EKM-Reported Errors, of the Problem Determination chapter in the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide*, GA76-0418.

Usually, you have to collect the following data if you want additional assistance from IBM support:

- Audit log (typically, /EKM/auditlogs/ekm_audit.log)
- EKM configuration file (typically, /EKM/KeyManagerConfig.properties)
- Standard output files if EKM was running as a batch job (typically, /EKM/stdout.log and /EKM/stderr.log)
- Debug log if debugging was enabled (typically, /EKM/debug.log)

Part 5

Appendixes

This part provides checklists, tips, diagnostics, and error conditions and messages. It also discusses implementation of EKM in z/OS.

A

z/OS planning and implementation checklists

This appendix provides various checklists to use as references in planning and implementing your z/OS encryption environment. These checklists are not necessarily in the order in which you might use them.

DFSMS Systems Managed Tape planning

These are the planning steps that you have to complete prior to implementing a z/OS encryption solution using DFSMS Systems Managed Tape.

DFSMS planning and the z/OS encryption planning checklist

Both native TS1120 drives (attached to a tape control unit) and TS1120 drives attached to a TS7700 are available for encryption in IBM Tape Library environments. In a stand-alone or C20 Silo implementation, only native TS1120 drives are supported in the z/OS environment. Many of the planning steps are common to both the native TS1120 and TS7700 environments, but certain steps are unique to one environment or the other environment. We describe both environments in the following checklist:

z/OS encryption planning checklist

Perform the following steps:

1. For both environments, plan for the installation of the Encryption Key Manager (EKM) and decide which of the supported keystores to use.
2. For both environments, plan for the key labels that will be used, the key encoding mechanism (label or Hash) for each key label, and where the key labels will be specified.
3. For encryption on native TS1120 drives:
   a. Determine which z/OS systems must have TS1120 encryption coexistence support and which systems must have TS1120 encryption full support. Check the IBM preventive service planning (PSP) bucket 3592DEVICE for this maintenance.
   b. Determine when to IPL the host systems after installing the coexistence program temporary fixes (PTFs) or the full support PTFs, if required.
   c. In a stand-alone or C20 Silo implementation, order feature code (FC) 5596 or FC9596 for each of the TS1120 drives to be used by z/OS.
   d. Order FC5595 or FC9595 on the 3592 Model J70 and the TS1120 Model C06 tape controllers that you want to support encryption.
   e. Determine which data has to be encrypted and set up the appropriate Data Class policies specifying EE2 (EEFMT2). Also, specify as appropriate, the non-encryption recording formats, E1 (EFMT1) or E2 (EFMT2). If an encryption-capable IBM TS1120 Tape Drive is allocated, the default recording format is E2 (EFMT2), which is non-encrypted E05 mode. Also, determine what modifications are required for automatic class selection (ACS) routines to associate the tape output functions you plan to use with a Data Class that has the appropriate recording format specified.
   f. Determine which key labels and key encodings are to be used and how they will be specified: in the Data Class, in the DD statement, or using Encryption Key Manager-established defaults.
   g. Determine the EKM TCP/IP addresses to be used to update the I/O Supervisor (IOS) PARMLIB member (IECIOS*xx*) using the new EKM command. Also, plan to create an OMVS (open MVS) segment for the IOS address space.
   h. If you will be sharing the same 3592 media between encryption-enabled TS1120 drives and either 3592 Model J1A or non-encryption-enabled TS1120 Model E05 drives, order or download microcode upgrades for the 3592 Model J1A and the non-encryption-enabled TS1120 Model E05 drives. The microcode enables these drives to recognize and convert the EEFMT2-formatted cartridges to be relabelled and

reused as EFMT1 or EFMT2 cartridges. Also, ensure that VOLNSNS=YES is in the DEVSUP*xx* member of PARMLIB.

i. If the TS1120 drives are installed in a library, identify whether any installation exit changes are required.

4. For encryption on TS1120 drives attached to the TS7700:
   a. Determine which data has to be encrypted and set up appropriate DFSMS Storage Group and Management Class policies. Also, modify or create ACS routines to select these Storage Group and Management Class constructs for data that you want to be encrypted within a storage pool in the TS7700.
   b. Determine which TS7700 storage pools will be encrypted and which key labels and key encodings are to be used.
   c. Plan for Storage Groups and Management Classes within the TS7700 that specify these storage pools. These must match the DFSMS Storage Group and Management Class constructs.
   d. Make sure you have ordered FC9900, Encryption Configuration, for the TS7700 (3957-V06).
   e. No additional software maintenance is required to support encryption within the TS7700.
5. For both environments, determine whether you will have help from your tape management system vendor and contact the vendor, if required.
6. For both environments, with availability of the new media (MEDIA9 and MEDIA10), determine if any microcode updates on the drives are required.
7. For both environments, identify whether any installation exit changes are required.

Storage administrator stand-alone environment planning

The planning steps are:

1. Determine how to set up your tape management system's pooling support to segregate rewritable (MEDIA5, MEDIA7, and MEDIA9) and WORM (MEDIA6, MEDIA8, and MEDIA10) media and also to segregate the standard, economy, and extended length cartridges, as appropriate for their jobs and application usage.
2. Review the usage of the DEVSUP*xx* PARMLIB option, ENFORCE_DC_MEDIA, (optional) to ensure that the media type mounted is the media type requested through the Data Class. This can be used in conjunction with the tape management system's pooling support as an additional safety check.
3. Review the existing SMS Data Class media policies to ensure compatibility with existing tape scratch pool policies before enabling the DEVSUP*xx* PARMLIB option, ENFORCE_DC_MEDIA.
4. Review the existing SMS Data Class recording technology policies to ensure that data set policies set to EFMT1 are being appropriately used. If an encryption-capable IBM TS1120 Tape Drive is allocated and the specified Data Class indicates EFMT1, the drive will record in the lower recording technology.
5. Determine the Data Class updates that are required to request the appropriate recording format for the encryption-capable TS1120 tape drives. If an encryption-capable IBM TS1120 Tape Drive is allocated, the default recording format is EFMT2 (non-encryption).
6. Determine if media has to use performance segmentation, with a fast access segment to be filled first, and a slower access segment to be filled after the fast access segment. If

you decide to use the performance segmentation attribute (available with MEDIA5 and MEDIA9 tape cartridges only and mutually exclusive with performance scaling), you can:

a. Define a Data Class that requests performance segmentation.

b. Modify or create ACS routines to associate the tape output functions using performance segmentation with a Data Class that requests performance segmentation.

7. Determine whether media has to be used at full capacity or scaled for optimal performance. If you decide to use the performance scaling attribute (available with MEDIA5 and MEDIA9 tape cartridges only), you can:

   a. Define a Data Class that requests performance scaling.

   b. Modify or create ACS routines to associate the tape output functions using performance scaling with a Data Class that requests performance scaling.

8. Determine how to allocate media to appropriate non-library drives. Consider using the IBM manual tape library. You can also segregate the real drives from the emulating drives, use third-party tape management software, or use client-written applications.

9. Identify any required changes to the hardware configuration definition (HCD) to define the new devices.

Storage administrator tape library environment planning

The planning steps are:

1. Review the usage of the DEVSUP*xx* PARMLIB option, MTL_NO_DC_WORM_OK, if the WORM cartridges in the manual tape library environment will be mounted through the use of the tape management system's pooling support as opposed to a Data Class WORM media specification.

2. Determine the 3592 media usage of rewritable (MEDIA5, MEDIA7, and MEDIA9) and WORM (MEDIA6, MEDIA8, and MEDIA10) media and also the usage of the standard, economy, and extended length cartridges. Then, make the appropriate Data Class definition updates to select the appropriate media type. WORM media can only be used if explicitly requested through the Data Class.

3. Review ACS routines for changes required in selecting tape storage groups and libraries that have the new encryption-capable IBM TS1120 Tape Drives.

4. Determine the Data Class updates that are required for using the recording technology, media type, and performance scaling or performance segmentation Data Class attribute (performance scaling or segmentation available with MEDIA5 and MEDIA9 tape cartridges only).

5. Identify any required changes to the HCD to define the new devices.

6. To define the partitioning category code for MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, and MEDIA10 tape cartridges, specify the appropriate parameter of the DEVSUP*xx* PARMLIB member.

DFSMS Systems Managed Tape implementation

After completing the planning steps, these are the task required to implement this solution.

Both native TS1120 drives and TS1120 drives attached to a TS7700 are available for encryption in IBM tape library environments. Many of the planning steps are common to both environments, but certain steps are unique to one environment or the other environment.

z/OS Encryption implementation checklist

To implement this solution:

1. For both native TS1120 drives and the TS7700 environment, install the Encryption Key Manager (EKM) and set up the keystore it will use.
2. For native TS1120 drives:
   a. When systems will be sharing the encryption-enabled TS1120 drive, install either the TS1120 encryption coexistence support or full support on all systems and IPL these systems.
   b. If the TS1120 drives are in a TS3500, 3494, or TS3400 tape library, use the TS3500, 3494, or TS3400 Tape Specialist to set (enable) the TS1120 drives (attached to the 3592 Model J70 or TS1120 Model C06 tape controllers) to the System-Managed Encryption method.
   c. If the TS1120 drives are in a rack or in a C20 Silo frame, have your service support representative (SSR) install FC5596 or FC9596 on each of the TS1120 drives. This enables the System-Managed Encryption method on each of the drives.
   d. Have your SSR install FC5595 or FC9595 on the 3592 Model J70 and the TS1120 Model C06 tape controllers, which also enables encryption on all attached encryption-capable drives. This must not be performed until the previously mentioned operating system maintenance has been implemented.
   e. Create the DFSMS Data Class policies specifying EE2 (EEFMT2). Also specify, as appropriate, the non-encryption recording formats: E1 (EFMT1) or E2 (EFMT2). In addition, you must specify a media type of MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, or MEDIA10, and the performance scaling and performance segmentation attributes (performance scaling and performance segmentation are available with MEDIA5 and MEDIA9 tape cartridges only).
   f. Modify the DFSMS Data Class constructs or use DD statements (or use Encryption Key Manager defaults) to specify the key labels and their encoding mechanism (label or Hash).
   g. Modify or create ACS routines to associate the tape output functions using with a Data Class that has the appropriate recording format specified.
   h. Update the IOS PARMLIB member (IECIOS*xx*) using the new EKM command. Also, create an OMVS (open MVS) segment for the IOS address space. Creating an OMVS segment for IOS requires an IPL.
   i. Upgrade 3592 Model J1A and base TS1120 Model E05 microcode to enable the drives to recognize the EEFMT2-formatted cartridges and enable the cartridges to be relabelled and reused. Also, ensure that VOLNSNS=YES is in the DEVSUP*xx* member of PARMLIB.

j. Make any required changes to the HCD to define the new devices; generally, this is not required except when installing 3592 or TS1120 drives for the first time. No additional HCD changes are required to support encryption.

k. If required, define the partitioning category code for MEDIA5, MEDIA6, MEDIA7, MEDIA8, MEDIA9, or MEDIA10 tape cartridges, and specify the appropriate parameter of the DEVSUP*xx* PARMLIB member.

3. For TS1120 drives attached to the TS7700 Virtualization Engine:

   a. Implement appropriate DFSMS Storage Group and Management Class policies. Also, modify or create ACS routines to select these Storage Group and Management Class constructs for data that you want to be encrypted within a storage pool in the TS7700.

   b. Implement matching Storage Groups and Management Classes in the TS7700 that specify the storage pools where encryption will be used.

   c. Using the TS3500, 3494, or TS3400 Tape Specialist, set (enable) the TS1120 drives that are to be used by the TS7700 to the System-Managed Encryption method.

   d. Using the instructions provided with FC9900, install the Feature License for 9900 in the TS7700 (3957-V06). This enables encryption microcode within the TS7700.

   e. Modify storage pools within the TS7700 to specify encryption and to specify the key labels and key encoding (Label or Hash) to be used.

   f. Modify the EKM addresses within the TS7700 to specify the TCP/IP addresses for your EKM implementations.

   g. If required, define the partitioning category code for MEDIA1 or MEDIA2 virtual volumes within the TS7700, and specify the appropriate parameter of the DEVSUP*xx* PARMLIB member.

4. For both environments:

   a. Ensure that the required microcode updates for the new media (MEDIA9 and MEDIA10) have been made.

   b. Make any required installation exit changes.

   c. Define or alter existing Storage Group constructs to include libraries with the new encryption-enabled TS1120 tape drives or the encryption-enabled TS7700.

   d. Update ACS routines to direct allocations to the encryption-enabled IBM TS1120 Tape Drive or to the encryption-enabled TS7700 as requested.

   e. Validate and activate any new or modified SMS configuration.

   f. Test your implementation.

Object access method planning

The planning steps that you must consider in tape environments that use object access method (OAM) objects vary depending upon the type of environment that is installed. OAM is not a common implementation.

Perform the following steps:

1. If you install the new encryption-capable IBM TS1120 Tape Drive in a stand-alone environment:

   a. Follow the system customization planning steps listed for the DFSMS environment.

b. Determine the esoteric or generic device names that have to be added to STORAGEGROUP statements in the CBROAM*xx* member of PARMLIB for the object storage groups that are to use the new devices.

c. Determine whether to use the global keyword DSNWITHSGNAME on the SETOAM statement in the CBROAM*xx* member of PARMLIB to append the object storage group name to the OAM object tape data set names.

2. If you install the new encryption-capable IBM TS1120 Tape Drive in an IBM tape library:

   a. Follow the system customization planning considerations listed for the DFSMS environment.

   b. Determine the new Data Classes that have to be defined in STORAGEGROUP statements in the CBROAM*xx* member of PARMLIB for the object storage groups that are to use the new devices.

3. In addition, if you install the new encryption-capable IBM TS1120 Tape Drive in an OAMplex:

   a. Ensure that the new encryption-capable IBM TS1120 Tape Drives are available to all instances of OAM where the full support software is installed.

   b. Determine whether systems exist that will require coexistence support. This situation is particularly important in an OAMplex where at least one system has the full-support software installed and enabled, and at least one system will not have all of the support installed or enabled. Coexistence support is required if not all of the systems in the OAMplex will be at the same full-support level.

   c. To provide this coexistence support, as appropriate for the support and the release level, install the OAM full-support PTF without the enabling PTF or any separate coexistence support PTF.

   d. Determine when to IPL the host system after installing the coexistence PTFs, if required.

Storage administrator OAM planning

The planning steps that you must consider in tape environments that use OAM objects vary depending upon the type of environment that is installed:

1. If you install the new encryption-capable IBM TS1120 Tape Drive in a stand-alone environment, follow the storage administration planning steps listed for the Storage Administrator stand-alone environment planning.

2. If you install the new encryption-capable TS1120 Tape Drive in an IBM tape library:

   a. Follow the storage administration planning steps listed for the Storage Administrator tape library environment planning.

   b. Review ACS routines for STORE or CTRANS environments and make any changes required to ensure proper class assignment.

3. If you install the new encryption-capable TS1120 Tape Drive in an OAMplex, you must make the devices available to all instances of OAM where the full support is installed.

OAM implementation

After completing the planning steps, these are the tasks required to implement this solution.

The migration steps that you must take in tape environments that use OAM objects vary depending upon the type of environment that is installed. Perform the following tasks:

1. If you install the new encryption-capable IBM TS1120 Tape Drive in an OAMplex:
   a. Make the new encryption-capable IBM TS1120 Tape Drives available to all instances of OAM where the full support software is installed.
   b. Install coexistence PTFs as appropriate.
   c. Consider setting DSNWITHSGNAME in the SETOAM statement in the CBROAM*xx* PARMLIB member. Review your ACS routines if you are appending the storage group name to OAM data set names (DSNWITHSGNAME).
   d. IPL the system.
2. If you install the new encryption-capable IBM TS1120 Tape Drives in an IBM tape library:
   a. Follow the migration steps listed for DFSMS Implementation.
   b. Define the new Data Classes in STORAGEGROUP statements in the CBROAM*xx* member of PARMLIB for the object storage groups that are to use the new devices.
   c. Make the required changes to ACS routines for ALLOC, STORE, or CTRANS environments.
3. If you install the new encryption-capable IBM TS1120 Tape Drives in a stand-alone environment:
   a. Follow the migration steps listed for DFSMS Implementation.
   b. Add new device esoteric unit names or generic unit names to STORAGEGROUP statements in the CBROAM*xx* member of PARMLIB for the object storage groups that are to use the new devices. The esoteric or generic unit name must consist of encryption-capable IBM TS1120 Tape Drives exclusively, because the EFMT2 recording technology is incompatible with other recording technologies.
   c. Consider setting DSNWITHSGNAME in the SETOAM statement in the CBROAM*xx* PARMLIB member. Review your ACS routines if you are appending the storage group name to OAM data set names (DSNWITHSGNAME).
   d. Make the required changes to ACS routines for the ALLOC, STORE, and CTRANS environments.

DFSMShsm tape environment

DFSMShsm allows the specification of tape unit names using either generic or esoteric names. In installations that have a mixture of non-SMS-managed 3590 or 3592 devices that are defined under the 3590-1 generic name, perform the following steps:

1. Define a unique esoteric name for each recording technology.
2. Use the SETSYS USERUNITTABLE command to define these esoteric names to DFSMShsm. This also applies to mixed devices in the 3490 generic device name. Installations, which use SMS-managed tape devices or have a single 3590-1 recording technology, do not have to define an esoteric name for those devices. However, if you have a mixed SMS-managed 3590 environment, review APAR OW57282.

B

z/OS Java and Open Edition tips

In this appendix, we provide helpful tips for UNIX and UNIX System Services environments to help you set up EKMs and Java. We include an introduction to Java.

JZOS

The JZOS job launcher and runtime APIs are the latest inclusion in the IBM Java SDK distribution for z/OS specifically tailored for submitting Java applications as a job.[1] You may obtain a more detailed discussion about JZOS in *Java Stand-alone Applications on z/OS Volume II,* SG24-7291.

Obtain the *JZOS Batch Launcher and Toolkit function in IBM SDK for z/OS Installation and User's Guide*:

http://www.ibm.com/servers/eserver/zseries/software/java/pdf/jzosinstal.pdf

JZOS includes Java classes that make the console communication from Java applications easy. Instead of implementing the console communication using Java Native Interface (JNI™) calls, JZOS provides a framework to register a *listener* for interaction with operators or write to operator (WTO). It also provides a method to write messages to MVS system logs directly from Java applications. Using this framework, Java programs are able to write messages to the MVS system log when they require special attention from the operator. This can be used as a means to interface with the system automation tools to monitor the status of the running Java batch programs. Receiving commands from the operator while Java batch jobs are running (accepting the *modify* command) is also possible. Consequently, it is possible to program Java programs to change behaviors depending on the commands received from the operator without restarting the job. Refer to *Java Stand-alone Applications on z/OS Volume II,* SG24-7291, for more information about JZOS.

The JZOS batch launcher directs *stdout* and *stderr* input streams to the standard MVS data sets and JESS SYSOUT data set. Also, Java programs are able to read from *stdin* from the standard MVS data sets. With BPXBATCH, it is only possible to write to and read files in UNIX System Services. Consequently, operators are able to monitor the output from Java programs using *System Display and Search Facility (SDSF)* just like monitoring any other job steps in the MVS environments.

JZOS supports the following functions:

- Runs in the same address space as the submitted job
- Supports DD statements
- Redirects stdin, stdout, and stderr to an MVS data set
- Supports console communication
- Supports return codes using System.exit

Console communication with batch jobs

A common practice is to use the write-to-operator (WTO) macro by running programs (jobs) to write messages to the job log and system console. The macro is typically used to report the status of the program. For instance, it is used to record events when failure occurs in a long running job. Based on the messages, the operator can decide to take appropriate actions to resolve the problems. Under such circumstances or when the operator has to interact with the job, a common practice is to use the MVS stop (P) command to stop the running job or use the MVS modify (F) command to send arbitrary commands to change the behavior of the job. Running jobs are programmed to accept and respond to these commands. JZOS provides APIs to add these capabilities to Java applications. In this section, we present how to interact with jobs using MVS console and how to use these APIs in Java applications.

[1] IBM acquired JZOS batch technology from Dovetailed Technologies, LLC, in 2006. At the time of writing this book, JZOS is available from alphaWorks® at no charge. JZOS is part of standard distribution of IBM Java SDK 5.0 for z/OS.

EKM and JZOS

A UNIX System Services process in a foreground shell environment is incapable of providing RAS (reliability, availability, and serviceability) as a task that is started under MVS. A UNIX System Services process is subject to the environment under which the user started it.

In the foreground, a started-EKM is tied to a specific terminal where the user started the process. If the terminal goes down or the user who was logged in to start that process exits, the process is killed, and the EKM is stopped. Running the EKM in the foreground provides an interaction with the EKM that BPXBATCH removes. The JZOS process started with the console wrapper application com.ibm.jzosekm.EKMConsoleWrapper provides the interface between the z/OS operator's console and the EKM. Therefore, the need for a shell process is eliminated, the EKM can run as a started task with all the security controls that environment entails, and messages come to the console and can be acted upon by automation processes.

MVS modify command

To interact with a Java application that was started using JZOS, the MVS modify command is used. The syntax of the MVS modify command is:

`/f jobName,commands`

This command sends the specified commands to the specified job. A job receiving the commands has to be programmed to receive and react to the commands.

Note: The JAR file `jzosekm.jar` provides the callback functionality to communicate with the EKM when it is started using JZOS.

Non-SMP/e JZOS installation

These instructions assume that the non-SMP/E pax file has been uploaded and extracted according to the Java SDK product installation instructions. Follow these steps:

1. Allocate any required MVS PDSE or PDS data sets. For your information, the SMP/E installation, by default, places a load module in `SYS1.SIEALNKE`, a sample PROC in `SYS1.PROCLIB`, and sample JCL in `SYS1.SAMPLIB`.

