

Encryption Facility for z/OS



User's Guide

Version 1 Release 2

Note

Before using this information and the product it supports, read the information in "Notices" on page 69.

This edition applies to version 1, release 2 of IBM Encryption Facility for z/OS (product number 5655-P97) and to all subsequent releases and modifications until otherwise indicated in new editions. This first edition contains information previously from SA23-1349-02.

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

© **Copyright IBM Corporation 2007, 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures	v
About This Book	vii
Who should read this book	vii
How to use this book	vii
Where to find more information	viii
Related publications	viii
Other sources of information.	ix
IBM discussion area.	ix
Internet sources	ix
How to send your comments to IBM	xi
If you have a technical problem.	xi
Summary of changes	xiii
Chapter 1. Overview of IBM Encryption Facility for z/OS	1
What is Encryption Facility?	1
Features available with Encryption Facility	1
Encryption Services	2
IBM Encryption Facility for z/OS Client	2
DFSMSdss Encryption	2
Comparison of Encryption Facility features and functions	2
Security Server RACF enhancements	3
Encryption Services CSDFILEN and CSDFILDE.	4
Encryption Facility for OpenPGP	4
Encryption Facility for z/OS Client.	5
How CSDFILEN and CSDFILDE and the Encryption Facility for z/OS Client work.	5
How Encryption facility for OpenPGP works	6
How DFSMSdss Encryption works	6
Hardware and software requirements	7
Hardware requirements.	7
Software requirements	8
Chapter 2. Getting started	11
How do I install IBM Encryption Facility for z/OS?	11
Getting started with Encryption Services.	11
Getting started with ICSF.	12
Getting started with Encryption Facility for z/OS Client	14
Getting started with DFSMSdss Encryption.	14
Getting started with RACF	14
Chapter 3. Encrypting files through CSDFILEN of Encryption Services	17
JCL DD statements for CSDFILEN.	17
Control statement keywords for CSDFILEN SYSIN DD.	19
User guidelines and samples for encrypting data	21
When should I use CLRTDES or ENCTDES?	21
Using PASSWORD and RSA options	22
Specifying multiple RSA keys	23
Using RSA keys and digital certificates	23
Specifying the ICOUNT value	23
When should I compress data for encryption?	23
Verifying encryption files when you archive	24
Using Encryption Facility and UNIX pipes	24

Format of the header record for the CSDFILEN output file	24
Format of the statistics report file for CSDFILEN	26
Return codes for CSDFILEN	29
JCL Examples for CSDFILEN	30
ICSF callable services for CSDFILEN	32
Chapter 4. Decrypting files through CSDFILDE of the Encryption Services	33
JCL DD statements for CSDFILDE	33
Control statement keywords for CSDFILDE SYSIN DD	34
User reference information for decrypting data	36
Format of the statistics report file for CSDFILDE	36
Return codes for CSDFILDE	38
JCL examples for CSDFILDE	39
ICSF callable services for CSDFILDE	40
Chapter 5. Using Encryption Facility for the Java-based Client	41
Encryption and decryption functions	41
Installing the Java code	41
Using RSA keys and certificates	41
Considerations using the Java-based Client	42
Chapter 6. Using DFSMSdss Encryption	45
JCL examples for DFSMSdss Encryption	47
Chapter 7. Using RACF with Encryption Facility	49
Using RACF to store keys, manage PKDS labels, and send digital certificates	49
Using the RACDCERT command	50
Considerations using RACDCERT	51
Chapter 8. User scenarios	53
Encrypting data using z/OS and decrypting using the Java-based Client	53
Scenario 1	53
Scenario 2	54
Scenario 3	56
Using the RACDCERT command for key and certificate management of encrypted data.	60
Using ICSF utilities panels for PKDS key management	61
Appendix. Accessibility	65
Accessibility features	65
Consult assistive technologies	65
Keyboard navigation of the user interface	65
Dotted decimal syntax diagrams	65
Notices	69
Policy for unsupported hardware	70
Minimum supported hardware	71
Programming interface information	71
Trademarks	71
Index	73

Figures

1.	Encrypting and decrypting data with Encryption Services and the Java-based Client of Encryption Facility for z/OS Client	5
2.	Encrypting and decrypting data and processing certificates with Encryption Facility for OpenPGP	6
3.	Encrypting and decrypting data with DFSMSdss Encryption	7
4.	Establishing a trusted exchange through digital certificates.	50

About This Book

The document contains information about using IBM Encryption Facility for z/OS (Encryption Facility). This document was formerly known as *IBM Encryption Facility for z/OS: User's Guide* and is new as of Version 1.2.

Encryption Facility provides encryption and decryption processing of data for exchange between different systems and platforms and for archiving purposes. It makes use of hardware compression and encryption and relies on a centralized key management based on the z/OS® Integrated Cryptographic Service Facility (ICSF). Encryption Facility consists of the following optional features:

- Encryption Services for z/OS
- Encryption Facility DFSMSdss Encryption

Encryption Facility for Version 1.2 also supports Encryption Facility for OpenPGP, which is part of the Encryption Facility Services priced feature for z/OS. Also provided is the Encryption Facility for z/OS Client, which is separately licensed and consists of two no-charge clients that are downloadable from the Worldwide web, the Java-based Client and the Decryption for z/OS Client.