   For installation into private data sets, the suggested allocation parameters for SAMPJCL and for SAMPPROC are:

   ```
   RECFM=F/FB,LRECL=80,SPACE=(TRK,(5,1))
   ```

2. Copy the load modules to a PDSE and copy the procedures and JCL to a PDS from the extracted (unpaxed) file. The load module is in <*JAVA_HOME*>`/mvstools`. The sample JCL and PROC is in <*JAVA_HOME*>`/mvstools/samples/jcl`. For example, for the 1.4 SDK and using the default target libraries, issue the following commands under a UNIX System Services shell:

   ```
   cp  JVMJCL14  "//'SYS1.SAMPLIB(JVMJCL14)'"
   cp  JVMPRC14  "//'SYS1.PROCLIB(JVMPRC14)'"
   cp  -X  JVMLDM14  "//'SYS1.SIEALNKE(JVMLDM14)'"
   ```

3. Change the sample JCL and PROC as appropriate for your environment. Specifically, update JCL with JOB card information, update the JCL and PROC with HLQ where the PROC and LOADLIB exist, and update the PROC with the location of JAVA_HOME. Make sure that your sample JCL or PROC includes a STEPLIB to the load library unless that load-library is included in your LNKLST member. In Example B-1 on page 568, we have modified the procedure to point to where we copied the loadlib. In this case, it is at:

   ```
   JBARNEY.PRIVATE.JZOS.LIB
   ```

Example: B-1 Sample modified procedure

```
//*********************************************************************
//*
//* Stored procedure for executing the JZOS Java Batch Launcher
//*
//* Tailor the procedure for your installation:
//* 1.) Replace '<HLQ>.JZOS.LOADLIB' with the PDSE that contains the
//*     JZOSVMxx modules that were installed during installation
//*
//*********************************************************************
//EXJZOSVM PROC JAVACLS=,                < Fully Qfied Java class..RQD
//   ARGS=,                              < Args to Java class
//   LIBRARY='JBARNEY.PRIVATE.JZOS.LIB',       < STEPLIB FOR JZOSVM module
//   VERSION='14',                       < JZOSVM version: 14, 50, 56
//   LOGLVL='',                          < Debug LVL: +I(info) +T(trc)
//   REGSIZE='0M',                       < EXECUTION REGION SIZE
//   LEPARM=''
//JAVAJVM  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 DDs can/should be present in the calling JCL
//*
//*STDIN   DD                   < OPTIONAL - Java System.in
//*STDENV  DD                   < REQUIRED - JVM Environment script
//*MAINARGS DD                  < OPTIONAL - Alternate method to supply args
// PEND
```

Example B-2 is a sample JCL that we copied into JBARNEY.PRIVATE.SAMPJCL, which also contains our procedure from Example B-1. This sample JCL calls the procedure with the setup for our Java environment and execute the Java class *Hello*. This class must be included in the class path setup in this JCL example.

Example: B-2 Sample JCL

```
//JBARNEY0 JOB ()
//PROCLIB JCLLIB ORDER=JBARNEY.PRIVATE.SAMPJCL
//*
//*********************************************************************
//*
//* Batch job to run the Java1.4 VM
//*
//* Tailor the proc and job for your installation:
//* 1.) Modify the Job card per your installation's requirements
//* 2.) Modify the PROCLIB card to point to this PDS
//* 3.) edit JAVA_HOME to point the location of the 1.4 JDK
//* 4.) Modify the CLASSPATH as required to point to your Java code
//* 5.) Modify JAVACLS and ARGS to launch desired Java class
//*
//*********************************************************************
//*
//*
//JAVA EXEC PROC=JZOSPRC,
```

```
// JAVACLS='Hello'
//STDENV DD *
# 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=/usr/lpp/java/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=/usr/lpp/java/J1.4/lib/ext

# Add required jars to end of CLASSPATH
for i in "${CLASSPATH}"/*.jar; do
    CLASSPATH="$CLASSPATH":"$i"
    done
export CLASSPATH="$CLASSPATH":

# Set JZOS specific options
# Use this variable to specify encoding for DD STDOUT and STDERR
# export JZOS_OUTPUT_ENCODING=Cp1047
# Use this variable to prevent JZOS from handling MVS operator commands
# export JZOS_ENABLE_MVS_COMMANDS=false
# Use this variable to supply additional arguments to main

# Configure JVM options
IJO="-Xms16m -Xmx128m"
# Uncomment the following to aid in debugging "Class Not Found" problems
 IJO="$IJO -verbose:class"
# Uncomment the following if you want to run with Ascii file encoding..
#IJO="$IJO -Dfile.encoding=ISO8859-1"
export IBM_JAVA_OPTIONS="$IJO "

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

4. SUBMIT the modified JCL and check the job log. If everything was set up properly, the SYSOUT DD contains similar to Example B-3.

Example: B-3 Sample JZOS output

```
JVMJZBL1001N  JZOS  batch  Launcher  Version:  2.0.0  2006-06-30
JVMJZBL1002N  Copyright  (C)  IBM  Corp.  2005.  All  rights  reserved.
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  cm142-20060803  (JIT  enabled:  jitc))
JVMJZBL1023N  Invoking  Hello.main()...
JVMJZBL1024N  Hello.main()  completed.
JVMJZBL1021N  JZOS  batch  launcher  completed,  return  code=0
Hello  World
```

To diagnose problems with the JZOS batch launcher, change the LOGLVL parameter to:

```
'+I':
```

For example:

```
//  EXEC  JVMLDM14,LOGLVL='+I',
```

Note: Setting this logging level (+I) dumps the environment that is passed to the JVM. The trace level setting '+T' produces many messages, some of which might be helpful in tracking down installation problems.

MVS Open Edition tips

The OMVS shell entered through TSO is not the most UNIX-like shell environment. Because of differences in the tn3270 protocol and a VT100 terminal, you can only use the ISPF editor when invoking OMVS through TSO by using the tn3270 protocol. In addition, the OMVS command line is limited to approximately 125 characters.

Several workarounds are available to the 125-character limit on the command line, including:

- Exporting long arguments to environment variables
- Setting up an alias
- Using relative paths instead of fully qualified paths for arguments
- Copying the escape character, the cent sign (¢), to the end of a line, pressing Enter, and continuing the command on a new line

Exporting a variable

To export a variable, enter a command similar to:

```
export shortName = someLongCommandArgument
```

Then, when you want to use the argument `someLongCommandArgument`, enter:

```
$shortName
```

A common practice is to set up a `.profile` file in your home directory with a list of commonly used environment variables that will be read each time that you log in.

Setting up an alias

An *alias* is a UNIX shorthand for a command that you commonly use. Aliases are typically set up in the `.profile` file. One of the most common aliases is:

```
alias ll = "ls -l"
```

When "`ll`" is entered at the command line, it performs the same as the `ls -l`" command.

It can also take all the additional `ls` arguments. A useful alias for dealing with EKM is:

```
alias startEKM = " java com.ibm.keymanager.KMSAdminCmd"
```

Now, enter the following command to start the EKM:

```
startEKM KeyManagerConfig.properties
```

Copying the escape character

Figure B-1 is an example of spanning new lines using the cent (¢) escape character.

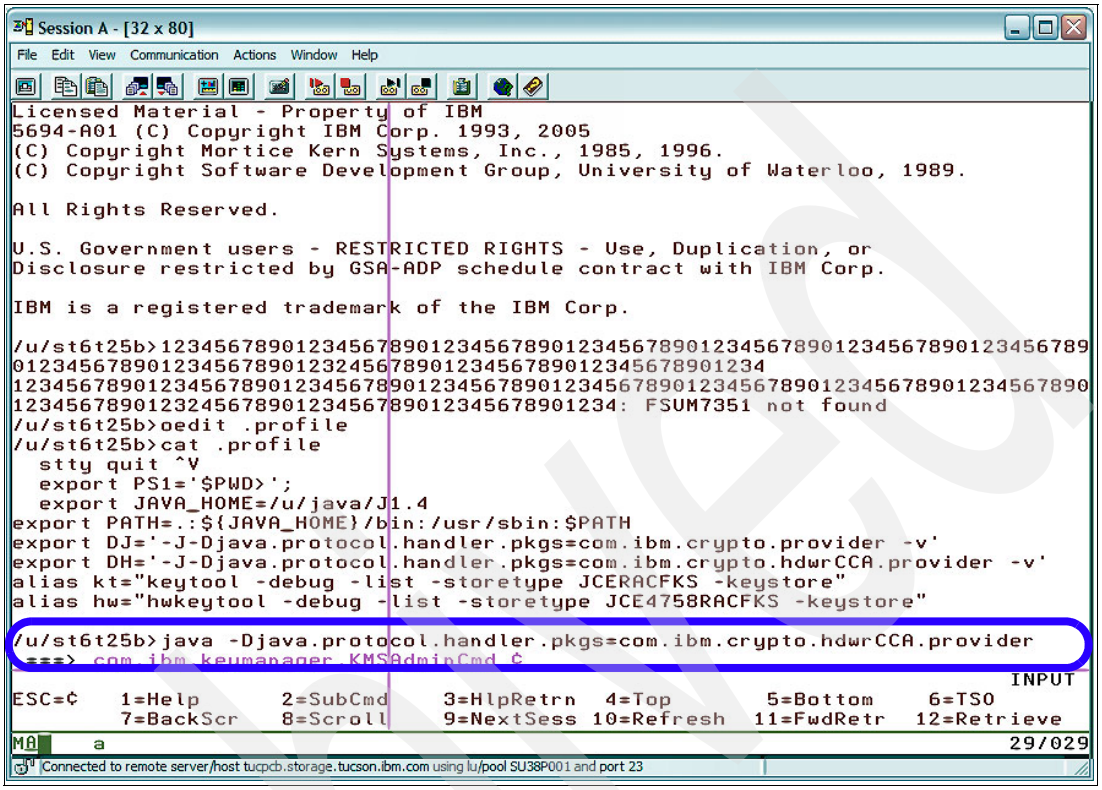


Figure B-1 Using the escape character ¢

In the example shown in Figure B-1:

1. On a single line, we entered all of the following information:
   - The **java** command
   - The **-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider** system argument
   - The ¢ character at the end of the line
2. We pressed Enter.
3. We added the **com.ibm.keymanager.KMSAdminCmd** class argument with another **¢**.

After we press Enter, we simply add the `KeyManagerConfig.properties` file, and EKM starts.

Advantages of VT100

It is possible to set up a Telnet daemon, rlogin daemon, or an SSH daemon to give alternative access to OMVS. Refer to *Communications Server for z/OS V1R2 TCP/IP Implementation Guide*, SG24-5227, for setting up Telnet and rlogin using **inetd**. Refer to *IBM Ported Tools for z/OS User's Guide*, SA22-7985, for details about setting up SSH.

If any of these three daemons are running on a z/OS system, using a VT100 terminal emulator to log in to the system becomes possible. Then, changing our shell from `/bin/sh` to `/usr/sbin/bash` is useful. More information about ported tools and bash are at:

http://wwwibm.com/servers/eserver/zseries/zOS/unix/port_tools.html

Replacing bash with the sh shell and using a VT100 terminal give us the following features:

- Tab completion

  When entering the name of a command, file, or directory, pressing the Tab key autocompletes them if you typed enough of the name to make it unique. If you did not enter enough of the name to make a unique match, pressing Tab twice gives you a list of possible matches.

- Ctrl + r

  Pressing `Ctrl + r` puts the command line into a reverse search of previous commands. Type a portion of a previously entered command and the shell tries to choose the closest match.

- Up and down arrow keys

  The up and down arrow keys navigate the history list of commands.

- Command line length

  Using a VT100 terminal lets you type in a virtually endless length command on the command line.

- History command

  Typing **`history`** echoes the complete command list, prefixed with a number, to the window. Entering **`! commandNumber`** executes that command.

- Directory stack

  When navigating long directory paths, if you use the **`pushd .`** command, the current working directory path is pushed onto a stack. Typing **`popd`** pops the directory path off the stack and returns you to that working directory.

Security alert: SSH is the most secure of the three daemons listed here. Connection to OMVS using SSH requires an *SSH client.* SSH is not included in Windows. For this book, we used SecureCRT on Windows. Many free SSH clients are available for other UNIX and Linux systems, including OpenSSH.

SSH encrypts all traffic between the SSH client and the SSH daemon.

Windows includes a Telnet client.

There are also free ssh clients for Linux.

When using an SSH client, the command **`stty quit ^V`** must be included in your `.profile` file. In Example B-4 on page 573, there is a sample `.bashrc` file. We suggest that when you log in with a VT100 terminal that you include a simple `.profile` in your user home directory, and at the end of the `.profile` file, execute the bash command. If the path to bash is lost, or you do not have a path to bash setup, and you end up logging in from TSO, entering OMVS fails, because the shell cannot be found. If a `.profile` is calling bash, and if `bash` is lost, you simply receive an error, and continue. When `bash` is executed, the `.bashrc` file is read, which is similar to when you log in to OMVS and the `.profile` gets read.

The `export` PS1 sets up this user's command line. In this case, we are setting the command line to always list the current working directory. This is a useful setting to have when navigating around an OMVS file system.

After that, we set up the JAVA_HOME environment variable. This environment variable has to point to the Java home on your system. Next, we add JAVA_HOME/bin to our path, in

addition to several other useful directories. Notice that we are careful to keep our original path here by appending it to our new PATH. After that, we create environmental shorthand.

Example: B-4 Simple .profile

```
stty quit ^V
  export PS1='$PWD>';
  export JAVA_HOME=/u/java/J1.4
export PATH=.:${JAVA_HOME}/bin:/usr/sbin:$PATH
export DJ='-J-Djava.protocol.handler.pkgs=com.ibm.crypto.provider -v'
export DH='-J-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider -v'
export JS='-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider'
alias kt="keytool -debug -list -storetype JCERACFKS -keystore"
alias hw="hwkeytool -debug -list -storetype JCE4758RACFKS -keystore"
export keyFile=KeyManagerConfig.properties
export EKM=com.ibm.keymanager.KMSAdminCmd
```

Advanced security hwkeytool and keytool scripts

In this section, we describe how to create scripts to aid in the generation of JCEKS and JCECCAKS keystores for use with EKM. Unlike previous sections, here we present complete scripts to make it easier to cut and past into a script file. We include showing you how to prevent passwords from being saved in the script file. The techniques outlined in this section can also be applied to preventing passwords from showing up in shell history, because they never appear in the window.

Complete keytool example for JCEKS using hidden passwords

The script in Example B-5 on page 574 creates a JCEKS keystore with two self-signed certificates, each with a public and private key pair. The script then exports the first certificate created and reimports that certificate with a new alias as a trusted certificate entry. This entry does not have a private key associated with it. Certificates are:

cert1	Certificate alias and public and private keys associated with this certificate
cert2	Certificate alias and public and private keys associated with this certificate
cert1ca	Trusted certificate entry with only a public key associated with this certificate
rsa2048Cert1.cer	DER-encoded file that contains exported public key that matches the public key of cert1

In this script, the first line tells the operating system that we are using the bash shell to execute the commands in the script. The first line is:

```
#!/bin/bash
```

This script is generic enough that it can be executed using `/bin/sh`.

After we set up the shell with which to execute the script, we then use the **`echo`** command to prompt for the correct passwords.

The following command prevents what we type, in this case, a password, from being printed to the command line:

```
stty -echo
```

We then read what is typed into the command line to a variable and plug the variable into our keytool commands.

Example: B-5 Keytool hidden password script

```
#!/bin/bash
echo -n "Please Enter the Keystore password"
echo ""
stty -echo
read STOREPASS
stty echo
echo ""          #force a carriage return to be output
echo -n "Please Enter the private key password"
echo ""
stty -echo
read KEYPASS
stty echo
echo ""          #force a carriage return to be output


echo -n "start key generation"
keytool  -genkey  \
-alias   cert1 \
-dname  CN=IBM  \
-keystore  testkeys.jcks  \
-provider  IBMJCE  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  $KEYPASS  \
-storepass  $STOREPASS \
-storetype  JCEKS \
-validity  999

echo "Finish genkey for alias = cert1"

keytool  -genkey  \
-alias  cert2  \
-dname  CN=IBM  \
-keystore  testkeys.jcks  \
-provider  IBMJCE  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  $KEYPASS \
-storepass $STOREPASS \
-storetype  JCEKS \
-validity  999

echo "Finish genkey for alias = cert2"

keytool  -export \
-file  rsa2048Cert1.cer \
-keystore  testkeys.jcks \
-alias  cert1 \
-storepass  $STOREPASS \
-storetype  JCEKS  \
-provider  IBMJCE  \
```

```
-keypass  $KEYPASS

echo "Finish export of cert1"

keytool  -import  \
-noprompt \
-alias  cert1ca \
-file  rsa2048Cert1.cer \
-keystore  testkeys.jcks \
-storepass  $STOREPASS \
-storetype  JCEKS \
-provider  IBMJCE
```

Complete hwkeytool example for JCE4758KS using hidden passwords

The script in Example B-6 on page 576 creates a JCE4758KS keystore with two self-signed certificates, each of which has a public and private key pair. The script then exports the first certificate created, and then the script reimports that certificate with a new alias as a trusted certificate entry. This entry does not have a private key associated with it. In this script, we use default values for the `hardwaretype` argument, which is going to be CLEAR. The default value for hardwareusage specified is KEYMANAGEMENT. Because we are creating RSA key pairs, the hardwareusage and `hardwaretype` defaults are different for DSA algorithms.

cert1	Certificate alias and public and private keys associated with this certificate
cert2	Certificate alias and public and private keys associated with this certificate
cert1ca	Trusted certificate entry with only a public key associated with this certificate
rsa2048Cert1.cer	DER-encoded file, which contains an exported public key that matches the public key of cert1

In this script, the first line tells the operating system that we use the bash shell to execute the commands in the script:

```
#!/bin/bash
```

This script is sufficiently generic that it can be executed using `/bin/sh`.

After we set up the shell with which to execute the script, we then use the **`echo`** command to prompt for the correct passwords.

The following command prevents what we type, in this case, a password, from printing to the command line:

```
stty -echo
```

We then read what is typed into the command line to a variable and plug the variable into our keytool commands.

Example: B-6 Password hidden hwkeytool script

```
#!/bin/bash
echo -n "Please Enter the Keystore password"
echo ""
stty -echo
read STOREPASS
stty echo
echo ""          #force a carriage return to be output
echo -n "Please Enter the private key password"
echo ""
stty -echo
read KEYPASS
stty echo
echo ""          #force a carriage return to be output


echo -n "start key generation"
hwkeytool  -genkey  \
-alias   cert1 \
-dname  CN=IBM  \
-keystore  testkeys.4758ks  \
-provider  IBMJCE4758  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  $KEYPASS  \
-storepass  $STOREPASS \
-storetype  JCE4758KS \
-validity  999

echo "Finish genkey for alias = cert1"

hwkeytool  -genkey  \
-alias  cert2  \
-dname  CN=IBM  \
-keystore  testkeys.4758ks  \
-provider  IBMJCE4758  \
-keyalg  RSA  \
-keysize  2048  \
-keypass  $KEYPASS \
-storepass $STOREPASS \
-storetype  JCE4758KS \
-validity  999

echo "Finish genkey for alias = cert2"

hwkeytool  -export \
-file  rsa2048Cert1.cer \
-keystore  testkeys.4758ks \
-alias  cert1 \
-storepass  $STOREPASS \
-storetype  JCE4758KS  \
-provider  IBMJCE4758  \
-keypass  $KEYPASS

echo "Finish export of cert1"

hwkeytool  -import  \
-noprompt \
-alias  cert1ca \
-file  rsa2048Cert1.cer \
```

```
-keystore  testkeys.4758ks \
-storepass  $STOREPASS \
-storetype  JCE4758KS \
-provider  IBMJCE4758

echo "Finish import of cert1ca"
```

Java

Java has become very popular for building new applications on distributed platforms and also on the mainframe. Java is a language whose specifications are maintained by SUN Microsystems. Java achieves this goal simply by compiling programs to byte code, which creates one or more class files, which can then be run by a machine specific Java Runtime Environment (JRE). A JRE consists of a Java virtual machine (JVM) and a just-in-time compiler (JIT). A JIT is used to translate a subset of Java byte code into native machine code at run time to improve performance, that is, *just in time*. Running a Java application without the JIT and incurring a large performance impact is possible. The existence of a JVM enables applications written in Java to be largely independent of the operating system in use.

Java was originally developed in the early 1990s by Sun Microsystems, Inc., as an object-oriented programming language based on the programming language C++. The syntax used in Java code originates from C++ and is therefore recognizable for those people with C++ experience.

To run a Java application, you have either a JRE or a Software Developer Kit (SDK) installed on the system where the application will run. A JRE only includes the JVM code required to execute Java applications. An SDK, which is also referred to as a JDK (Java Developer Kit), includes everything a JRE has, and in addition, it includes developer tools, such as the javac compiler, and gives developers access to Java APIs. An SDK must be downloaded and installed to run the EKM.

Security and providers

A set of security providers is included in the IBM Java SDKs. A *provider* is a set of common APIs to extend Java to add security capabilities. The provider architecture supplies algorithm implementations for service classes. A collection of all of the implementations is referred to as a *service provider* or simply a *provider*. The provider architecture supports the development of these plug-replaceable components.

More information about implementing your own security provider is at:

http://java.sun.com/j2se/1.4.2/docs/guide/security/HowToImplAProvider.html

A sampling of IBM providers includes:

IBMJCE	Java Cryptographic Extension
IBMJCE4758	JCE using z/OS hardware cryptographic services JDK 1.42 and earlier
IBMJCECCA	JCE using z/OS hardware cryptographic services JDK 5.0 and later
IBMJSSE	Java Secure Sockets Extension (SSL and TLS)
IBMJSSE2	Java Secure Sockets Extension (SSL and TLS). JSSE2 is aliased in JDK 5 and later as JSSE
IBMJAAS	Java Authentication and Authorization Service

IBMJGSS	Generic Security Services: Kerberos and GSS-API (JDK 1.4.0 and later only)
IBMPKCS11	Use hardware cryptography using the Public Key Cryptographic Standards #11(PKCS#11 standard)

Garbage Collector

Another important feature of the JVM is the Garbage Collector. The *Garbage Collector* is a background thread of the JVM to reclaim unusable space and is extremely important for the performance of the JVM.

When a Java program makes a request for storage and the JVM is unable to comply, an allocation failure occurs. This allocation failure triggers the Garbage Collection process. Garbage Collection is the process of automatically freeing objects that are no longer referenced by the program.

Note: The behavior of the Garbage Collector is related to the heap size. A small heap can produce more frequent, but shorter Garbage Collector cycles. A large heap size, alternatively, produces less frequent, but longer Garbage Collector cycles.

IBM is a major supporter and user of Java across all of the IBM computing platforms, from Open Systems to z/OS. As a middleware product, Java offers an application programming interface (API) and a JVM to run programs. The existence of a JVM means that applications written in Java are largely independent of the operating system in use.

Over the years, Java has been shown to be very popular and the JVM is available on nearly all available operating systems, such as Microsoft Windows, Apple OS-X, i5/OS, OS/400, Linux, UNIX, and z/OS UNIX.

The Java SDK contains the program products to assist application developers in the use of the Java APIs.

Note: To simulate the use of Java on the mainframe, IBM came up with a special processor for running Java applications. It is called the System z Application Assist Processor, also known as zAAP. This type of processor is an optional feature in System z hardware. When bought and installed, zAAP allows the client to benefit from additional resources available only for Java executable code. In return, the client gets no additional software license fees, because this processor type is outside of the scope of IBM mainframe software licensing and pricing calculations.

Java SDKs contain industry standard APIs, and each SDK product can be ordered and serviced independently and separately. The available Java SDKs for IBM systems are available through the Internet or by placing a regular software order through IBM Software Manufacturing. For more information about these products and for download instructions, refer to:

`http://www.ibm.com/developerworks/java/jdk/`

Verifying the installation

Your path must contain the binary directory where Java is located:

```
PATH=/usr/lpp/java/J5.0/bin
CLASSPATH=/usr/lpp/java/J5.0/lib/
LIBPATH=/usr/lpp/java/J5.0/lib/ext/:
JAVA_HOME=/usr/lpp/java/J5.0/
```

The PATH statement is an environment variable to tell a UNIX-type system where the programs (binaries) are located, in this case, the program `java`.

Now, verify that Java is correctly installed (Figure B-2).

```
/u/johann>java -fullversion
java full version "J2RE 1.5.0 IBM z/OS build pmz31dev-20060607
(SR2)"
/u/johann>
```

Figure B-2 Java version

Note: Installing multiple versions of SDK products is possible. The environment variable is what points to a specific version of the SDK at run time. Each user can use a SDK version at any time, independent from other users.

z/OS region size

Java programs can take up a lot of system resources, sometimes even more than you might expect. Therefore, it is always necessary to examine your region size and allow for heap and stack storage requirements, Language Environment® code, Java code, and Language Environment internal control blocks.

The *region size* describes the amount of storage in which a user is allowed to run or execute programs. This value determines what kinds of programs (depending on their size) and how many programs are executable at the same time.

Policy files

Because of governmental export regulations, JDKs are set up with the ability to perform limited function cryptography. For full function cryptography, you must install the unrestricted policy files. They are located at this Web site:

`http://www.ibm.com/servers/eserver/zseries/software/java/j5jcecca.html`

Copy the unrestricted policy files to `$JAVA_HOME/lib/security`. Make sure that the old policy files are replaced or moved out of the JDK filesystem.

Tip: Some JDKS have policy files located in `$JAVA_HOME/demo/jce/policy-files/`.

C

Asymmetric and Symmetric Master Key change procedures

This appendix presents detailed directions about how to change the Asymmetric Master Key (ASM-MK) and Symmetric Master Key (SYM-MK) for Integrated Cryptographic Service Facility (ICSF).

Asymmetric Master Key change ceremony

This section outlines the steps for changing the Public key algorithm (PKA) Master Key. After a new Master Key has been generated, the new key parts must be secured.

Prerequisites

The following steps outline actions that must be completed before beginning a key change ceremony:

1. Know the date that the ceremony will take place. This is required for the naming of the PKDS, and for uniquely identifying the correct key parts.
2. A PKDS must be created prior to the key change ceremony. The PKDS must be created with the following naming convention, as shown here:

   `PKDS: MVSSYS.CSF.<SYSPLEX>.D<YYMMDD>.CSFPKDS`
3. The devices, which will hold the key parts, must be obtained prior to starting the key change ceremony.

Encryption and decryption test

This section is optional, but is recommended. This procedure proves that data keys can be correctly accessed from the PKDS, and data can be encrypted and decrypted before attempting to change the ASYM-MK key. This procedure also provides data to verify data keys that are protected by the ASYM-MK key are available after the ASYM-MK key has been changed. This procedure must be done once for each Sysplex:

1. From the main ISPF panel, enter option **6 PPINIT**.
2. From this panel, enter the command EX DATA.SET.NAME(*MEMBER*).
3. Return to the main ISPF panel (press F3) and enter **MVS**.
4. From the MVS panel, enter **SDSF**.
5. From the SDSF panel, enter **ST**.
6. Select the job that was just executed with an **S**. It will be called <*JOBNAME*>.
7. Scroll to the bottom to verify that the procedure worked correctly.

Pre-key change: Disable PKA services for all images in the Sysplex

Before changing the ASYM-MK Master Key, the PKA services and all PKDS access must be disabled within the Sysplex:

1. Go to the Integrated Cryptographic Service Facility (ICSF) main panel shown in Figure C-1 on page 583.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-1 ICSF panel

2. From the main ICSF panel, choose option **4 ADMINCNTL**. Press Enter. This takes you to the panel shown in Figure C-2.
3. Select the following services to disable them. Refer to Figure C-2:
   - **PKA Callable Services**
   - **PKDS Read Access**
   - **PKDS Write, Create, and Delete Access**

   A status message appears in the upper right corner of the panel indicating whether the services have been stopped.

```
------------------- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

        Active CKDS: MYSYS.TST.TESTPLEX.CKDS
        Active PKDS: MYSYS.TST.TESTPLEX.PKDS


 To change the status of a control, enter the appropriate character
  (E - ENABLE, D - DISABLE) and press ENTER.


   FUNCTION                                              STATUS
   --------                                              ------
   Dynamic CKDS Access                                   ENABLED
d  PKA Callable Services                                 ENABLED
d  PKDS Read Access                                      ENABLED
d  PKDS Write, Create, and Delete Access                 ENABLED
******************************Bottom of data ********************************
```

Figure C-2 ICSF Administrative Control Functions panel

4. Return to the beginning of this section “Pre-key change: Disable PKA services for all images in the Sysplex” on page 582 and execute these instructions on all logical partitions (LPARs) in the Sysplex.

Key change: First LPAR in the Sysplex

The process for changing Master Keys in a Sysplex is different on the first LPAR than it is on subsequent LPARs in the Sysplex. On the first LPAR, the key parts are generated and the PKDS is reenciphered. On all subsequent LPARs in the Sysplex, the Master Keys are simply set and the newly reenciphered PKDS is activated.

Generate ASYM-MK key parts

To generate ASM-MK key parts:

1. From the main ICSF panel in Figure C-3, choose option **5 UTILITY**. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 5
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-3 ICSF panel

2. Select option **3 RANDOM** to access the Random Number Generator panel (Figure C-4). Press Enter.

```
------------------------------ ICSF - Utilities -------------------------------
OPTION ===> 3

Enter the number of the desired option.

   1  ENCODE           - Encode Data
   2  DECODE           - Decode Data
   3  RANDOM           - Generate a random number
   4  CHECKSUM         - Generate a checksum and verification and
                         hash pattern
   5  PPKEYS           - Generate master key values from a pass phrase




   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-4 ICSF Utilities panel

3. Type `RANDOM` as the parity of the number you are generating. Asymmetric keys can have either EVEN or ODD parity. Press Enter. See Figure C-5 on page 585.

```
----------------------- ICSF - Random Number Generator ------------------------
COMMAND ===>

Enter data below.

  Parity option  ===> RANDOM            ODD, EVEN, RANDOM
  Random Number1    : 23A836458CE01957  Random Number 1
  Random Number2    : 3E1C3987AC763980  Random Number 2
  Random Number3    : D5E3733972057A19  Random Number 3



Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure C-5 ICSF Random Number Generator panel

4. Press F3 to return, and then select option **4 CHECKSUM** (see Figure C-6). Press Enter.

```
------------------------------ ICSF - Utilities ------------------------------
OPTION ===> 4

Enter the number of the desired option.

   1  ENCODE           - Encode Data
   2  DECODE           - Decode Data
   3  RANDOM           - Generate a random number
   4  CHECKSUM         - Generate a checksum and verification and
                         hash pattern
   5  PPKEYS           - Generate master key values from a pass phrase



   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-6 ICSF Utilities panel

5. Enter PKAMSTR for the key type (Figure C-7 on page 586).

```
--------------- ICSF - Checksum and Verification and Hash Pattern ---------------
OPTION ===>

Enter data below:

   Key Type       ===> PKAMSTR             (Selection panel displayed if blank)

   Key Value      ===> 2E48D75FF03BF357    Input key value 0 - 7
                  ===> 3E1C0B0D7DAC7443    Input key value 8 - 15
                  ===> D5BA722FE0F99296    Input key value 16 - 23 (PKA Keys only)

   Checksum          : BC                  Check digit for key value
   Key Part VP       :                     Verification Pattern
   Key Part HP       : A3AC4D3428D8A6D7    Hash Pattern
                     : 6C0C471D55B8E123


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-7 ICSF Checksum and Verification and Hash Pattern panel

6. Copy the entire panel as text into a Notepad document and save it on the USB device as shown in Figure C-8. The USB device has a folder for each Sysplex. Be sure the key part is saved in the correct folder. The naming convention of the file has to be:

   ASYM-MK.<*SYSPLEX*>.D<*YYMMDD*>.<*KEYPART*>.txt

   <*KEYPART*> can be one of FIRST, MIDDLE*x*, or FINAL.

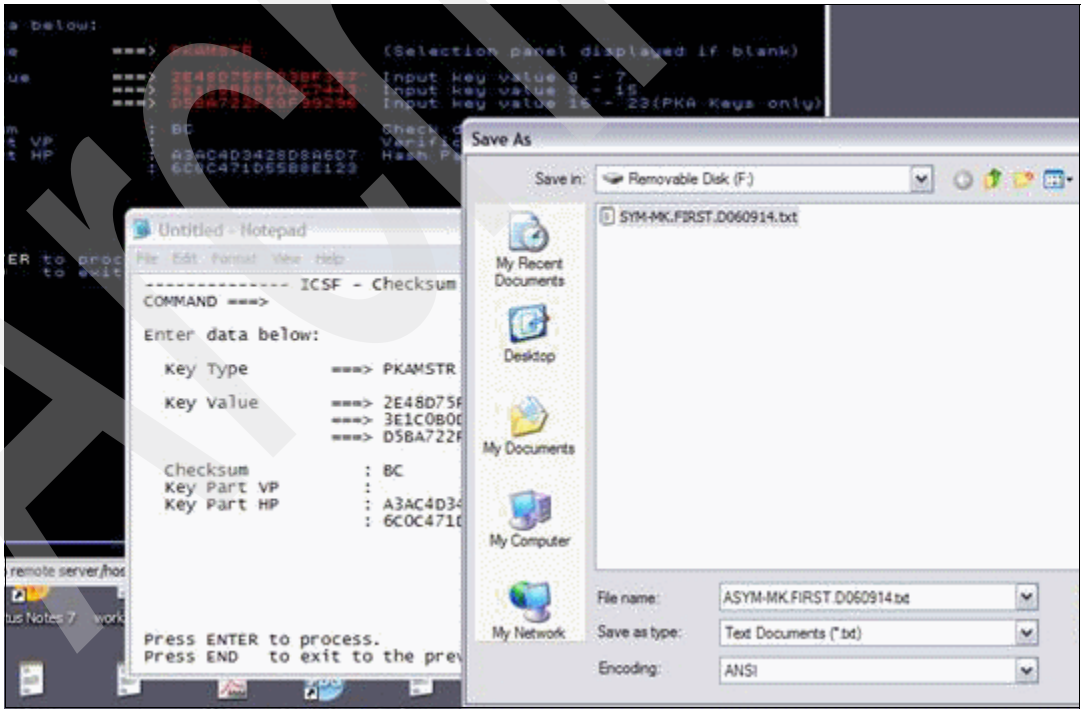


Figure C-8 Copying the panel text

Note: There is no Verification Pattern for the asymmetric key part. For your convenience, this key part file must remain open until it has been entered into ICSF.

Enter the ASYM-MK key parts

To enter the key parts:

1. Return two panels back (F3 two times) from this menu and select option **1 COPROCESSOR MGMT** (Figure C-9). Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-9 ICSF main panel

2. As shown in Figure C-10, select the coprocessors on which to set the Master Key by typing `e` to the left of the coprocessor name. Any coprocessor that has a prefix of `E` or `X` (for example `X01`) must be selected here. Press Enter.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR       SERIAL NUMBER   STATUS
   --------          -------------   ------
e  X00               77712345        ACTIVE
e  X01               77712346        ACTIVE
e  X02               77712347        ACTIVE
e  X03               77712348        ACTIVE
******************************Bottom of data ********************************
```

Figure C-10 ICSF Coprocessor Management panel

3. In Figure C-11 on page 588, enter `ASYM-MK`, one of `FIRST`, `MIDDLE`, or `FINAL` depending on which key part you are entering, the checksum, and the key value that was just generated.

   Note that the Asymmetric-keys master register is empty. Press Enter and you see a status message in the upper right corner indicating that the key part has been loaded. Check to be sure the Hash Pattern (HP) is the same as what you recorded earlier.

```
---------------------- ICSF - Clear Master Key Entry ------------------------
COMMAND ===>
              Symmetric-keys new master key register          :  FULL
              Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> ASYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST            (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> BC

   Key Value ===> 2E48D75FF03BF357
             ===> 3E1C0B0D7DAC7443
             ===> D5BA722FE0F99296   (ASYM-MK only)


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-11 ICSF Clear Master Key Entry panel

4. Go back to the beginning of this section. Generate ASYM-MK key parts and repeat the same instructions for each key part. For the previous step, enter either `MIDDLE` or `FINAL` for the key part as appropriate. Ensure the FINAL key part is indeed entered last. Also, be sure that you record the order in which the keys have been entered so that they can be reentered when recovering the Master Keys in Disaster Recovery or if the device has been tampered with.

Reencipher the PKDS under the new asymmetric-keys master keys

To reencipher:

1. From the main ICSF panel, enter option **2 Master Key** as shown in Figure C-12.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 2
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-12 ICSF Main Panel

2. Select option **6 REENCIPHER PKDS** as shown in Figure C-13 on page 589. Press Enter.

```
------------------------- ICSF - Master Key Management   ------------------
OPTION ===> 6

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initialize a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-13 ICSF Master Key Management panel

3. Enter the current and new data set names for the PKDS as shown in Figure C-14. Note that for convenience, the PKDS data set name has the date that the reencipher happened.

```
--------------------------- ICSF - Reencipher PKDS ----------------------------
COMMAND ===>

To reencipher all PKDS entries from encryption under the old signature/
asymmetric-keys master key to encryption under the current master key
enter the PKDS name below.

  Input  PKDS ===> 'MYSYS.CSF.CSFPKDS'
  Output PKDS ===> 'MYSYS.CSF.TESTPLEX.D060920.CSFPKDS'


Press ENTER to reencipher the PKDS.
Press END   to exit to the previous menu.
```

Figure C-14 ICSF Reencipher PKDS panel

Activate the PKDS

To activate the PKDS:

1. Return to the previous panel and select option **7 ACTIVATE PKDS**. See Figure C-15 on page 590. Press Enter.

```
------------------------- ICSF - Master Key Management   ------------------
OPTION ===> 7

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initialize a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-15 ICSF Master Key Management panel

2. Note that the New PKDS specified in the previous panel displays here. If it does not, type it now as shown in Figure C-16. Press Enter.

```
------------  ICSF - Activate PKA Cryptographic Key Data Set -----------------------
Command =====>

Enter the name of the new PKDS below.

  New PKDS =====> 'MYSYS.CSF.TESTPLEX.D060920.CSFPKDS'




Press ENTER to activate the PKDS.
Pree END    to exit to the previous menu.
```

Figure C-16 ICSF Activate PKA Cryptographic Key Data Set panel

Key change: Subsequent LPARs in the Sysplex

These instructions assume that a new ASYM-MK Master Key has already been generated and the new PKDS has already been reenciphered.

Enter the ASYM-MK key parts

To enter the key parts:

1. From the main ICSF panel, choose option **1 COPROCESSOR MGMT** as shown in Figure C-17 on page 591. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-17 ICSF Main panel

2. Select the processors on which to set the Master Key by typing e to the left of the coprocessor as shown in Figure C-18. Press Enter.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR       SERIAL NUMBER   STATUS
   --------          -------------   ------
e  X00               77712345        ACTIVE
e  X01               77712346        ACTIVE
e  X02               77712347        ACTIVE
e  X03               77712348        ACTIVE
*******************************Bottom of data ********************************
```

Figure C-18 ICSF Coprocessor Management panel

3. Open the document residing on the USB device, which corresponds to the correct key part as shown in Figure C-19 on page 592.

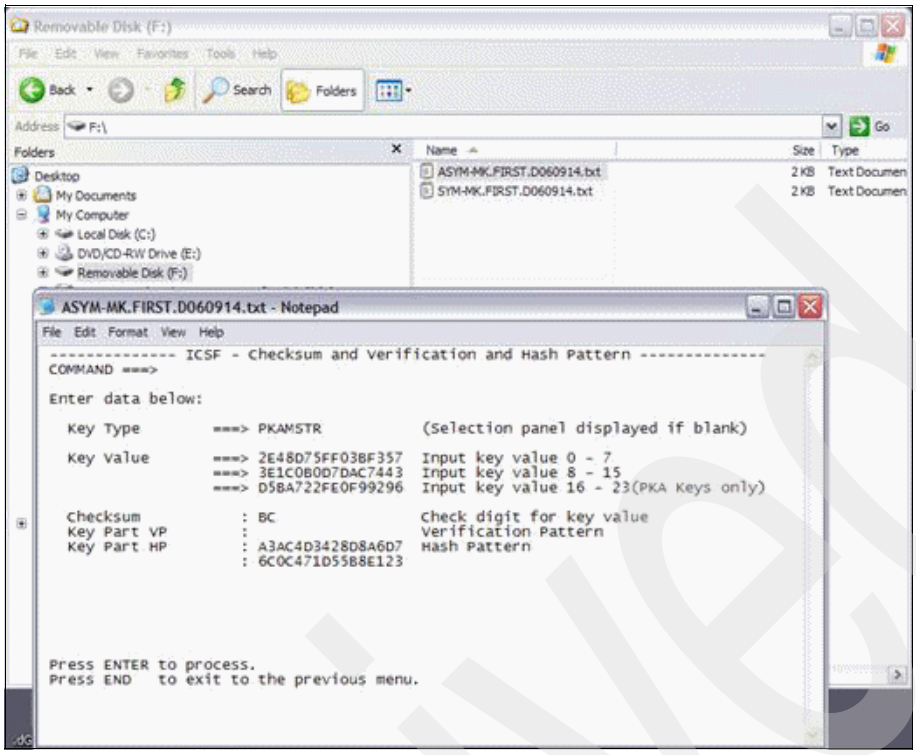


Figure C-19 Working with the previously stored document on the USB device

4. In this panel, enter ASYM-MK, Part (FIRST, MIDDLE, or FINAL), the Checksum, and the Key Value (see Figure C-20). Note that the key registers are both empty. Press Enter and you see a status message in the upper right corner indicating the key part has been loaded. Check to be sure the Verification Pattern (VP) and Hash Pattern (HP) are the same as what you recorded earlier.

```
---------------------- ICSF - Clear Master Key Entry ------------------------
COMMAND ===>
            Symmetric-keys new master key register          :  EMPTY
            Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> ASYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST           (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> BC

   Key Value ===> 2E48D75FF03BF357
             ===> 3E1C0B0D7DAC7443
             ===> D5BA722FE0F99296   (ASYM-MK only)


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-20 ICSF Clear Master Key panel

5. Go back to the beginning of this section. Enter ASYM-MK key parts and repeat the same instructions for each key part owner. For the previous step, enter either MIDDLE or FINAL for the key part as appropriate. Ensure the FINAL key part is indeed entered last.

Activate the new PKDS

After all key parts have been entered, continue to activate the new PKDS:

1. From the ICSF Main panel, select option **2 MASTER KEY** as shown in Figure C-21. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 2
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-21 ICSF Main Panel

2. Enter option **7 ACTIVATE PKDS** to activate the new PKDS. See Figure C-22.

```
------------------------- ICSF - Master Key Management   -----------------
OPTION ===> 7

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initialize a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-22 ICSF Master Key Management panel

3. Enter the new PKDS data set as shown in Figure C-23 on page 594.

```
------------  ICSF - Activate PKA Cryptographic Key Data Set -----------------------
Command =====>

Enter the name of the new PKDS below.

  New PKDS =====> 'MYSYS.CSF.TESTPLEX.D060920.CSFPKDS'




Press ENTER to activate the PKDS.
Pree END    to exit to the previous menu.
```

Figure C-23 ICSF Activate PKA Cryptographic Key Data Set panel

Post-key change: All LPARs in the Sysplex

This set of instructions must be completed on all LPARs in the Sysplex after the PKDS has been activated on all LPARs.