This document provides you with information to help you plan the Encryption Facility product including Encryption Services, Encryption Facility for z/OS Client, Encryption Facility for OpenPGP, and DFSMSdss Encryption. It also provides detailed information for the Encryption Services CSDFILEN and CSDFILDE batch programs.

Who should read this book

Anyone who plans, installs, customizes, administers, and uses Encryption Facility should use this document. It should also be used by those who install, configure, or provide support in the following areas:

- z/OS
- Integrated Cryptographic Service Facility (ICSF)
- Security Server Resource Access Control Facility (RACF)
- DFSMSdss

This product assumes that you have experience installing, configuring, and using z/OS, ICSF, RACF, and DFSMSdss. If you plan to use the Java reference code, the product assumes that you understand Java-related concepts and tasks.

How to use this book

This document contains the following chapters:

- Chapter 1, "Overview of IBM Encryption Facility for z/OS," on page 1 presents an overview and planning information of Encryption Facility, the functions of the product, and hardware and software requirements.
- Chapter 2, "Getting started," on page 11 presents information about installation and getting started with Encryption Facility.

- Chapter 3, “Encrypting files through CSDFILEN of Encryption Services,” on page 17 presents information about using the CSDFILEN batch program for the Encryption Services.
- Chapter 4, “Decrypting files through CSDFILDE of the Encryption Services,” on page 33 presents information about using the CSDFILDE batch program for the Encryption Services.
- Chapter 5, “Using Encryption Facility for the Java-based Client,” on page 41 presents overview information about using the Java-based Client of Encryption Facility for z/OS Client and how to obtain more information.
- Chapter 6, “Using DFSMSdss Encryption,” on page 45 presents overview information about using DFSMSdss Encryption and how to obtain more information.
- Chapter 7, “Using RACF with Encryption Facility,” on page 49 presents information about using RACF® with Encryption Facility.
- Chapter 8, “User scenarios,” on page 53 presents user scenarios for Encryption Facility.

Where to find more information

Where necessary this document references information in other documents. For complete titles and order numbers for all elements of z/OS, see *z/OS Information Roadmap*.

Related publications

The Encryption Facility library contains the following books:

- IBM Encryption Facility for z/OS: Licensed Program Specifications
- IBM Encryption Facility for z/OS: Program Directory
- *IBM Encryption Facility for z/OS: Planning and Customizing*
- *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*

The following ICSF books are referenced in this book:

- *z/OS Cryptographic Services ICSF Administrator's Guide*
- *z/OS Cryptographic Services ICSF Application Programmer's Guide*
- *z/OS Cryptographic Services ICSF Overview*
- *z/OS Cryptographic Services ICSF System Programmer's Guide*

The following RACF book is referenced in this book:

- *z/OS Security Server RACF Command Language Reference*

Documentation for the PCI Cryptographic Coprocessor is found on the web at www.ibm.com/security/cryptocards/pcicc/library.shtml.

- *IBM 4758 PCI Cryptographic Coprocessor CCA Support Program Installation Manual for IBM 4758 Models 002 and 023*
- *IBM 4758 PCI Cryptographic Coprocessor CCA Basic Services Reference and Guide for the IBM 4758 Models 002 and 023*
- *IBM 4758 PCI Cryptographic Coprocessor General Information*
- *IBM 4758 PCI Cryptographic Coprocessor Installation*

Other sources of information

IBM® provides customer-accessible discussion areas where PKI Services and RACF may be discussed by customer and IBM participants. Other information is also available through the Internet.

IBM discussion area

IBM provides the *ibm.servers.mvs.racf* newsgroup for discussion of PKI Services and RACF-related topics. You can find this newsgroup on news (NNTP) server *news.software.ibm.com* using your favorite news reader client.

Internet sources

The following resources are available through the Internet to provide additional information about PKI Services, RACF, and many other security-related topics:

- **Online library**

To view and print online versions of the z/OS publications, use this address:
<http://www.ibm.com/systems/z/os/zos/bkserv/>.

- **Redbooks™**

The Redbooks® that are produced by the International Technical Support Organization (ITSO) are available at the following address: <http://www.ibm.com/redbooks>.

- **Enterprise systems security**

For more information about security on the zSeries® platform and z/OS, use this address: <http://www.ibm.com/systems/z/advantages/security/>.

- **PKI Services home page**

You can visit the PKI Services home page on the World Wide Web using the following address. Check this site for updates regarding PKI Services.
<http://www.ibm.com/servers/eserver/zseries/zos/pki/>.

- **Techdocs**

You can visit the Techdocs - Technical Sales Library home page on the World Wide Web using the following address. Use the search keyword “crypto” to help narrow your search: <http://www.ibm.com/support/techdocs/>.

- **RACF home page**

You can visit the RACF home page on the World Wide Web using the following address. <http://www.ibm.com/systems/z/os/zos/features/racf/goodies.html>.