Change installation options to reflect the new PKDS

To change installation options:

1. Go to the ISPF Primary Options Menu panel and enter option **3 Utilities** as shown in Figure C-24.

```
                         - ISPF Primary Options Menu-
OPTION ==>  3
             --- My Options ---                    .----- All The Options -----.
LOG  SPF Options          CP   Copy/Move           !  ----- Top Of Data ------ !
1    Browse               DU   Dataset Utility     !  0    Settings            !
2    Edit                 LU   Library Utility     !  1    View                !
3    Utilities            TE   Dialog Test         !  2    Edit                !
4    SPF Foreground       V    VTOC Utility        !  3    Utilities           !
5    SPF Background                                !  4    Foreground          !
6    TSO                                           !  5    Batch               !
7    Tutorial                                      !  6    Command             !
SD   SDSF                                          !  7    Dialog Test         !
                                                   !  8    LM Facility         !
                                                   !  9    IBM Products        !
                                                   !  10   SCLM                !
                                                   '---------------------------'
                                                   ****************************
                                                   *     Workbench Options    *
                                                   * XX   Change Colours/Title *
                                                   * YY   Set Standard Options *
                                                   * ZZ   Use Personal Options *
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
```

Figure C-24 ISPF Primary Options Menu panel

2. Select option **4** for the Dslist Utility and press Enter. See Figure C-25 on page 595.

```
 Menu  Help
_______________________________________________________________________________

                          Utility Selection Panel

1  Library     Compress or print data set.  Print index listing.  Print,
                 rename, delete, browse, edit or view members
2  Data Set    Allocate, rename, delete, catalog, uncatalog, or display
                 information of an entire data set
3  Move/Copy   Move, or copy members or data sets
4  Dslist      Print or display (to process) list of data set names.
                 Print or display VTOC information
5  Reset       Reset statistics for members of ISPF library
6  Hardcopy    Initiate hardcopy output
7  Transfer    Download ISPF Client/Server or Transfer data set
8  Outlist     Display, delete, or print held job output
9  Commands    Create/change an application command table
11 Format      Format definition for formatted data Edit/Browse
12 SuperC      Compare data sets                          (Standard Dialog)
13 SuperCE     Compare data sets Extended                 (Extended Dialog)
14 Search-For  Search data sets for strings of data       (Standard Dialog)
15 Search-ForE Search data sets for strings of data Extended (Extended Dialog)
Option ===>   4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
```

Figure C-25 Utility Selection panel

3. Enter the data set name (Dsname Level), in our example, SYS1.TEST.PARMLIB, from the Data Set List Utility panel shown in Figure C-26.

```
  Menu  RefList  RefMode  Utilities  Help
_______________________________________________________________________________

                          Data Set List Utility
                                                                    More:     +
   blank Display data set list              P Print data set list
       V Display VTOC information           PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . . SYS1.TEST.PARMLIB
   Volume serial  . .

Data set list options
   Initial View . . . 1  1. Volume       Enter "/" to select option
                         2. Space        /  Confirm Data Set Delete
                         3. Attrib       /  Confirm Member Delete
                         4. Total        /  Include Additional Qualifiers
                                         /  Display Catalog Name

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
Option ===> 4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
```

Figure C-26 Data Set List Utility panel

4. Select the ICSF Installation Options data set by using the E (edit) option as shown in Figure C-27 on page 596.

```
  Menu  Options  View  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
DSLIST - Data Sets Matching SYS1.TEST.PARMLIB                    Row  1 of  1

Command - Enter "/" to select action                Message           Volume
-------------------------------------------------------------------------------
E        SYS1.TEST.PARMLIB                                                TSOE01
****************************** End of Data Set list ***************************









Command ===>                                                  Scroll ===> CSR
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Figure C-27 Selecting the ICSF Installation Options data set

5. Select the correct CSFPRM*xx* PARMLIB member and press Enter.
6. Change the name of the PKDS in the Installation Options Data Set as shown in Figure C-28.

```
****************************** Top of Data ***************************
/*                                                                    */
/*     LICENSED MATERIALS - PROPERTY OF IBM                           */
/*                                                                    */
/*     "RESTRICTED MATERIALS OF IBM"                                  */
/*     5694-A01                                                       */
/*                                                                    */
/*     (C) COPYRIGHT IBM COPR. 1990, 2003                             */
/*                                                                    */
/*     STATUS = HCR770A                                               */
/*                                                                    */
CKDSN(MYSYS.CSF.TESTPLEX.D060920.CSFCKDS)
PKDSN(MYSYS.CSF.TESTPLEX.D060920.CSFPKDS)
COMPAT(NO)
SSM(NO)
KEYAUTH(NO)
CKTAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

Figure C-28 Editing the PARMLIB member

7. Save the PARMLIB member.

Enable PKA services and PKDS read and write access

To enables services and access:

1. Return to the main ICSF panel and select option **4 ADMINCNTL** from the ISCF Main menu shown in Figure C-29 on page 597. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-29 ISCF Main menu

2. Select the services to enable and press Enter as shown in Figure C-30. A message appears in the upper right corner indicating whether the changes were successful.

```
------------------- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

         Active CKDS: MYSYS.CSF.TESTPLEX.CSFCKDS
         Active PKDS: MYSYS.CSF.TESTPLEX.CSFPKDS


 To change the status of a control, enter the appropriate character
 (E - ENABLE, D - DISABLE) and press ENTER.


   FUNCTION                                          STATUS
   --------                                          ------
   Dynamic CKDS Access                               ENABLED
e  PKA Callable Services                             DISABLED
e  PKDS Read Access                                  DISABLED
e  PKDS Write, Create, and Delete Access             DISABLED
********************************Bottom of data ********************************
```

Figure C-30 ICSF Administrative Control Functions panel

Key change verification

Now that the ASYM-MK Master Keys have been changed on all logical partitions (LPARs), you must run a sample application, which uses keys protected by the ASYM-MK:

1. Run a sample job, which demonstrates that keys are again accessible.
2. Return to “Post-key change: All LPARs in the Sysplex” on page 594 and repeat for each LPAR in the Sysplex.

Post key change: Key part backup

After the ASYM-MK has been changed, the key parts must be backed up to the secondary USB device. This is labeled <*INSERT LABEL CONVENTION*>.

To back up the key parts:

1. Connect both the primary and backup USB devices to the computer. This opens a Windows Explorer window for each device as shown in Figure C-31.

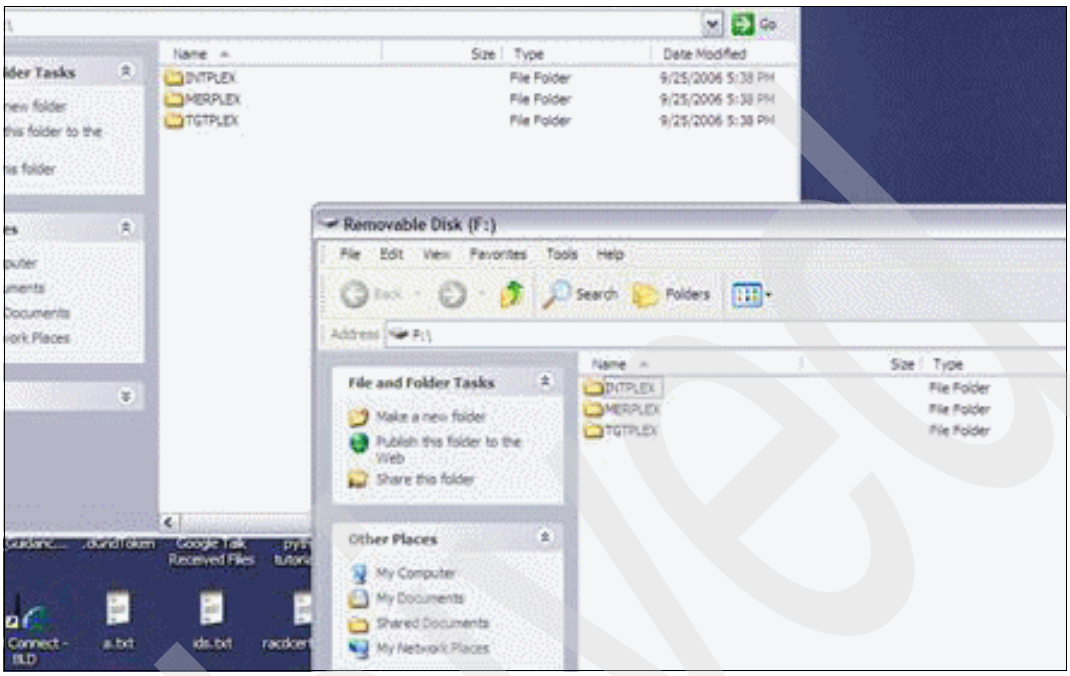


Figure C-31 Opening Windows Explorer windows

2. From the primary USB device window, copy the new key parts from each folder into the corresponding folder in the backup device as shown in Figure C-32.

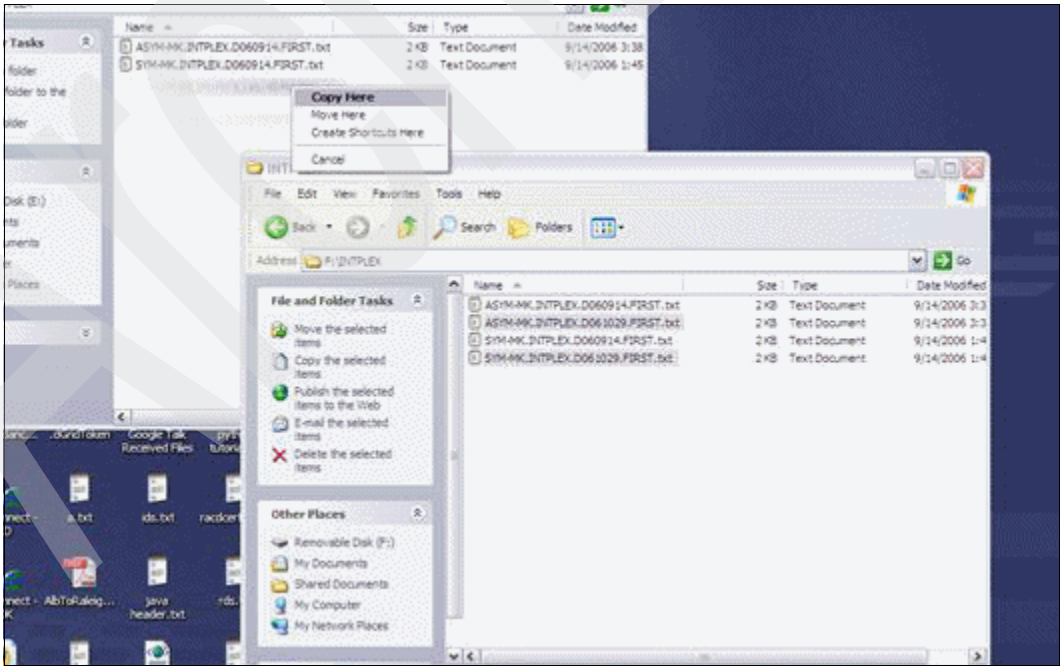


Figure C-32 Copying the new key parts

3. Repeat this process for each folder so that all new key parts get copied to the backup device.

ICSF tips

In this section, we summarize a few hints and tips for ICSF.

Creating a PKDS VSAM data set

Example C-1 lists the sample JCL to create a PKDS VSAM cluster. The items noted in bold in Example C-1 must be changed before attempting to execute the JCL:

- The Job card parameters must be changed to reflect your practices.
- Change <*SYSPLEX*> to the Sysplex name.
- Change <*YYMMDD*> to the date the key change ceremony will occur.
- Change VOLUME(*XXXXXX*) to the volume where the PKDS must reside.

Example: C-1 Sample JCL to create a PKDS VSAM cluster

```
//CSFPKDS JOB     job card parameters
//*******************************************************************
//* Licensed Materials - Property of IBM *
//* 5694-A01 *
//* (C) Copyright IBM Corp. 2002,2003 *
//* *
//* THIS JCL DEFINES A VSAM PKDS TO USE FOR ICSF *
//* *
//* CAUTION: This is neither a JCL procedure nor a complete JOB. *
//* Before using this JOB step, you will have to make the following *
//* modifications: *
//* *
//* 1) Add the job parameters to meet your system requirements. *
//* 2) Be sure to change CSF to the appropriate HLQ if you choose *
//*    not to use the default. *
//* 3) Change xxxxxx to the volid where you want your PKDS to *
//*    reside. The PKDS needs to be on a permanently resident *
//*    volume. *
//* *
//*******************************************************************
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  DEFINE CLUSTER (NAME(<SYSID>.CSF.<SYSPLEX>.D<YYMMDD>.CSFPKDS) -
                  VOLUME(xxxxxx) -
                  RECORDS(100,50) -
                  RECORDSIZE(350,2800) -
                  KEYS (72 0) -
                  FREESPACE(0,0) -
                  SHAREOPTIONS(2,3) -
                  UNIQUE) -
         DATA (NAME(<SYSID>.CSF.<SYSPLEX>-D<YYMMDD>.CSFPKDS.DATA -
                  BUFFERSPACE(100000) -
                  ERASE -
                  CISZ(8192) -
                  WRITECHECK) -
         INDEX (NAME(<SYSID>.CSF.<SYSPLEX>-D<YYMMDD>.CSFPKDS.INDEX))
```

Note: Changing parameters other than those marked in bold in this job can cause unpredictable results with ICSF. Make any other changes at your own risk.

This checklist gives you an overview of the process for changing the ASYM-MK:

- Create a new PDKS.
- All required people are available and have retrieved their key part device (USB device or paper).
- Do this to all images in the Sysplex before continuing:
  - Disable PKA services and PKDS read/write access.
- These steps must be done only to the first image in the Sysplex:
  - Generate the ASYM-MK key part.
  - Record ASYM-MK key part.
  - Enter ASYM-MK key part.
- Do this only to the first image in the Sysplex:
  - Reencipher the PKDS under the new asymmetric-keys Master Keys.
  - Activate the PKDS.
  - Change the Installation Options data set to reflect the new PKDS.
- Do this on all subsequent LPARs in the Sysplex:
  - Enter the ASYM-MK key parts: Requires all key part owners.
  - Activate the new PKDS: Requires only one person.
- Do this to each member of the Sysplex after all ASYM-MK keys have been set:
  - Enable PKA services.
  - Enable PKDS read/write access.
  - Verify cryptographic services are working correctly.

  This requires only one administrator.

Symmetric Master Key change ceremony

This section outlines the steps for changing the Symmetric Master Key (SYM-MK), also known as the DES Master Key. Error handling procedures are out of the scope of this section. If an error occurs, return to the start of the section and start over.

Prerequisites

The following list outlines the actions, which must be completed prior to beginning a key change ceremony:

1. Determine the date that the ceremony will take place. This date is required for naming the CKDS, as well as for uniquely identifying the correct key parts.
2. A CKDS must be created prior to the key change ceremony. The CKDS must be created with the following naming convention:

   `CKDS: <SYSID>.CSF.<SYSPLEX>.D<YYMMDD>.CSFCKDS`
3. Devices, which will hold the key parts, must be obtained prior to starting the key change ceremony.

Encryption and decryption test

This section is optional, but we highly recommend it. This procedure proves that data keys can be correctly accessed from the CKDS and that data can be encrypted and decrypted before attempting to change the SYM-MK key. This procedure also provides data to verify data keys protected by the SYM-MK key are available after the SYM-MK key has been changed. This procedure has to be done once for each Sysplex:

1. From the main ISPF panel, enter option **6**.
2. From this panel, enter the following command 'EX DATA.SET.NAME(*MEMBER*)'.
3. Return to the main ISPF panel (F3) and enter MVS.
4. From the MVS panel, enter SDSF.
5. From the SDSF panel, enter ST.
6. Select the job that was just executed with an S. It is called <*JOBNAME*>.
7. Scroll to the bottom to verify the procedure worked correctly.

Disable dynamic CKDS updates for all images in the Sysplex

The first step is to disable dynamic CKDS updates on every image in the Sysplex. This ensures that the CKDS does not become out of sync between images in the Sysplex. This step must be done on each image of the Sysplex before changing the Master Keys.

Follow these steps:

1. Go to the ICSF main panel and enter option **4 ADMINCNTL** as shown in Figure C-33.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-33 Utility Selection panel

2. Disable Dynamic CKDS Access by selecting it with a D; see Figure C-34 on page 602.

```
------------------- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

         Active CKDS: MYSYS.TST.TESTPLEX.CKDS
         Active PKDS: MYSYS.TST.TESTPLEX.PKDS


 To change the status of a control, enter the appropriate character
 (E - ENABLE, D - DISABLE) and press ENTER.


   FUNCTION                                                  STATUS
   --------                                                  ------
d  Dynamic CKDS Access                                       ENABLED
d  PKA Callable Services                                     ENABLED
d  PKDS Read Access                                          ENABLED
d  PKDS Write, Create, and Delete Access                     ENABLED
******************************Bottom of data ********************************
```

Figure C-34 ICSF Administrative Control Functions panel

Key change: First LPAR in the Sysplex

The process for changing Master Keys in a Sysplex is different on the first LPAR than it is on subsequent LPARs in the Sysplex. On the first LPAR, the key parts are generated and the CKDS is reenciphered. On all subsequent LPARs in the Sysplex, the Master Keys are simply set and the newly reenciphered CKDS is read into memory.

Generate SYM-MK key parts

To generate the SYM-MK key parts:

1. From the main ICSF panel, choose option **5 UTILITY** as shown in Figure C-35. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 5
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-35 ICSF main menu

2. Choose option **3 RANDOM**, to access the Random Number Generator panel; see Figure C-36 on page 603. Press Enter.

```
------------------------------ ICSF - Utilities ------------------------------
OPTION ===> 3

Enter the number of the desired option.

   1  ENCODE           - Encode Data
   2  DECODE           - Decode Data
   3  RANDOM           - Generate a random number
   4  CHECKSUM         - Generate a checksum and verification and
                         hash pattern
   5  PPKEYS           - Generate master key values from a pass phrase




   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-36 ICSF Utilities panel

3. In the panel shown in Figure C-37, choose ODD as the parity of the number you are generating because of a restriction that ICSF only accepts odd parity numbers for SYM-MK key parts. Press Enter.

```
---------------------- ICSF - Random Number Generator ----------------------
COMMAND ===>

Enter data below.

  Parity option  ===> ODD               ODD, EVEN, RANDOM
  Random Number1     : 024357CD6E20573B  Random Number 1
  Random Number2     : 5ED68A97E5DF9234  Random Number 2
  Random Number3     : AB43E0E5F797795B  Random Number 3



Press ENTER to process.
Press END   to exit to the previous menu.
```

Figure C-37 ICSF Random Number Generator panel

4. Return with F3 to the panel shown in Figure C-38 on page 604 and select option **4 CHECKSUM**. Press Enter.

```
------------------------------ ICSF - Utilities ------------------------------
OPTION ===> 4

Enter the number of the desired option.

   1  ENCODE           - Encode Data
   2  DECODE           - Decode Data
   3  RANDOM           - Generate a random numberol Functions
   4  CHECKSUM         - Generate a checksum and verification and
                         hash pattern
   5  PPKEYS           - Generate master key values from a pass phrase




   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-38 ICSF Utilities panel

5. Because we know we are generating Master Key parts, enter `MASTER` in the panel shown in Figure C-39. This panel generates the new key part. Press Enter.

```
--------------- ICSF - Checksum and Verification and Hash Pattern ---------------
OPTION ===>

Enter data below:

   Key Type       ===> MASTER           (Selection panel displayed if blank)

   Key Value      ===> 024357CD6E20573B   Input key value 0 - 7
                  ===> 5ED68A97E5DF9234   Input key value 8 - 15
                  ===> 0000000000000000   Input key value 16 - 23 (PKA Keys only)

   Checksum          : B7                 Check digit for key value
   Key Part VP       : B01E7C6CCC40D9F2   Verification Pattern
   Key Part HP       : 597846226F3A612F   Hash Pattern
                     : 60952D2A50A45A15



   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-39 ICSF Checksum and Verification and Hash Pattern panel

6. Copy the entire panel as text into a Notepad document and save it on the USB device (see Figure C-40 on page 605). The USB device has a folder for each Sysplex. Be sure the key part is saved in the correct folder. The naming convention of the file is:

   `SYM-MK.<`*SYSPLEX*`>.D<`*YYMMDD*`>.<`*KEYPART*`>.txt`

   <*KEYPART*> can be one of FIRST, MIDDLE*x*, or FINAL.

Note: For your convenience, this key part file must remain open until it has been entered into ICSF.

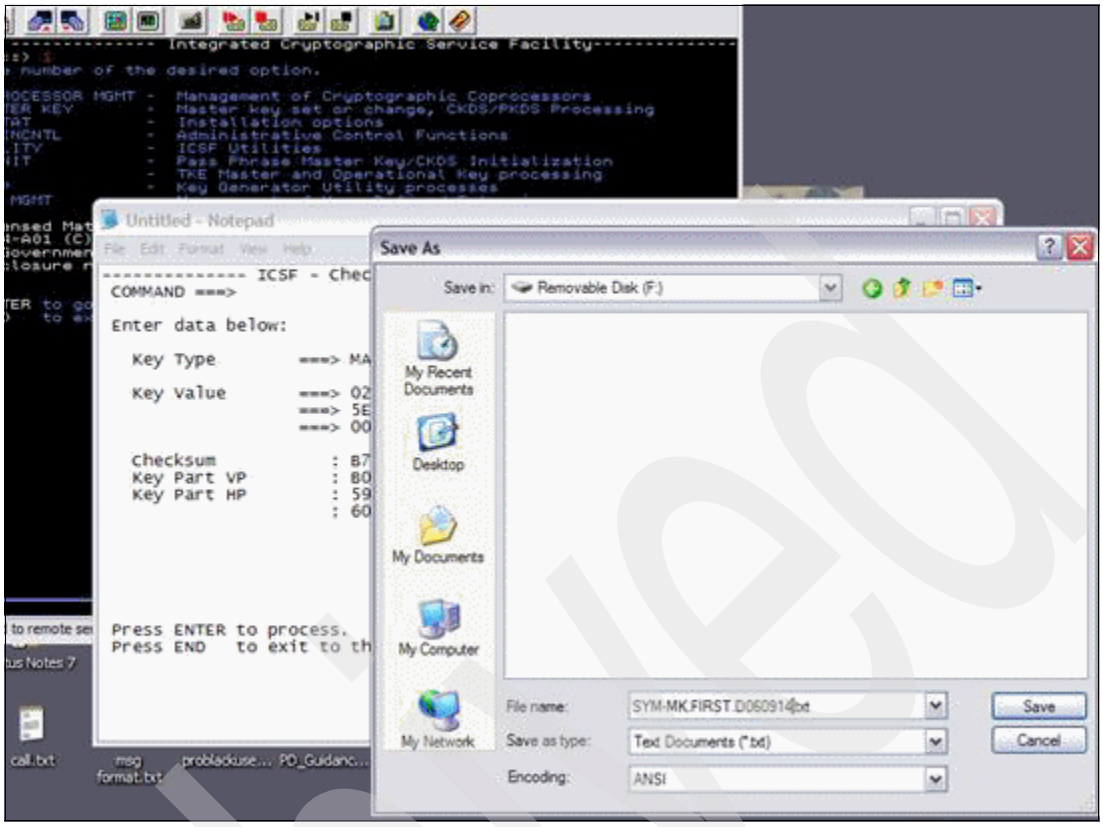


Figure C-40 Saving the data to the USB device

Enter SYM-MK key parts

To enter SYM-MK key parts:

1. From the panel shown in Figure C-39 on page 604, return two panels back (F3 twice) to the panel in Figure C-41 and select option **1 COPROCESSOR MGMT**. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-41 ICSF main menu

2. Select the processors to set the Master Key on with the E option (see Figure C-42). Any coprocessor with an action prefix of E or X must be selected here. Press Enter.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR       SERIAL NUMBER   STATUS
   --------          -------------   ------
e  X00               77712345        ACTIVE
e  X01               77712346        ACTIVE
e  X02               77712347        ACTIVE
e  X03               77712348        ACTIVE
*******************************Bottom of data ********************************
```

Figure C-42 ICSF Coprocessor Management panel

3. In this panel, enter SYM-MK, FIRST, the checksum, and the key value that was just generated. Note that the key registers in Figure C-43 are both empty.

```
---------------------- ICSF - Clear Master Key Entry -----------------------
COMMAND ===>
              Symmetric-keys new master key register          :  EMPTY
              Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> SYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST           (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> B7

   Key Value ===> 023457CD6E20573B
             ===> 5ED68A97E5DF9234
             ===> 0000000000000000   (ASYM-MK only)


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-43 ICSF Clear Master Key Entry panel

Press Enter and you see a status message in the upper right corner indicating the key part has been loaded. Check to be sure the Verification Pattern (VP) and Hash Pattern (HP) are the same as what you recorded earlier.

4. Go back to the beginning of this section. Generate SYM-MK key parts and repeat the same instructions by each key part owner. For the previous step, enter either MIDDLE or FINAL for the key part as appropriate. Ensure the FINAL key part is indeed entered last.

Reencipher the CKDS under the new SYM-MK Master Key

To reencipher:

1. Now that we have generated the new key parts, we can proceed to reenciphering the CKDS.

   Select option **2 Master Key** from the main ICSF menu shown in Figure C-44 on page 607. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility-----------------
OPTION ===> 2
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-44 ICSF main menu

2. Select option **3 REENCIPHER CKDS** from the ICSF Master Key Management panel as shown in Figure C-45. Press Enter.

```
-------------------------- ICSF - Master Key Management   -----------------
OPTION ===> 3

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initializw a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-45 Master Key Management panel

3. In the panel shown in Figure C-46 on page 608, specify the name of the existing CKDS and the name of the CKDS that will replace it. For convenience, a date stamp has been added to this data set, so that the date when the SYM-MK was changed last will be obvious to those people who maintain it.

```
--------------------------- ICSF - Reencipher CKDS ----------------------------
COMMAND ===>

To reencipher all CKDS entries from encryption under the old signature/
asymmetric-keys master key to encryption under the current master key
enter the PKDS name below.

  Input  PKDS ===> 'MYSYS.CSF.CSFCKDS'
  Output PKDS ===> 'MYSYS.CSF.TESTPLEX.D060920.CSFCKDS'


Press ENTER to reencipher the CKDS.
Press END   to exit to the previous menu.
```

Figure C-46 ICSF Reencipher CKDS panel

The message "REENCIPHER SUCCESSFUL" appears on the top right of the panel if the reencipher succeeds.

Change the new SYM-MK Master Key and activate the reenciphered CKDS

To change and activate:

1. From the panel shown in Figure C-46, return from the previous panel by pressing F3 and select option **4 CHANGE MK** as shown in Figure C-47. Press Enter.

```
------------------------- ICSF - Master Key Management  -----------------
OPTION ===> 4

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initialize a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-47 ICSF Master Key Management panel

2. The New CKDS, which was specified on the previous panel, must be filled in here. If not, fill in the CKDS data set name and press Enter.

```
-------------------------  ICSF - Change Master Key -----------------------------
Command =====>

Enter the name of the new CKDS below.

  New CKDS =====> 'MYSYS.CSF.TESTPLEX.D060920.CSFCKDS'

When the master key is changed, the new CKDS will become active.




Press ENTER to activate the CKDS.
Press END   to exit to the previous menu.
```

Figure C-48 ICSF Change Master Key panel

After changing the Master Key, a message appears in the upper right corner indicating whether the Master Key was changed.

Key change: Subsequent LPARs in the Sysplex

These instructions assume that a new SYM-MK Master Key has already been generated and the new CKDS has already been reenciphered.

Enter SYM-MK key parts

To enter key parts:

1. From the main ICSF panel, choose option **1 COPROCESSOR MGMT** as shown in Figure C-49. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 1
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-49 ICSF main panel

2. Select the processors to set the Master Key on with the E option (see Figure C-50 on page 610). Press Enter.

```
------------------- ICSF - Coprocessor Management ------------ Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, K, R and S. See the help panel for details.

   COPROCESSOR        SERIAL NUMBER   STATUS
   --------           -------------   ------
e  X00                77712345        ACTIVE
e  X01                77712346        ACTIVE
e  X02                77712347        ACTIVE
e  X03                77712348        ACTIVE
*******************************Bottom of data ********************************
```

Figure C-50 ICSF Coprocessor Management panel

3. Open the document residing on the USB device, which corresponds to the correct key part; see Figure C-51.

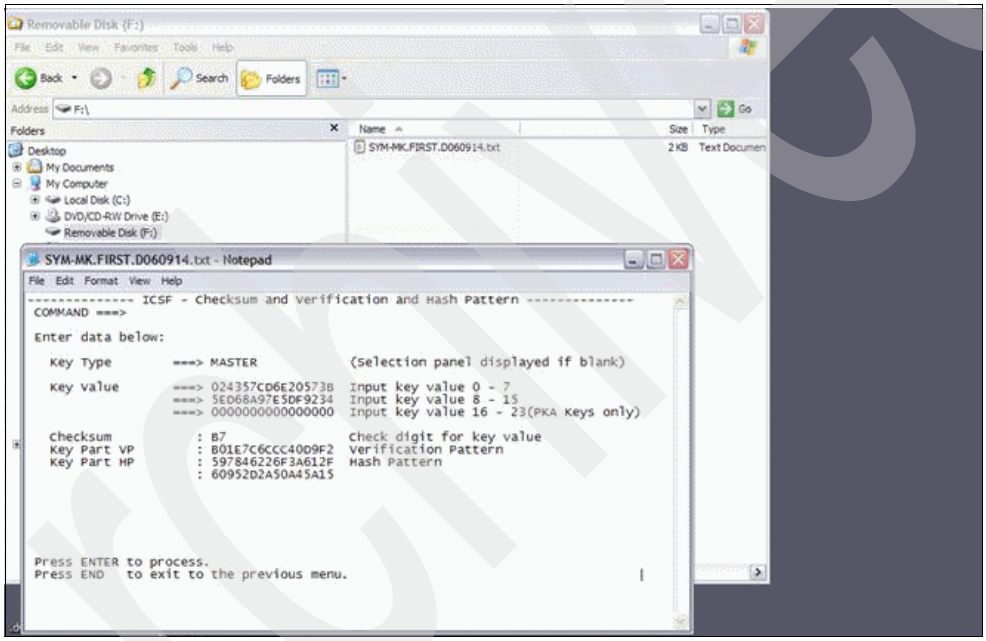


Figure C-51 Opening the document on the USB device

4. In the panel shown in Figure C-52 on page 611, enter `SYM-MK`, key part (`FIRST`, `MIDDLE`, or `FINAL`), the checksum, and the key value. Note that the key registers are both empty.

```
---------------------- ICSF - Clear Master Key Entry ------------------------
COMMAND ===>
              Symmetric-keys new master key register          :  EMPTY
              Asymmetric-keys new master key register         :  EMPTY

 Specify information below

   Key Type  ===> SYM-MK          (SYM-MK, ASYM-MK)

   Part      ===> FIRST           (RESET, FIRST, MIDDLE, FINAL)

   Checksum  ===> B7

   Key Value ===> 023457CD6E20573B
             ===> 5ED68A97E5DF9234
             ===> 0000000000000000   (ASYM-MK only)


   Press ENTER to process.
   Press END   to exit to the previous menu.
```

Figure C-52 ICSF Clear Master Key Entry panel

Press Enter and you see a status message in the upper right corner indicating the key part has been loaded. Check to be sure the Verification Pattern (VP) and Hash Pattern (HP) are the same as what you recorded earlier.

5. Go back to the beginning of this section to “Enter SYM-MK key parts” on page 605 and repeat the same instructions by each key part owner. For step 4, enter either `MIDDLE` or `FINAL` for the key part as appropriate. Ensure the FINAL key part is indeed entered last.

Activate the new Master Key

After all key parts have been entered, continue to activate the Master Key on this LPAR:

1. From the main ICSF panel, enter option 2 Master Key; see Figure C-53.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 2
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-53 ICSF main menu

2. Enter option **4 CHANGE MK** to change the Master Key as shown in Figure C-54 on page 612.

```
------------------------- ICSF - Master Key Management   ------------------
OPTION ===> 4

Enter the number of the desired option.

   1  INIT/REFRESH CKDS -  Initializw a Cryptographic Key Data Set or
                           acticvate an updated Cryptographic Key Data Set
   2  SET MK            -  Set a DES/symmetric-keys master key
   3  REENCIPHER CKDS   -  Reencipher the CKDS prior to changing the FDES
                           /symmetric-keys master key
   4  CHANGE MK         -  Change the DES/symmetric-keys master key and
                           activate the reenciphered CKDS
   5  INITIALIZE PKDS   -  Initialize or update a PKDS Cryptographic
                           Key Data Set header record
   6  REENCIPHER PKDS   -  Reencipher the PKA Cryptographic Key Data Set
   7  ACTIVATE PKDS     -  Activate the PKDS after it has been reenciphered
   8  REFRESH CACHE     -  Refresh the PKDS Cache if enabled

Press ENTER to go to the selected option.
Press END   to exit to the previous menu.
```

Figure C-54 ICSF Master Key Management panel

This panel displays a message in the upper right corner indicating whether the change was successful. If it was successful, proceed to the next LPAR in the Sysplex or to the Post key change section.

Post-key change: All LPARs in the Sysplex

This set of instructions must be completed on all LPARs in the Sysplex after the PKDS has been activated on all LPARs.

Change installation options to reflect the new CKDS

To change installation options:

1. Go to the ISPF Primary Options Menu panel and select option **3 Utilities** as shown in Figure C-55 on page 613. Press Enter.

```
                        - ISPF Primary Options Menu-
OPTION ==>  3
             --- My Options ---                    .----- All The Options -----.
LOG  SPF Options          CP   Copy/Move           !  ----- Top Of Data ------ !
1    Browse               DU   Dataset Utility     !  0    Settings            !
2    Edit                 LU   Library Utility     !  1    View                !
3    Utilities            TE   Dialog Test         !  2    Edit                !
4    SPF Foreground       V    VTOC Utility        !  3    Utilities           !
5    SPF Background                                !  4    Foreground          !
6    TSO                                           !  5    Batch               !
7    Tutorial                                      !  6    Command             !
SD   SDSF                                          !  7    Dialog Test         !
                                                   !  8    LM Facility         !
                                                   !  9    IBM Products        !
                                                   !  10   SCLM                !
                                                   '---------------------------'
                                                   ****************************
                                                   *     Workbench Options    *
                                                   * XX   Change Colours/Title *
                                                   * YY   Set Standard Options *
                                                   * ZZ   Use Personal Options *
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
```

Figure C-55 ISPF Primary Options Menu panel

2. Type 4 for the Dslist utility; see Figure C-56. Press Enter.

```
 Menu  Help
------------------------------------------------------------------------------

                          Utility Selection Panel

1  Library     Compress or print data set.  Print index listing.  Print,
                 rename, delete, browse, edit or view members
2  Data Set    Allocate, rename, delete, catalog, uncatalog, or display
                 information of an entire data set
3  Move/Copy   Move, or copy members or data sets
4  Dslist      Print or display (to process) list of data set names.
                 Print or display VTOC information
5  Reset       Reset statistics for members of ISPF library
6  Hardcopy    Initiate hardcopy output
7  Transfer    Download ISPF Client/Server or Transfer data set
8  Outlist     Display, delete, or print held job output
9  Commands    Create/change an application command table
11 Format      Format definition for formatted data Edit/Browse
12 SuperC      Compare data sets                          (Standard Dialog)
13 SuperCE     Compare data sets Extended                 (Extended Dialog)
14 Search-For  Search data sets for strings of data       (Standard Dialog)
15 Search-ForE Search data sets for strings of data Extended (Extended Dialog)
Option ===>   4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
```

Figure C-56 Utility Selection panel

3. Enter the data set name (Dsname Level) SYS1.TEST.PARMLIB as shown in Figure C-57 on page 614. Press Enter.

```
  Menu  RefList  RefMode  Utilities  Help
_______________________________________________________________________________

                          Data Set List Utility
                                                                      More:     +
   blank Display data set list               P Print data set list
       V Display VTOC information           PV Print VTOC information

Enter one or both of the parameters below:
   Dsname Level . . . SYS1.TEST.PARMLIB
   Volume serial  . .

Data set list options
   Initial View . . . 1  1. Volume       Enter "/" to select option
                         2. Space        /  Confirm Data Set Delete
                         3. Attrib       /  Confirm Member Delete
                         4. Total        /  Include Additional Qualifiers
                                         /  Display Catalog Name

When the data set list is displayed, enter either:
  "/" on the data set list command field for the command prompt pop-up,
  an ISPF line command, the name of a TSO command, CLIST, or REXX exec, or
Option ===> 4
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure C-57 Data Set List Utility panel

4. In the panel in Figure C-58, select the ICSF Installation Options data set by using the E (edit) option. Press Enter.

```
  Menu  Options  View  Utilities  Compilers  Help
_______________________________________________________________________________
DSLIST - Data Sets Matching MYSYS.TST.TEXTPLEX                  Row  1 of  1

Command - Enter "/" to select action                Message           Volume
-------------------------------------------------------------------------------
e       SYS1.TEST.PARMLIB                                             TSOE11
****************************** End of Data Set list ****************************



Command ===>                                                 Scroll ===> CSR
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Figure C-58 DSLIST panel

5. Select the correct CSFPRM*xx* PARMLIB member and change the name of the CKDS in the Installation Options Data Set as shown in Figure C-59 on page 615. Press Enter.

```
/*     LICENSED MATERIALS - PROPERTY OF IBM                          */
/*                                                                   */
/*     "RESTRICTED MATERIALS OF IBM"                                 */
/*     5694-A01                                                      */
/*                                                                   */
/*     (C) COPYRIGHT IBM COPR. 1990, 2003                            */
/*                                                                   */
/*     STATUS = HCR770A                                              */
/*                                                                   */
CKDSN(MYSYS.CSF.TESTPLEX.D060920.CSFCKDS)
PKDSN(MYSYS.CSF.TESTPLEX.D060920.CSFPKDS)
COMPAT(NO)
SSM(NO)
KEYAUTH(NO)
CKTAUTH(NO)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
PKDSCACHE(64)
```

Figure C-59 Updating the CSFPRMxx PARMLIB member

Enable Dynamic CKDS updates

The final step for the CKDS processing is to enable the Dynamic CKDS Access:

1. From the main ICSF panel, choose option **4 ADMINCNTL** as shown in Figure C-60. Press Enter.

```
HCR7720 ---------- Integrated Cryptographic Service Facility------------------
OPTION ===> 4
Enter the number of the desired option.

   1  COPROCESSOR MGMT - Management of Cryptographic Coprocessors
   2  MASTER KEY       - Master key set or change, CKDS/PKDS Processing
   3  OPSTAT           - Installation options
   4  ADMINCNTL        - Administrative Control Functions
   5  UTILITY          - ICSF Utilities
   6  PPINIT           - Pass Phrase Master Key/CKDS Initialization
   7  TKE              - TKE Master and Operational Key processing
   8  KGUP             - Key Generator Utility processes
   9  UDX MGMT         - Management of User Defined Extensions

      Licensed Materials - Property of IBM
      5694-A01 (C) Copyright IBM Corp. 1989, 2004. All rights reserved.
      US Government Users Restricted Rights - Use, duplication or
      disclosure restricted by GSA ADP Schedule Contract with IBM Corp.


   Press ENTER to go to the selected option.
   Press END   to exit to the previous menu.
```

Figure C-60 ICSF main panel

2. Enable Dynamic CKDS Access by selecting it with an `E` as shown in Figure C-61 on page 616. Press Enter.

```
------------------- ICSF - Administrative Control Functions -- Row 1 to 4 of 4
COMMAND ===>                                                  SCROLL ===> PAGE

         Active CKDS: MYSYS.CSF.TESTPLEX.D060920.CSFCKDS
         Active PKDS: MYSYS.CSF.TESTPLEX.D060920.CSFPKDS


 To change the status of a control, enter the appropriate character
 (E - ENABLE, D - DISABLE) and press ENTER.


   FUNCTION                                              STATUS
   --------                                              ------
e  Dynamic CKDS Access                                   DISABLED
   PKA Callable Services                                 ENABLED
   PKDS Read Access                                      ENABLED
   PKDS Write, Create, and Delete Access                 ENABLED
*******************************Bottom of data ********************************
```

Figure C-61 ICSF Administrative Control Functions panel

Key change verification

Now that the SYM-MK Master Keys have been changed on all LPARs, a sample application, which uses keys protected by the SYM-MK, must be run:

1. Run the sample job, which demonstrates keys are again accessible.
2. Return to “Post-key change: All LPARs in the Sysplex” on page 594 and repeat for each LPAR in the Sysplex.

Post key change

After the SYM-MK has been changed, the key parts must be backed up according to your company’s rules and standards. You can choose to use a USB device to store the key parts.

To save the key parts on a USB device:

1. Connect both the primary and backup USB devices to the computer.

   This opens a Windows Explorer window for each device (see Figure C-62 on page 617).

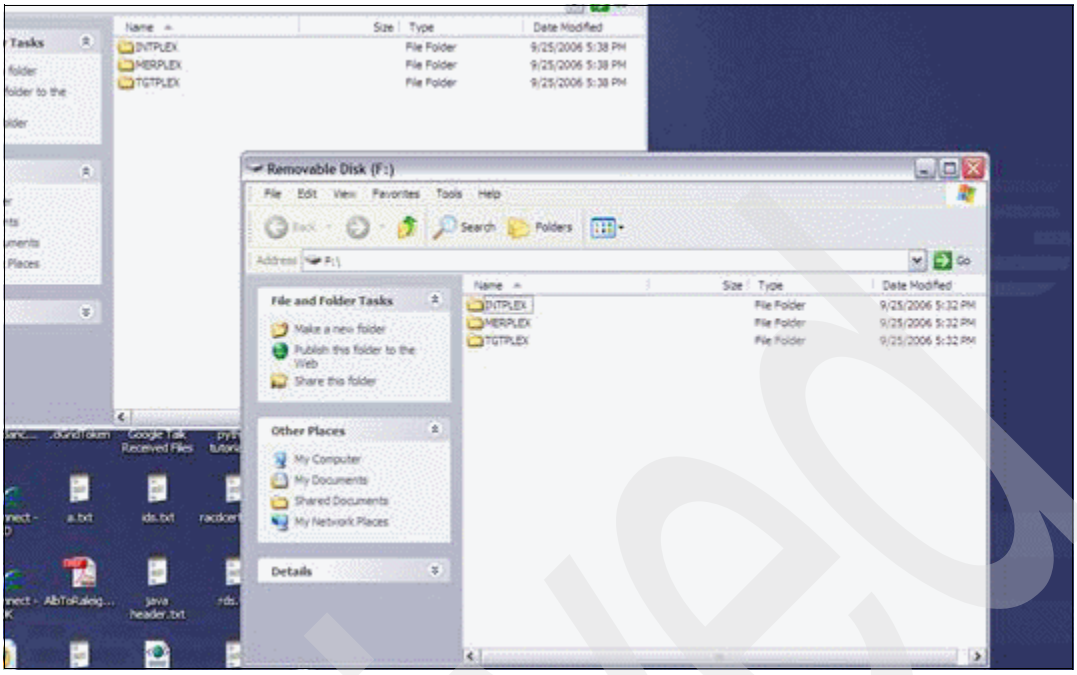


Figure C-62 Windows Explorer windows for the files on the USB devices

2. From the primary USB device window, copy the new key parts from each folder into the corresponding folder in the backup device; see Figure C-63.

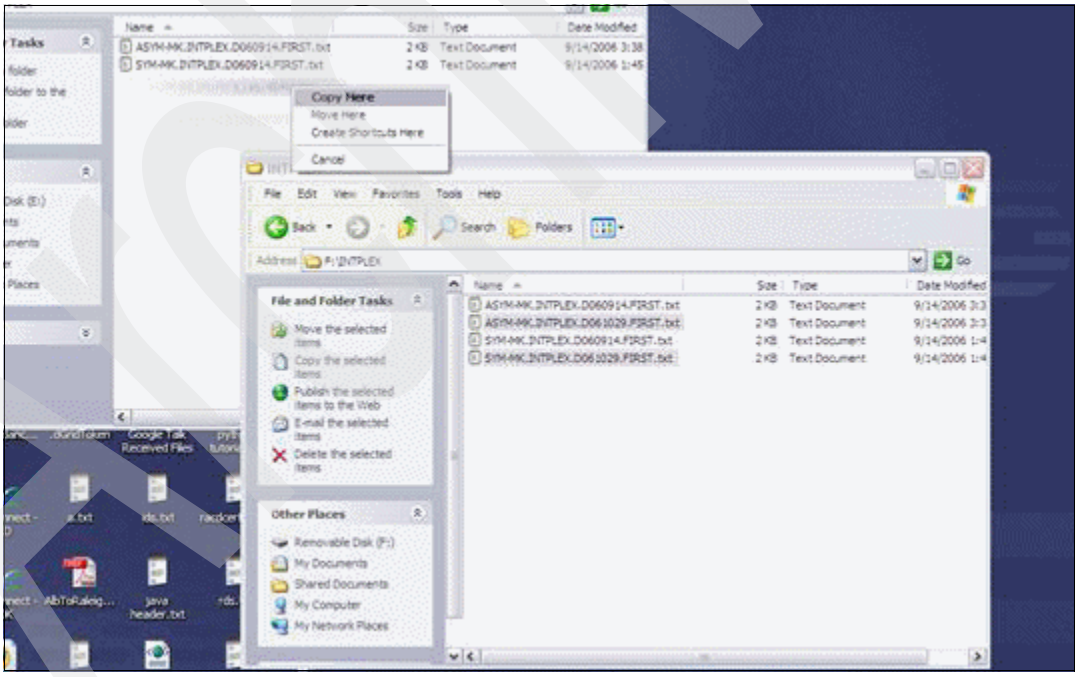


Figure C-63 Copying the new key parts

3. Repeat this process for each folder, so that all new key parts get copied to the backup device.

D

z/OS tape data encryption diagnostics

This appendix is intended to assist you in problem determination if you encounter a tape data encryption error. We discuss error diagnostic scenarios and error codes.

EKM problem determination when running z/OS

When running the EKM on z/OS, the three places to look for errors are:

- The EKM audit log

  Most error messages appear in the audit log. The location and filename are set in the EKM KeyManagerConfig.properties file in the Audit.handler.file.directory and Audit.handler.file.name properties.

- Standard error (stderr)

  When running EKM as a started task using the EKM console wrappers, examine the Execution log for errors. When running EKM in the foreground using UNIX System Services/OMVS, these errors appear where you have directed STDERR, which can simply be the window.

- The EKM debug log

  The location is set in the EKM KeyManagerConfig.properties file debug.output.file property, and the data written to the file is controlled by the debug property. For space reasons, we recommend that you initially set the property to debug=none. If an error is encountered while EKM 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 EKM audit log or Standard Error, we recommend that you set debug=all.

Error scenarios

Errors that you might see when running EKM on z/OS, and possible causes include:

Error one

```
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 (that is, JCE4758RACFKS or JCERACFKS), this error can occur:

- If the user ID running EKM is not the owner of the keyring/private key. RACF only allows a private key to be retrieved by its owner.
- When starting EKM if your keyring has a public key (that does not contain a corresponding private key, such as a business partner's key) *and* that key was not connected as CERTAUTH.

Error two

```
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 can occur if unrestricted policy files were not installed. To check and correct this, regardless of which IBM Software Developer Kit (SDK) version that 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 (EKM) to serve Advanced Encryption Standard (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 Software Developer Kit (SDK) build under the jce demo directory. You only have to copy them to the `lib/security` directory using the following command:

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

Alternatively, you can download the unrestricted policy files from the following Web site:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=jcesdk

Be sure to select the Unrestricted JCE Policy files for SDK 1.4.2, which works for both Java 1.4.2 and Java 5.0 SDKs. Do not select the 1.4.1 version, which is incompatible.

This error usually appears in the EKM audit log.

Error three

```
# 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 might be installed. Note that this error is sent to STDERR (your job execution log) and not the EKM audit log.

Error four

```
***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 that there is a new Java installation or upgrade if you are planning to use the IBM Integrated Cryptographic Service Facility (ICSF) hardware keys.

If you have decided to use a keystore of either of the following types to make use of the security advantages of ICSF, you must add the Java hardware provider:

- JCE4758KS/JCECCAKS
- JCE4758RACFKS/JCECCARACFKS

You might decide that for test purposes, you want to start 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, 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 following examples. 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 in Example D-1.

Example: D-1 Add the IBMJCE4758 provider

```
#
# 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
```

Note: IBMJCECCA is not supported in SDK 1.4.2.

For SDK 5.0 and higher, add the IBMJCECCA provider as you see in Example D-2.

Example: D-2 Add the IBMJCECCA provider

```
#
# 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 about the Java hardware provider, visit this Web site:

http://www.ibm.com/servers/eserver/zseries/software/java/jcecca14.html

Error five

```
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 six

```
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)
```

Possible cause

A CA Certificate is not connected to the keyring (note: At least one CERTAUTH certificate is required, even if all certificates are self-signed). This message might appear only in the debug log and might only occur when you attempt to start the EKM server.

Error seven

```
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 might 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
```

Diagnostic scenarios

The following scenarios are targeted toward encryption failures and are meant to assist you in problem diagnostics. If an error occurs, follow these steps:

1. Determine if an IOS000I message has been issued indicating `Encryption Failure`. See Example D-3.

Example: D-3 IOS000I message example

```
IOS000I 0BD0,60,IOE,01,0E00,,**,JJC046,ATNCMP1 031
804C08C022402751 0001FF0000000000 0005EE2500000092 2004E82061BA2111
ENCRYPTION FAILURE
CU=00 DRIVE=000000 EKM=05EE25
```

Note: The critical piece of a non-zero EKM error code is the last two bytes. In the previous example, this is the EE25 value.

Do the following tasks:

a. If `Encryption Failure` occurred, verify if IOS628E has also been issued. If IOS628E is issued, make sure the complete text of the IOS628E message and the complete text of the IOS000I message are included in the problem record (this problem must be reported to the IOS group - COMPID 5752SC1C3). However, if the IOS628E message indicates either of the following messages, this problem has to be reported through IBM Hardware Support:

- `Encryption Status Not Returned`
- `IO Error`

See Example D-4.

Example: D-4 IOS628E message example

```
IOS628E ENCRYPTION ON DEVICE 1DA1 HAS FAILED
DUE TO CONNECTION FAILURE SOCKET ERNO=0467

IOS000I 1DA1,87,IOE,01,0E00,,**,J12523,DDR1A
804C08C022402751 0301FF0000000000 0000000000000092 2004E82000272011
ENCRYPTION FAILURE
CU = 03 DRIVE = 000000 EKM = 000000
```

Note: If the IOS628E message indicates that a failure occurred connecting to the key manager (as indicated in Example D-4), refer to additional information in this book at “IOS628E message indicates connection failure” on page 630 that pertains to the failing socket error number.

b. If `Encryption Failure` occurred but no IOS628E message, note the following values:

- Non-zero control unit (CU) value:
  Verify which CU is associated with the error, then contact Hardware Support. Provide the complete IOS000I message and also identify the failing hardware so that Hardware Support can dial in to the machine and gather diagnostic information.
- Non-zero drive value:
  Verify which drive is associated with the error and look up the error code reported by the drive (FID message: descriptions are included in “Drive error codes” on page 628) and determined if the problem can be easily resolved. If it cannot be resolved, contact Hardware Support. Provide the complete IOS000I message as well as information about which hardware is having a problem (to allow Hardware Support to gather information from that hardware).
- Non-zero EKM value:
  Look up the reported codes for the EKM (see Table D-1 on page 626) to determine if the problem can be easily resolved and if it cannot, report the problem to the Java Service Group (COMPID - 5648C9800) and provide the key manager audit log and (if available) the key manager debug log. The Java team will then forward the information to the Java Security group.
- If more than one non-zero value, and if the EKM value is non-zero, follow the previous non-zero EKM process; otherwise, follow the previous non-zero CU process.

2. IOS000I message indicates something other than an `Encryption Failure`, verify which hardware is associated with the error. The problem must be reported through IBM Hardware Support. Also, provide the complete IOS000I message output, as well as identify the failing hardware (so IBM Hardware Support can dial in to the machine and gather diagnostic information).

See Example D-5 and Example D-6 for sample messages.

Example: D-5 IOS000I message example: Long Busy Timeout

```
IOS000I 0963,86,IOE,03,0E00,,**,L00500,JOBX
4040C0C60120050  0000000000000000 0000000000000000 0000000000000000
DEVICE HAS EXCEEDED LONG BUSY TIMEOUT
```

Example: D-6 IOS000I message example: Equipment Check

```
IOS000I 0963,86,IOE,03,0E00,,**,L00500,JOBX
000102FF04057007 0000000000000000 0000000000000000 0000000000000000
EQUIPMENT CHECK
```

Or a different failure indicated in line 3 of the IOS000I message.

Depending on the failure, Hardware Support might also ask you to re-create the error and provide a generalized trace facility (GTF) trace with the parameters shown in Example D-7.

Example: D-7 Sample GTF TRACE command

```
TRACE =SIOP,IOP,CCWP,XSCH,CSCH,HSCH
SSCH=IO=(xxxx)                          (where xxxx = device address)
CCW=(SI,CCWN=1024,DATA=256,IOSB)
```

3. If an encryption job is stopped, this problem must initially be handled by the IOS service group (COMPID - 5752SC1C3). The group has to look at the CTRACE records to determine whether the hanging job is related to the new encryption support (new CTRACE records are cut in the trace entries for SYSIOS). See Example D-8.

Example: D-8 Sample DUMP command

```
DUMP COMM=(TAPEENC HANG)
04 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R
04,JOBNAME=(IOSAS),SDATA=(CSA,PSA,SQA,NUC,LSQA,TRT,SUM,SWA,RGN, GRSQ,LPA,NUC)
```

4. If the failure is *none of the above*, ensure that the problem is routed to the correct service area (software or hardware).

Encryption Key Manager error codes and recovery actions

Encryption Key Manager (EKM) error codes and recovery actions are in Table D-1 on page 626. For more details, also refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418, which is available at:

http://www.ibm.com/support/docview.wss?rs=1139&context=STCXRGL&dc=DA400&uid=ssg1S7001885&loc=en_US&cs=utf-8&lang=en

Note: When looking up the EKM error reported in the IOS000I message EKM=*xxyyzz*, focus on the last two bytes *yyzz* for the EKM error number in Table D-1.

Table D-1 Encryption Key Manager error codes

Error number	Description	Recovery action
EE02	Encryption Read Message Failure: DriverErrorNotifyParameterError: "Bad ASC & ASCQ received. ASC & ASCQ do not match with either of Key Creation/Key Translation/Key Acquisition operation."	The tape drive asked for an unsupported action. Ensure that you are running the latest version of the EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug tracing on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
EE0F	Encryption logic error: Internal error: "Unexpected error. Internal programming error in EKM."	Ensure that you are running the latest version of the EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug tracing on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
	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 cannot be parsed because of a general error. Ensure that you are running the latest version of the EKM. Enable debug on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
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 EKM 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 EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug tracing and retry the operation. If the problem persists, contact IBM Disk/Tape SupportLine.
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 EKM. Enable debug on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.

Error number	Description	Recovery action
EE2B	Encryption Read Message Failure: Internal error: "`Either no signature in DSK or signature in DSK cannot be verified`".	Ensure that you are running the latest version of the EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug tracing on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
EE2C	Encryption Read Message Failure: QueryDSKParameterError: "`Error parsing a QueryDSKMessage from a device. Unexpected dsk count or unexpected payload`."	The tape drive asked the EKM to do an unsupported function. Ensure that you are running the latest version of the EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug tracing on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
EE2D	Encryption Read Message Failure: Invalid Message Type	The EKM 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 EKM. Enable debug on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Software Support and ask for SupportLine.
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 EKM. Enable debug on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
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 EKM 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 EKM server to verify whether the drive is correctly configured (for example, the drive serial number, and associated aliases or key labels are correct). If the drive in question has no aliases or key labels associated with it, check the values of default.drive.alias1 and default.drive.alias2. If this does not help or the alias/key label exists, collect debug logs and contact IBM Disk/Tape SupportLine.

Error number	Description	Recovery action
EEE1	Encryption logic error: Internal error: "`Unexpected error: EK/EEDK flags conflict with subpage`."	Ensure that you are running the latest version of the EKM. Check the versions of drive or proxy server firmware and update them to the latest release, if necessary. Enable debug on the key manager server. Try to re-create the problem and gather debug logs. If the problem persists, contact IBM Disk/Tape SupportLine.
EF01	Encryption Configuration Problem: "`Drive not configured`"	The drive that is trying to communicate with the EKM 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 Disk/Tape SupportLine.

Drive error codes

Action plans for drive errors are determined from the `FID`*xx* messages displayed on the drive panel, reported to the host server, and displayed on the 3584 library Web interface to the drive. The new encryption FIDs are `FID5`*x* messages. These are documented in the drive machine interface (MI) along with action plans. The encryption FIDs are listed in Table D-2.

Table D-2 Drive error codes

FID	Description	Comments
50	Encryption Configuration 100% Drive Canister	Encryption Configuration installed during manufacturing is incorrect. Drive canister must be replaced.
51	Encryption Self-Test (POST HW) 100% Drive Canister	Encryption hardware power on self-test (POST) failed. Drive canister must be replaced.
52	Encryption Self-Test (POST FW)	Encryption firmware (FW) power on self-test (POST) failed.
53	Encryption Self-Test (Invoked) 50% Drive Canister; 50% Microcode	An explicitly invoked encryption self-test failed.
54	Encryption Self-Test (Automatic) 80% Drive Canister; 20% Microcode	An automatically invoked encryption diagnostic failed.
55	Encryption Module Failure 80% Drive Canister; 20% Microcode	An unexpected failure of hardware function occurred.
58	Encryption Error 100% Drive Canister	An error was detected during the encryption of data.

FID	Description	Comments
59	Decryption Error 25% Drive Canister; 25% Microcode; 25% EKM; 25% Cartridge	An error was detected during the decryption of data.
5A	Encryption Key Manager (EKM) Failure	An unexpected status was returned by the key manager. Check library/proxy interface, and check EKM log. Not a drive or microcode problem. Requires investigation by client.
5B	Encryption PROXY Failure	A failure or timeout occurred on the proxy interface. Check library/proxy interface, and check EKM log. Not a drive or microcode problem. Requires investigation by client.
5F	Security Prohibited Function	A function was attempted, which is prohibited because of the current security settings.

Control unit error codes

The control unit has been enhanced to report additional errors for Encryption failures. See Table D-3.

Table D-3 Control unit error codes

Error code	Description	Recovery action
00	Not a control unit-reported failure.	Refer to the EKM and drive error codes reported in the IOS000I message for failure information.
01	The key manager was not available for an out-of-band key exchange.	Verify the key manager that the control unit is configured to use and the state of that key manager. However, if the intent was to use in-band key management, use the EKM subcommand of the IECIOS*xx* PARMLIB member or the SETIOS command to specify your key managers.
02	Timeout for an out-of-band key exchange.	The EKM might have gone down mid-sequence, or there might be a network problem. Verify the state of the key manager and the TCP/IP network. However, if the intent was to use in-band key management, use the EKM subcommand of the IECIOS*xx* PARMLIB member or the SETIOS command to specify your key managers.
03	An in-band key exchange was canceled by the host.	Check for an IOS628E message for additional information to learn why the in-band proxy might have canceled the key exchange.

IOS628E message indicates connection failure

A connection failure can be indicated by an IOS628E message as shown in Example D-9.

Example: D-9 Connection failure

```
IOS628E ENCRYPTION ON DEVICE 0540 HAS FAILED DUE
   TO CONNECTION FAILURE SOCKET ERNO=0468
```

Use this link to obtain the table that has the documented error codes:

http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/bpxza870/3.0?ACTION=MATCHES&REQUEST=0467&TYPE=FUZZY&SHELF=&DT=20060615093634&CASE=&searchTopic=TOPIC&searchText=TEXT&searchIndex=INDEX&rank=RANK&ScrollTOP=FIRSTHIT#FIRSTHIT

The table is located in the manual *z/OS V1R8.0 UNIX System Services Messages and Codes,* SA22-7807.

E

IEHINITT exits and messages for rekeying

This appendix provides additional details about the *exits* that support the rekeying function in the IEHINITT program. It also lists new IEHINITT messages that are available with the rekeying support.

Tip: For additional information about IEHINITT REKEY refer to the appendix in *IBM TS3500 Tape Library with System z Attachment A Practical Guide to Enterprise Tape Drives and TS3500 Tape Automation*, SG24-6789.

Dynamic Exits Service Facility support

IEHINITT's REKEY support provides an interface (through the Dynamic Exits Facility) that enables notification of a volume's key labels changes to be received through a user exit. This is of particular importance if the key labels associated with a volume are being recorded. As with IEHINITT's Dynamic Exits support, a single exit (REKEY_EXIT) is predefined to the system, enabling a user-written exit routine to be associated with the REKEY_EXIT, using the Dynamic Exits Services macro, CSVDYNEX.

To add an exit routing, refer to *z/OS MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALESERV-DYNALLOC)*, SA22-7609.

With tape labeling (INITT), a single exit, IEHINITT_EXIT, exists and it has two associated exit points:

- The *Pre-Label exit* is invoked just *before* issuance of the labeling I/O occurs. The intent of this exit point is to allow the installation to indicate whether labeling of the volume is to occur, and, to a degree, the values that are to be used in the labeling.
- The *Post-Label exit* is invoked just *after* issuance of the labeling I/O has occurred. The intent of this exit point is to inform the installation exit routines of the results of the labeling request and associated Pre-Label exit processing.

The Dynamic Exits support for REKEY is provided in a similar fashion as the INITT Post-Label exit processing in which the new REKEY_EXIT will be invoked once, after the rekey I/O successfully completes. All exit routines associated with this exit will always be called for each successful rekey. A new indicator will be set in the parameter list passed to the exit routines to indicate that the exit is called for REKEY. The key labels listed in Example E-1 and their associated encoding mechanisms are passed in the parameter list (IEHUEXIT).

Example: E-1 Key labels and their encoding mechanisms

```
INXKYLB1        Char(64)     First key label
INXKYLB2        Char(64)     Second key label
INXENCD1        Char(1)      Encoding mechanism for the first key label
INXENCD2        Char(1)      Encoding mechanism for the second key label
```

REKEY function code for INXFUNC:

```
INXREKEY        Fixed(8)      Constant(3) Exit Routine is called for REKEY
```

Currently, no requirement exists to return any return or reason codes or any values from the exit routine. All returned values will be ignored and a warning message will be issued.

Error conditions

If a failure occurs during rekey I/O processing, no exit routines are called. If the Dynamic Exits Services fails, no subsequent rekey exit routines are called. Refer to 12.9.2, "IEHINITT enhancements" on page 417 for information about the setup, use, and invocation of exits associated with the IEHINITT utility.

Programming considerations

The following programming considerations apply:

- The exit routines are required to be in AMODE 31.
- The exit routines run in Key 5, authorized state.

- The parameter list passed to the exit routines is not fetch-protected and is put in Key 5 storage. Therefore, the exit routines are given control in Key 5.
- Although the service does not enforce reentrancy, we suggest that the exit routine is reentrant.

Registers on entry

Registers on entry are listed in Table E-1.

Table E-1 Registers at entry

Register	Contents
1	Address of the parameter list, INXPLIST
13	Address of the standard 72 byte save area[a]
14	Return address of the caller
15	Entry point address of the exit routine

a. The 72-byte save-area is provided solely for exit routines that expect to be able to save the registers on entry. Its contents are not used by the Dynamic Exits Service Facility nor by IEHINITT.

Registers on exit

No requirement exists to return any values in registers from the exit routines, so restoring any registers on exit from the exit routine is not necessary.

Return and reason code values

No requirement exists to return any *return* and *reason* code other than the default values of return code 4 and reason code 0 from the exit routines for rekey Dynamic Exits processing. However, if any other value is returned by the exit routine, it is ignored and a warning message is issued. Values are:

INXRC Contains the exit routine return code. It will be set to a value of 4 each time that an exit is called.

INXRSN Contains the exit routine reason code. It will be set to zero.

REKEY messages

This section lists new messages and messages that have been changed for REKEY support.

New messages

The new messages are:

IEH630I REKEY FAILED FOR VOLUME, *VOLSER*

Explanation: Rekey processing failed for volume *VOLSER* for various reasons. Refer to other IEH6*xx*I and IOS*xxxx* messages for additional explanation.

IEH631I ONE KEYLABLX AND KEYENCDX ARE REQUIRED

Explanation: At least one key label and its associated encoding mechanism are required for REKEY function. If only one key label and its encoding mechanism are specified, the same values will be used for the other key label and encoding mechanism.

IEH632I KEYENCDX REQUIRED IF KEYLABL*X* SPECIFIED

Explanation: If the KEYLABL*X* key word is specified, its associated KEYENCD*X* must also be specified and vice versa.

IEH633I KEYLABL*X* REQUIRED IF KEYENCD*X* SPECIFIED

Explanation: If the KEYENCD*X* key word is specified, its associated KEYLABL*X* must also be specified and vice versa.

IEH635I INVALID VALUE FOR KEYENCD*X* KEYWORD – EXPECTED L FOR LABEL OR H FOR HASH

Explanation: Invalid 1-char value specified for the KEYENCDX keyword. Valid values are L for LABEL and H for HASH.

IEH636I CALL TO DFSMSrmm API ENDED WITH RC=*XXX*, RSN=*XXXX*

Explanation: Call to DFSMSrmm API fails for RC=*xxx*, RSN=*xxxx*. The key label information has not been updated in the DFSMSrmm inventory.

IEH637I DEVICE DOES NOT SUPPORT REKEY FEATURE

Explanation: The device does not support rekeying feature or has a downlevel version.

IEH638I TAPE IS NOT ENCRYPTED

Explanation: The mounted tape is not encrypted. Rekey processing is terminated.

IEH639I ERROR ACQUIRING MEDIUM SENSE INFORMATION

Explanation: Error on the medium sense command. Unable to verify whether the tape is encrypted.

IEH640I KEY LABELS WERE NOT SUCCESSFULLY CHANGED

Explanation: The key labels were not successfully changed.

Modified messages

The following messages have been modified:

IEH621I VOLUME *VOLSER*, NOT REKEYED, RACF AUTHORIZATION FAILURE SAF RC: *return_code*, RACF RC: *return_code*, REASON CODE: *reason_code*.

Explanation: Rekeying processing of a volume failed, because the user did not have RACF TAPEVOL access to the volume.

IEH629I CALL TO DYNAMIC EXIT SERVICE CSVDYNEX [FAILED DURING *aaaa* PROCCESSING] RC = *XX*, RSN = *ZZZZ*.

Where *XX* is SPLDEXRC, the dynamic exit service return code, when the CSVDYNEX return code is not 0 or 4.

Where *ZZZZ* is SPLDEXRS, the dynamic exit service reason code, when CSVDYNEX return code is not 0 or 4.

Explanation: Call to the Dynamic Exits Services CSVDYNEX failed for the following processing:

Note: *aaaa* is one of three possibilities:

[FAILED DURING PRE LABEL PROCCESSING] Pre-label dynamic exit processing

[FAILED DURING POST LABEL PROCCESSING] Post-label dynamic exit processing

[FAILED DURING REKEY PROCCESSING] Rekeying dynamic exit processing

F

TS1100 and LTO4 SECURE key EKM on z/OS

In this appendix, we describe how to implement an EKM on z/OS to take advantage of cryptographic hardware enabling us to have SECURE key processing to server keys to both TS1100 and LTO4 drives. This chapter is intended to be a cookbook to help customers get started with a unified keystore, that takes advantage of both the security advantages of having cryptographic hardware and the superior security of z/OS.We will assume in this chapter that the drives and libraries are already setup to take advantage of an EKM running on z/OS. It will also be assumed that the Libraries can get to the z/OS EKM through the TCP/IP network, and that our encryption policies will be using the default keys as set in the EKM configuration file.

Implementing the EKM in z/OS

The Encryption Key Manager (EKM) is a common platform application written in Java that runs under the Java Virtual Machine (JVM). The EKM can also reside outside of the z/OS environment, however for the purposes of this section we have to implement it on z/OS, additionally the z/OS LPAR we implement it on will also need access to ICSF and a CEX2C. On z/OS the EKM runs under UNIX System Services, which is part of the Open MVS (OMVS) address space.

The EKM interfaces with the associated keystores using application programming interfaces (APIs). Secure TCP/IP connections are used to communicate with the tape drives (in-band or out-of-band).

For SECURE symmetric key processing on z/OS, we have to use a JCECCAKS, which also supports SECURE asymmetric key processing, and which enables us to have a keystore that has the highest level of protection for both LTO4 and TS1100 drives.

This hardware-based keystore works with the Integrated Cryptographic Service Facility (ICSF).

We suggest that you use more than one EKM because of redundancy. EKMs can either share keystores or use separate keystores. For this implementation, special key management considerations will be explained in “Enterprise-wide key management” on page 652. For more detailed information about EKMs, refer to the *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide,* GA76-0418, which you can find at:

http://www-01.ibm.com/support/docview.wss?rs=1139&context=STCXRGL&dc=D400&uid=ssg1S4000504&loc=en_US&cs=utf-8&lang=en

Prerequisites

The hwkeytool support for SECURE LTO4 keys in a JCECCAKS is:

- IBM 31-bit SDK for z/OS, J2TE, V5.0, (z/OS 1.6 and z/OS.e 1.6 and later only)
- Order 5655-N98 (at the SDK 1.5 SR5 level or later)
- ICSF, which must be up and running on the LPAR that will serve as the EKMs home

z/OS UNIX System Services

As we mentioned, on z/OS the EKM runs under UNIX System Services, which is part of the Open MVS (OMVS) address space.

The System Control Program (SCP) of z/OS contains address spaces for general Multiple Virtual Storage (MVS) and address spaces for open MVS, which is also called UNIX System Services. When a Time Sharing Option (TSO) user logs on to the system and tries to use UNIX System Services, the appropriated UNIX System Services address space will be created.

It is assumed that the UNIX System Services address space is already present. Refer to *UNIX System Services z/OS Version 1 Release 7 Implementation,* SG24-7035, for more information about how to install and tailor UNIX System Services in z/OS.

To ensure that an UNIX System Services address space is up and running, use the **/D A,L** MVS command as shown in Figure F-1 on page 639.

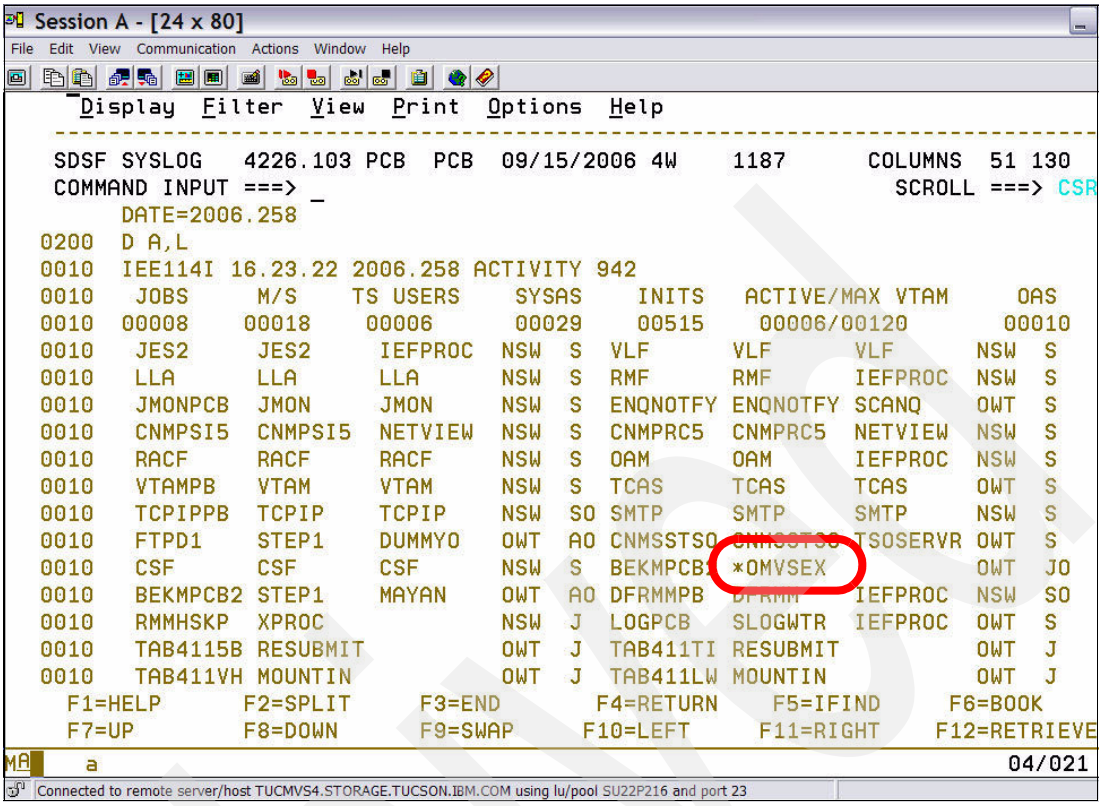


Figure F-1 The OMVSEX address space is present

If the UNIX System Services address space is present, the OMVSEX for OMVS is shown.

Installing the Encryption Key Manager in z/OS

The Encryption Key Manager (EKM) is automatically installed as part of the IBM Java Developer Kit (JDK). However the EKM and IBM JDK are on differing development streams. To ensure that the latest EKM program code is installed, the EKM must be downloaded and copied to the JAVA SDK directory.

To download the latest version of the IBM EKM, go to:

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

You can download the IBM 31-bit Software Developer Kit (SDK) for z/OS Java 2 Technology Edition from the following Web site:

`http://www.ibm.com/servers/eserver/zseries/software/java/j5pcont31.html`

> **Note:** Before you can download the latest version of the IBM SDK, you must register your company or yourself to give IBM statistics for a comparative look at which SDKs are being downloaded.

In ISPF or TSO, use option **6**, (Commands), type `OMVS`, and then press Enter, or use **`telnet/rlogin,ssh`** into an OMVS session.

Use a File Transfer Protocol (FTP) program and copy the file into file system. Then, extract the file into `/usr/lpp/java` using the following command:

```
pax -rf UK17593.PAX.Z
```

TIP: Installing JAVA with SMP/E also installs JZOS. If you download and install Java through this process, JZOS must be manually installed.

To verify that the correct version of the EKM was installed, use the following command at the OMVS command prompt:

```
/u/st6t25c>java -version
```

The information shown in Example F-1 appears.

Example: F-1 Output from java -version

```
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build pmz31devifx-20060302 (SR
1))
IBM J9 VM (build 2.3, J2RE 1.5.0 IBM J9 2.3 z/OS s390-31 j9vmmz3123-20060223 (JI
T enabled)
J9VM - 20060220_05389_bHdSMr
JIT  - 20060220_2133_r8
GC   - 20060214_AA)
JCL  - 20060301
```

Download and install the latest EKM server code. Obtain the code and documentation from:

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

After you have downloaded the EKM code, place it in `$JAVA_HOME/lib/ext`. If your environment variables are set up correctly and the EKM program code is in the working directory, simply enter the following command to copy it:

```
cp IBMKeymanagerServer.jar $JAVA_HOME/lib/ext
```

At this point, the installation of the EKM for z/OS is already complete.

Note: You cannot start the EKM without an associated keystore, and a rudimentary `KeyManagerConfig.properties` file.

The *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User's Guide,* GA76-0418, provides additional information.

To perform the installation steps necessary to use EKM, special authorization rights are required; for example, the RACDCERT command must be enabled for the OMVS segment. We also require access to the crypto hardware through the CSFSERV and CSFKEYs classes. In Chapter 8, "EKM operational considerations" on page 275, you can read about types of keystores and their usage. In the following sections, we create a hardware keystore.

EKM does not provide cryptographic capabilities, and therefore, it does not require, nor is it allowed to obtain, FIPS 140-2 certification. However, EKM 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, EKM will use the IBMJCEFIPS provider for all cryptographic functions.

In addition to setting the FIPS parameter in the configuration properties file, the following provider must be added to the `java.security` file, just before the IBMJCE provider:

```
$JAVA_HOME/lib/security/security.provider.?=com.ibm.crypto.fips.provider.IBMJCEFIPS
```

Renumber the providers accordingly.

Note: We do not recommend that you use hardware-based keystore types with FIPS.

Create a JCECCAKS for EKM

In this section, we create a JCECCAKS, which we will later use as the keystore for EKM initialization. This is the most secure of the keystores that use symmetric keys. Access to the keystore file is protected by UNIX System Services file permissions; access to use the crypto hardware is through RACF administration. Our symmetric keys themselves are stored in the Cryptographic Key Data Set (CKDS), which is itself protected by cryptographic hardware.

Refer to Chapter 7, “Planning and managing your keys” on page 227 for a more detailed discussion of this keystore type.

The following hwkeytool commands in Example F-2 will create a keystore with five triple DES (or TDES) 192-bit symmetric keys in it. This means that our JCECCAKS file will contain labels that point to the master key encrypted entries in the CKDS. These are the keys that will be used to protect data on LTO4 cartridges.

Example: F-2 JCECCAKS with symmetric keys in the CKDS

```
hwkeytool -genseckey -v \
-aliasRange EKM2008101-2008105 \
-keypass "CLEARTEXTPASS" \
-keyalg TDES \
-keysize 192 \
-hardwaretype CKDS \
-keystore ekmkeys.cca \
-storepass "CLEARTEXTPASS" \
-storetype JCECCAKS \
-provider IBMJCECCA
```

Tip: These hwkeytool commands are listed in these examples so that you can cut and paste them directly into a shell script and run the script. To run these commands from the OMVS command line you would remove the trailing backslashes (\) and type the whole command on a line. We have put them in the example in a script format for readability and usability reasons.

To serve keys to TS1100 tape drives, we create an x.509 certificate and key-pair in our keystore. Example 15-4 is the **hwkeytool** command that creates the key-pair in the keystore. It generates a self-signed certificate with a validity of one year. This certificate could be later signed by a third-party certificate authority, but for the purposes of this description, that exercise is left to the reader.

Example 15-4 X.509 Asymmetric key creation

```
hwkeytool -genkey -alias "Company.TEKM.20081210" \
-keyalg RSA \
-keysize 2048 \
-dname "cn=Company,c=US" \
-keypass "CLEARTEXTPASS" \
-storepass "CLEARTEXTPASS" \
-keystore ekmkeys.cca \
-storetype JCECCAKS \
-provider IBMJCECCA \
-hardwaretype PKDS
```

For our EKM, we should also create a keystore for our SSL certificates to enable our EKMs communicate over an SSL connection. In Example F-3, we add our SSL certificate to the keystore that was created in Example F-2 on page 641. This command stores our key material in the Public Key Data Set (PKDS). This SSL certificate is valid for 10 years from today, with a key size of 2048 bits.

Example: F-3 Creating SSL certificate in a JCECCAKS

```
hwkeytool -genkey -alias "SSL.TEKM.20181210" \
-keyalg RSA \
-keysize 2048
-validity 3650 \
-dname "cn=Company,c=US" \
-keypass "CLEARTEXTPASS" \
-storepass "CLEARTEXTPASS" \
-keystore ekmkeys.cca \
-storetype JCECCAKS \
-provider IBMJCECCA \
-hardwaretype PKDS
```

Setting up the EKM environment

For this example, we assume that the user ID EKM is going to run on EKMSERV and that the UNIX System Services home directory for this user is `/u/ekmserv`. To create this user, issue the following command:

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

This command creates the user ID EKMSERV, sets up its home directory to `/u/ekmserv`, sets the default shell to `/bin/sh`, and associates it with the next available UID. If a more strict security policy is in effect, the RACF administrator must replace the AUTOUID with UID (*UserIDNumber*). Avoid using UID `0` (zero).

In addition, this user will have access to the IRR.DIGTCERT.* facility classes as described in Example 6-4 on page 164 in 6.1.4, “Create a JCE4758RACFKS for EKM” on page 164.

Before continuing, download and save the following files to /u/ekmserv/:

- IBM EKM application
- IBM EKM sample configuration file
- JZOSEKM files for z/OS batch

You can find the application and files at the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=0&dc=D400&q1=ekm&uid=ssg1S4000504&loc=en_US&cs=utf-8&cc=us&lang=en

We first have to ensure that the ekmserv ID has an acceptable .profile file. In /u/ekmserv, edit the .profile file to make it look like the sample profile shown in Example F-4.

In the example, include the line stty quit ^V in a profile when logging on to OMVS using Secure Shell (SSH). The line export PS1 is setting up this user's command line. In this case, we are setting up the command line to always list the current working directory. This is a useful setting to have when navigating around an OMVS file system.

After that, we are setting up the JAVA_HOME environment variable. This variable must point to the Java home on your system. Next, we add JAVA_HOME/bin to our path in addition to other useful directories. Notice that we are careful to keep our original path here by appending it to our new PATH.

Now, we are creating environmental shorthand. When using OMVS as the shell, and not logging on using Telnet, rlogin, or SSH, we have a limited length command line. The arguments to start EKM and perform Java **hwkeytool** commands might be longer than the command line; by setting some of these long parameters as environment variables, we can save space on our limited command line.

Example: F-4 Sample .profile

```
stty quit ^V
  export PS1='$PWD>';
  export JAVA_HOME=/u/java/J1.4
export PATH=.:${JAVA_HOME}/bin:/usr/sbin:$PATH
alias hw="hwkeytool -debug -list -storetype JCECCAKS -keystore"
export keyFile=KeyManagerConfig.properties
```

After our profile is set up, we can reread it by either of the following methods:

- Type the following line:

  ```
  . ./.profile
  ```

- Log out and log back in, which executes the .profile again.

Now, we must copy the updated EKM:

```
cp IBMKeyManagementServer.jar $JAVA_HOME/lib/ext
```

This command copies the EKM program code into JAVA_HOME/lib/ext. The JARs in the lib/ext directory are all loaded into the JVM's class path. A simple listing of that directory reveals security provider JAR files among other things.

The JZOS EKM files must be extracted. Enter the command:

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

This command extracts the following files:

- PROCLIB.EKM2ENV
- README
- jzosekm.jar
- PROCLIB.EKM2

The file jzosekm.jar contains Java wrapper code for the EKM. To interact with the EKM using write to operator (WTO), we must register a callback using APIs from the JZOS toolkit. The program code contained in this JAR does that for us. To ensure that this code is in our class path, we copy it to lib/ext:

```
cp jzosekm.jar $JAVA_HOME/lib/ext
```

The EKM configuration file `KeyManagerConfig.properties` is now edited with our information. In Example F-5, we have turned on extra tracing and debugging information. All of our keystore and directory structure properties in the configuration file are set to relative file paths.

Example: F-5 Sample EKM config

```
TransportListener.ssl.port = 4430
TransportListener.tcp.port = 38010
fips = Off
Admin.ssl.keystore.name = ekmkeys.cca
config.keystore.provider = IBMJCECCA
Admin.ssl.truststore.password = CLEARTEXTPASS
TransportListener.ssl.clientauthentication = 0
config.keystore.password = CLEARTEXTPASS
TransportListener.ssl.ciphersuites = JSSE_ALL
Audit.handler.file.size = 500
zOSCompatibility = true
drive.acceptUnknownDrives = true
TransportListener.ssl.truststore.name = ekmkeys.cca
Audit.handler.file.directory = keymanager/audit
TransportListener.ssl.protocols = SSL_TLS
debug.output = simple_file
TransportListener.ssl.truststore.type = JCECCAKS
config.keystore.file = ekmkeys.cca
TransportListener.ssl.keystore.name = ekmkeys.cca
TransportListener.ssl.keystore.password = CLEARTEXTPASS
TransportListener.ssl.truststore.password = CLEARTEXTPASS
Audit.event.outcome.do = success,failure
Audit.eventQueue.max = 0
debug.output.file = keymanager/debug/ekmdebug.txt
Audit.handler.file.name = ekm.audit.log
TransportListener.ssl.keystore.type = jceccaks
config.keystore.type = jceccaks
Audit.event.types.backup = authentication,authorization,data synchronization,runtime,audit
management,authorization terminate,configuration management,resource management,none
Audit.event.outcome = success,failure
debug = all
Audit.event.types = all
Admin.ssl.truststore.name = ekmkeys.cca
config.drivetable.file.url = FILE:////keymanager/drivetab
symmetricKeySet = EKM2008101-2008105
default.drive.alias2 = Company.TEKM.20081210
default.drive.alias1 = Company.TEKM.20081210
Admin.ssl.keystore.password = CLEARTEXTPASS
```

At this point, we can create the directories the EKM will use, or let the EKM create the directories on startup. Allow the EKM to create its directories, which will let the EKM set itself as owner, and will also avoid any typing errors when creating the directories. The `debug.output.file` option points to a file. If the assumption is made that the option points to a directory, `java.io.permission` errors can occur. If the file does not exist, EKM creates it, but the file cannot point to a directory.

Note: Our configuration file has the following information:
`zOSCompatibility = true`
`This is because of the way the symmetric keys are stored in the CKDS. Thus option must be set to use SECURE symmetric keys.`

Starting EKM

In this section, we discuss how to perform interactions with an EKM as a started task. We also discuss how to set up the EKM to run as a *started task* using JZOS.

Starting EKM with JZOS as a started task

A z/OS *started task* is a procedure that 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 it can be associated with a functional subsystem.

We suggest that the EKM run as a started task on z/OS using the JZOS batch launcher, which is shipped as part of the z/OS Java product. Refer to “EKM and JZOS” on page 567 for more information.

To define the EKM as a started procedure, update the started class table with the z/OS user ID of the EKM. Previously in this section, we created EKMSERV as the user ID to be used with the EKM and the group associated with that started procedure will be SYS1. Details of RACF processing and the definition of started procedures are discussed in the *z/OS Security Server RACF Security Administrator’s Guide*, SA22-7683.

To set up the STARTED class, enter the commands in Example F-6.

Example: F-6 Define EKM as a started task

```
SETROPTS  GENERIC(STARTED)
RDEFINE  STARTED  EKM*.*  STDATA(USER(EKMSERV)  GROUP(STCGROUP)  TRACE(YES))
SETROPTS CLASSACT(STARTED) SETROPTS RACLIST(STARTED)
```

The JZOSEKMFiles.pax.Z file downloaded in the beginning of this section consists of `jzosekm.jar`, sample JCL, the STDENV file for the sample JCL, and a EKMConsoleWrapper. Refer to the readme file that explains where each file must located and any specific installation and tailoring tasks that might be required.

To extract the contents of the download file, issue the following command:

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

Place the `jzosekm.jar` file into `$JAVA_HOME/J1.4/lib/ext`.

The EKM can create audit records, which wrap the log to three files. When the last file is full, the first file is rewritten. On z/OS, you have to allocate file system space for the EKM audit logs, and possibly the EKM debug log.

You may choose to allocate a file system specifically for use by the EKM for audit and debug file storage. Assume 500 cylinders of space to allocate to the EKM's audit and debug log file system until you have observed, based on tape and EKM activity, how quickly the audit logs wrap. The file system must not be shared by the EKM instances running in a Sysplex environment, but the files must be private to each EKM instance. This setup prevents any possible interleaving of audit or debug logs between EKM instances.

Mount the ekmlogs file system and create a directory for each system on which EKM will run. For example, the two file systems created can be ekmlogs with JA0 and JB0 for two system names of two images within a Sysplex:

- /ekmlogs/JA0
- /ekmlogs/JB0

Create a new partitioned data set (PDS) to contain the STDENV environment variables. In this example, a PDS was allocated with the name EKMSERV.ENCRYPT.CONFIG that has the following attributes:

Organization	PO
Record format	FB
Record length	80
Block size	6160
First extent cylinders	3
Secondary cylinders	1

Create a member in EKMSERV.ENCRYPT.CONFIG named EKM2ENV. Create or edit the shell script contents as shown in Example F-7.

Example: F-7 EKM environment script

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

#Set  these  variables  to  the  installation  unique  values
#  EKM_HOME  =  directory  from where  EKM  runs
#  JAVA_HOME  =  directory  where  Java  is  mounted
#Update to point to your EKM Home directory
export  EKM_HOME="/u/ekmserv"
#Update to point to your JDK
export  JAVA_HOME="/u/java/J1.5"
export  PATH=/bin:"${JAVA_HOME}"/bin:

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

export  LIBPATH="$LIBPATH":

#  Customize  your  CLASSPATH  here
CLASSPATH=${JAVA_HOME}/lib
CLASSPATH=$CLASSPATH:"${EKM_HOME}"

export  CLASSPATH="$CLASSPATH":

#  Set  JZOS  specific  options
#Update  with  location  of  config  data
export  keyFile="KeyManagerConfig.properties"
#The EKM main class
```

```
export  ekmClass="com.ibm.keymanager.KMSAdminCmd"
export  JZOS_MAIN_ARGS="$ekmClass $keyFile"

#  Configure  JVM  options  (if  any)
#Uncomment this only for JCERACFKS
#IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwr.provider"
#Uncomment this only for JCE4758RACFKS
#IJO="-Djava.protocol.handler.pkgs=com.ibm.crypto.hdwrCCA.provider"
#  for  jceks  and  jce4758ks/jceccaks,  no  IJO  definition  is  required.
#  comment them out

export  IBM_JAVA_OPTIONS="$IJO  "

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

Customize the sample procedure in Example F-8 for your system.

Example: F-8 Start procedure

```
//EKM  PROC  JAVACLS='com.ibm.jzosekm.EKMConsoleWrapper',
//    ARGS=,                                 <  Args  to  Java  class
//    LIBRARY='SYS1.SIEALNKE',               <  STEPLIB  FOR  JVMLDM  module
//    VERSION='15',                          <  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=EKMSERV.ENCRYPT.CONFIG(EKM2ENV),DISP=SHR
//*
```

Starting EKM with a command

The EKM process can now be started with the operator start command as a started task. The following command starts the EKM server:

```
S EKMSERV
```

In Figure F-2, we have started the EKM as a job using JZOS. We can see the `Loaded drive keystore` message and that EKM is running and communicating with the console.

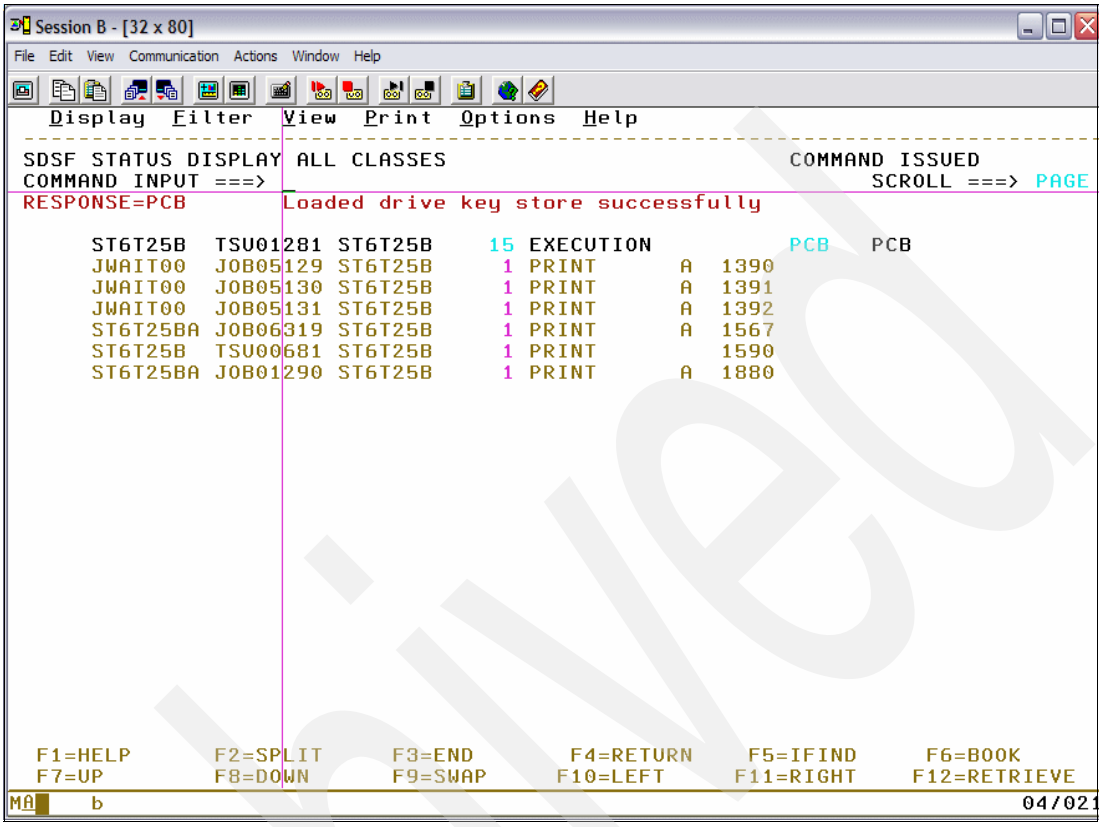


Figure F-2 EKM start message

If we execute the following command, the message in Figure F-3 on page 649 displays:

```
f EKMSERV,appl='status'
```

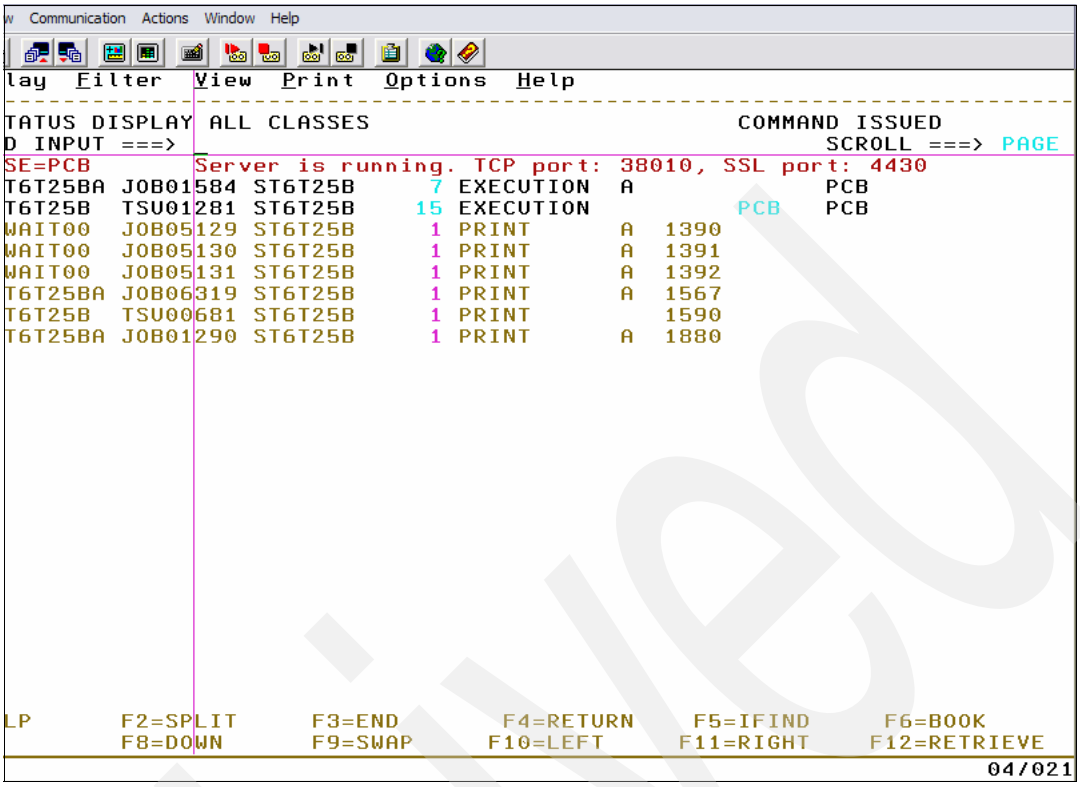


Figure F-3 EKM status

We can also list how many certificates were loaded into the EKM by executing:

```
f EKMSERV,appl='listcerts'
```

The output from this command is shown in Figure F-4 on page 650.

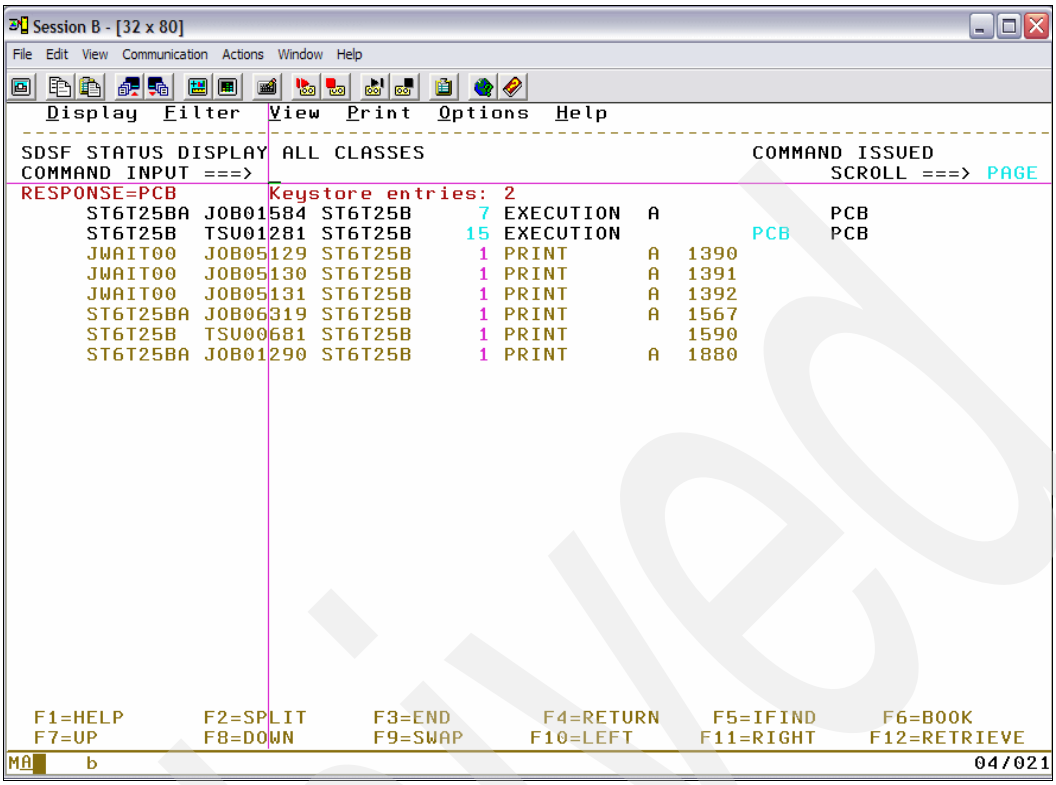


Figure F-4 EKM certificate list

For a complete listing of EKM commands, refer to 8.1, “EKM commands” on page 276.

Note: In this example, note that our EKM is running its SSL listener on port 4430. WebSphere might use 4430 for SSL processing, so we changed the port so EKM does not collide with WebSphere SSL processing.

EKM TCP/IP configuration

Most often, the definitions described in 6.1.8, “Virtual IP Addressing” on page 175 are sufficient. However, if you have a large environment and want to use automatic backup solutions for EKM in a complex Sysplex environment, the hints appended to the commands can help give you an idea of how to define a solution. TCP/IP experienced and trained personnel will be able to define this type of structure.

The entire TCP/IP configuration, including VIPAs, is done in the TCP/IP profile. For all images that are running the EKM, the following items are required in the profile:

- Ports 3801 and 1443 must be reserved in the PORT section.

  PORT
  1443 TCP EKM ;Key Manager SSL
  3801 TCP EKM ;Key Manager

- If a port is already being used, the option SHAREOPT must be added to each of the applications using the port. For example:

  PORT
  1443 TCP EKM SHAREOPT ;Key Manager
  1443 TCP IMWEB SHAREOPT ;Web Server

- EKM can also be started when TCP/IP starts up by adding it to the AUTOLOG section. For example:

  AUTOLOG
  EKM ;Key Manager
  ENDAUTOLOG

Sysplex Distributor

The following description is for a 10-way Sysplex with EKM running on most of the images. A Sysplex Distributor was configured using distributed VIPAs. The *Sysplex Distributor* distributes each key request based on the Workload Manager (WLM), if configured that way, to the appropriate stack/image. In case of a stack failure, WLM will move the Sysplex Distributor to another image where the requests will be handled. The backup stack has to be configured in the TCP/IP profile.

To set up a Sysplex Distributor:

1. Ensure you have a dynamic cross-system coupling facility (XCF) component. This is the path that is used for the requests through the Sysplex Distributor:

   ```
   IPCONFIG DYNAMICXCF 192.168.49.30 255.255.255.03
   ```

   This must be done on all images where the Sysplex Distributor might be the backup.

2. Define the required DVIPA parameters in the VIPADYNAMIC section of the profile:

   ```
   VIPARANGE DEFINE 255.255.255.255 9.xxx.xxx.xxx;keystore
   ```

3. Define where the requests will be distributed that are sent to the DVIPA 9.*xxx.xxx.xxx*. This definition might look like the following example:

   ```
   VIPADEFINE MOVEABLE IMMED 255.255.255.0 9.xxx.xxx.xxx
   VIPADISTRIBUTE DEFINE SYSPLEX DISTM ROUNDROBIN 9.xxxxxxxxx PORT 3801 1443
   DESTIP ALL
   ENDVIPADYNAMIC
   ```

   The described definitions tell you that all requests that come to DVIPA address 9.*xxx.xxx.xxx* Port 3801 or Port 1443 are to be sent to all images in the sysplex that are configured to accept requests (dynamic XCF setup).

4. For the images that were configured as backup, you need dynamic XCF EKM running and the following example, which tells the sysplex which DVIPAs you are backing up:

   ```
   VIPADYNAMIC
   VIPABACKUP 9.xxx.xxx.xxx
   ENDVIPADYNAMIC
   ```

Example F-9 shows a complete definition example.

Example: F-9 Definition example

```
IPCONFIG SYSPLEXROUTING DYNAMICXCF 193.9.200.4 255.255.255.240.1
VIPADYNAMIC
VIPADEFINE 255.255.255.192 9.67.240.02
VIPADISTRIBUTE DEFINE 9.67.240.02 PORT 3801 1443 DESTIP
193.9.200.2
193.9.200.4
193.9.200.6
ENDVIPADYNAMIC
```

Enterprise-wide key management

A common theme with improving security of a solution is a decrease in accessibility of the solution. For tape encryption solutions it is suggested to have more than one EKMs, all with the same keys in their keystores to enable EKM to EKM failover.

The best way to keep the keys used by the EKM implementation described in this appendix is to have your EKMs on LPARs in a sysplex with a shared CKDS. This means that anytime you create keys in the CKDS using **`hwkeytool`**, all LPARs in the sysplex that have the same master keys, and point to the same CKDS will have the ability to access the keys. The JCECCAKS file itself can be copied, because it is simply pointing to the key labels in the CKDS, but you will need those keys and key labels in the CKDS for that to work.

To share keys between sites it becomes a bit more complex. If the CKDS is to contain all the same keys site to site, you would simply have to create the keys in one CKDS and copy the whole CKDS to the other site, making sure that both sites are using the same master keys. Then, you may copy the JCECCAKS file to the second site and the EKM will be able to load the same keys.

Things get a bit more complicated if you have CKDSes across different sites that are not going to have the same keys in them. One way to keep the keys synchronized between CKDSes is to use the ICSF Key Generation Utility Program (KGUP) to export the CKDS key protected by a transport key from one CKDS to another CKDS. This is described in The *z/OS Cryptographic Services Integrated Cryptographic Service Facility Administrator's Guide,* SA22-7521-09.

After CKDS keys are moved from one CKDS to another you should then be able to copy the JCECCAKS file to the other system, start the EKM, and have the same keys between both.

To share our asymmetric keys, a similar situation is in place. The best solution when using the hwkeytool commands to create a X.509 certificate and keypair in the PKDS the -keylabel option should be used. This keylabel option lets the user specify the label the private key material will get in the PKDS. Then we can copy out the private key material from one PKDS to another PKDS, assuming they have the same master keys, using the key transfer (keyxfer) utility. Here is the link to keyxfer:

`ftp://ftp.software.ibm.com/s390/zos/tools/keyxfer/keyxfer.rexx.txt`

After the private keys are copied, the JCECCAKS file can be copied to the system and then the EKM can be started.

If the destination system will have the exact same PKDS, this manual process can be avoided. You may simply copy the entire PKDS and CKDS to the destination system, and then move the JCECCAKS file, and start the EKM. For more programing examples of how to move key material around programmatically, refer to *Java Security on z/OS - The Complete View*, SG24-7610.

Conclusions

This appendix is intended as a cookbook to get an EKM up and running on z/OS with a JCECCAKS keystore, using both SECURE key types, asymmetric keys for the TS1100 drives and symmetric keys stored in a CKDS to handle LTO4 tape drives. This type of EKM implementation takes advantage of the improved security of both using cryptographic hardware for protecting key material, and putting the EKM on z/OS to take advantage of all the security benefits for running on that platform.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 655. Note that some of the documents referenced here might be available in softcopy only:

- *IBM TS3500 Tape Library with System z Attachment: A Practical Guide to TS1120 Tape Drives and TS3500 Tape Automation*, SG24-6789
- *IBM TotalStorage 3494 Tape Library: A Practical Guide to Enterprise-Class Tape Drives and Tape Automation,* SG24-4632
- *IBM System Storage Tape Library Guide for Open Systems*, SG24-5946
- *Communications Server for z/OS V1R7 TCP/IP, Implementation Volume 3 - High Availability, Scalability, and Performance,* SG24-7171
- *Java Stand-alone Applications on z/OS Volume II,* SG24-7291
- *Communications Server for z/OS V1R2 TCP/IP Implementation Guide*, SG24-5227
- *IBM System Storage Virtualization Engine TS7700: Tape Virtualization for System z Servers*, SG24-7312
- *UNIX System Services z/OS Version 1 Release 7 Implementation,* SG24-7035

Other publications

These publications are also relevant as further information sources:

- *IBM Encryption Key Manager component for the Java platform, EKM Introduction, Planning, and User’s Guide,* GA76-0418
- *z/OS V1R8.0 UNIX System Services Messages and Codes,* SA22-7807
- *IBM Ported Tools for z/OS User’s Guide*, SA22-7985
- *z/OS Security Server RACF Security Administrator’s Guide*, SA22-7683
- *z/OS V1R8.0 Security Server RACF System Programmer’s Guide,* SA22-7681
- *IBM System Storage TS1120 and TS1130 Tape Drives and TS1120 Controller Introduction and Planning Guide 3592 Models J1A, E05, E06, EU6, J70 and C06*, GA32-0555.
- *IBM System Storage TS3500 Tape Library Operator Guide,* GA32-0560
- *IBM System Storage 3953 Library Manager Model L05 Operator Guide*, GA32-0448
- *z/OS V1R3.0-V1R4.0 DFSMS Using Magnetic Tapes*, SC26-7412-01
- *DFSMSdfp Utilities*, SC26-7414-02

- *z/OS DFSMS Software Support for IBM System Storage TS1120 Tape Drive (3592),* SC26-7514
- *z/OS V1R3.0-V1R8.0 DFSMS Software Support for IBM TotalStorage Enterprise Tape System 3592*, SC26-7514
- *z/OS V1R7.0 MVS Initialization and Tuning Guide,* SA22-7591
- *IBM Tivoli Storage Manager for AIX Administrator's Guide,* GC32-0768
- *z/OS V1R7.0 MVS Initialization and Tuning Reference,* SA22-7592
- *z/OS V1R7.0 MVS System Commands,* SA22-7627
- *z/OS V1R7.0 DFSMSdfp Storage Administration Reference*, SC26-7402
- *z/OS UNIX System Services Planning*, GA22-7800
- *IBM Tape Device Driver Installation and User's Guide*, GC27-2130

Online resources

These Web sites are also relevant as further information sources:

- Privacy Rights Clearing House

  `http://www.Privacyrights.org`
- Electronic Privacy Information Center

  `http://www.epic.org/privacy/bill_track.html`
- IBM Encryption Key Manager Home Page

  `http://www.ibm.com/support/docview.wss?uid=ssg1S4000504`
- IBM Home Page

  `http://www.ibm.com`
- BMJCEFIPS provider and its selection and use

  `http://www.ibm.com/developerworks/java/jdk/security/50/FIPShowto.html`
- *z/OS Security Server RACF Command Language Reference*

  `http://publibz.boulder.ibm.com/epubs/pdf/ichza460.pdf`
- Tivoli Storage Manager documentation

  `http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/index.jsp`
- TS1120 Interoperability matrix

  `http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_interop.pdf`
- TS1120 ISV matrix

  `http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts1120_isv_matrix.pdf`
- TS1120 supported host bus adapters

  `http://www.ibm.com/systems/support/storage/config/hba/index.wss`
- Linear Tape-Open organization home page

  `http://www.lto.org`
- IBM LTO Home Page

  `http://www.ibm.com/storage/lto`

- TS3500 Interoperability matrix

  http://www.ibm.com/systems/storage/tape/pdf/compatibility/ts3500_interop.pdf
- IBM System Storage Interoperation Center

  http://www.ibm.com/systems/support/storage/config/ssic/
- TS3500 firmware upgrades

  http://www.ibm.com/servers/storage/support/lto/3584/downloading.html
- IBM Virtualization Engine TS7700 Series Encryption Overview

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101000
- IBM Software Developer Kit

  http://www.ibm.com/servers/eserver/zseries/software/java

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Redbooks

IBM System Storage Tape Encryption Solutions

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM System Storage Tape Encryption Solutions

Learn about IBM TS1130 Tape Drive and Tivoli Key Lifecycle Manager

Encrypt data on LTO and TS1100 series cartridges

Discover usability enhancements

This IBM Redbooks publication gives a comprehensive overview of the IBM System Storage Tape Encryption solutions that started with the TS1120 Tape Drive in 2006 and have been made available in the TS7700 Virtualization Engine in early 2007. Also in 2007, the IBM Ultrium Linear Tape-Open (LTO) Generation 4 Tape Drive was announced including its support for tape data encryption. In 2008, additional enhancements to the tape drives that support encryption and to key management have been made. This edition of the book has been updated with information about the TS1130 Tape Drive and the IBM Tivoli Key Lifecycle Manager (TKLM).

This publication is intended for System Programmers, Storage Administrators, Hardware and Software Planners, and other IT personnel involved in planning, implementing, and operating IBM tape data encryption solutions, and anyone seeking details about tape encryption.

This book also provides practical guidance for how to implement an enterprise-wide encryption solution. We describe the general concepts of encryption and the implementation options that are available when using IBM Tape to encrypt tape data. We explain the key management options, including the Encryption Key Manager, which is a Java application that allows for enterprise-wide keystores and key management across a wide variety of platforms. We also provide detailed information for planning, implementation, and operation of tape data encryption for IBM z/OS and Open Systems hosts.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7320-02 ISBN 0738432733