- **RACF-L discussion list**

Customers and IBM participants may also discuss RACF on the RACF-L discussion list. RACF-L is not operated or sponsored by IBM; it is run by the University of Georgia.

To subscribe to the RACF-L discussion and receive postings, send a note to:
listserv@listserv.uga.edu

Include the following line in the body of the note, substituting your first name and last name as indicated:

`subscribe racf-l first_name last_name`

To post a question or response to RACF-L, send a note, including an appropriate Subject: line, to:

racf-l@listserv.uga.edu

- **RACF sample code**

You can get sample code, internally-developed tools, and exits to help you use RACF. This code works in our environment, at the time we make it available, but is not officially supported. Each tool or sample has a README file that describes the tool or sample and any restrictions on its use.

To access this code from a Web browser, go to the RACF home page and select the “Downloads” topic from the navigation bar, or go to `ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/`.

The code is also available from `ftp.software.ibm.com` through anonymous FTP. To get access:

1. Log in as user anonymous.
2. Change the directory, as follows, to find the subdirectories that contain the sample code or tool you want to download:
`cd eserver/zseries/zos/racf/`

An announcement will be posted on RACF-L discussion list and on newsgroup *ibm.servers.mvs.racf* whenever something is added.

Note: Some Web browsers and some FTP clients (especially those using a graphical interface) might have problems using `ftp.software.ibm.com` because of inconsistencies in the way they implement the FTP protocols. If you have problems, you can try the following:

- Try to get access by using a Web browser and the links from the RACF home page.
- Use a different FTP client. If necessary, use a client that is based on command line interfaces instead of graphical interfaces.
- If your FTP client has configuration parameters for the type of remote system, configure it as UNIX instead of MVS[™].

Restrictions

Because the sample code and tools are not officially supported,

- There are no guaranteed enhancements.
- No APARs can be accepted.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (<http://www.ibm.com/systems/z/os/zos/webqs.html>).

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
IBM Encryption Facility for z/OS: User's Guide
SA23-1349-04
- The topic and page number that is related to your comment.
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS Support Portal (<http://www-947.ibm.com/systems/support/z/zos/>).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for SA23-1349-03 Encryption Facility for z/OS Version 1 Release 1

New Information: This release of *IBM Encryption Facility for z/OS: User's Guide* contains a reference to the Web site with information about downloading and using IBM Decryption Client for z/OS (Decryption Client).

Summary of changes for SA23-1349-02 Encryption Facility for z/OS Version 1 Release 1

New Information: This release of *IBM Encryption Facility for z/OS: User's Guide* contains information about specifying multiple RSA control statements for Encryption Facility Services and multiple RSA certificate aliases for Encryption Facility for z/OS Client. Encryption Facility for z/OS Client also supports fixed records for encryption. In addition this document provides overview information about using ICSF utility panels to manage RSA public/private key pairs in the ICSF public key data set (PKDS).

Summary of changes for SA23-1349-01 Encryption Facility for z/OS Version 1 Release 1

New Information: This release of Encryption Facility contains information for DFSMSHsm and the DFSMSdss Encryption feature. For complete information about using DFSMSHsm with the DFSMSdss Encryption feature, see the following publications:

- *z/OS DFSMSHsm Storage Administration*
- *z/OS DFSMSHsm Implementation and Customization Guide*

Chapter 1. Overview of IBM Encryption Facility for z/OS

This chapter presents an overview of IBM Encryption Facility for z/OS (Encryption Facility), the functions of the product, and hardware and software requirements.

What is Encryption Facility?

The need for creating secure archived copies of business data is a critical security concern. Encrypting data that can be recovered at any time offers a high degree of privacy protection from unwanted access. Encryption Facility provides this protection by offering encryption of data for exchange between different systems and platforms and for archiving purposes. It makes use of hardware compression and encryption and relies on a centralized key management based on the z/OS Integrated Cryptographic Service Facility (ICSF) that is highly secure and easy to use.

Encryption Facility can make use of ICSF to perform encryption and decryption and to manage cryptographic keys. To encrypt data files Encryption Facility uses the following kinds of cryptographic keys:

- TDES triple-length keys
- 128-bit AES keys, and for Encryption Facility for OpenPGP of Encryption Services, 128-bit, 192-bit, and 256-bit AES keys.

For information about cryptographic keys, see the following publications:

- *z/OS Cryptographic Services ICSF Administrator's Guide*
- *z/OS Cryptographic Services ICSF Application Programmer's Guide*

For information about hardware requirements for Encryption Facility, see "Hardware and software requirements" on page 7.

Features available with Encryption Facility

Version 1 Release 2 of IBM Encryption Facility for z/OS provides the following optional features:

Table 1. Features for Encryption Facility

Feature	Description
IBM Encryption Facility for z/OS Encryption Services	Services for encrypting and decrypting certain System z file formats on z/OS. These services can allow you to transfer data to remote sites within your enterprise, transfer them to partners and vendors, and archive them. See "Encryption Services" on page 2 and "IBM Encryption Facility for z/OS Client" on page 2.
IBM Encryption Facility for z/OS DFSMSdss Encryption	Services that run on z/OS DFSMSdss and allow you to use DFSMSdss commands to encrypt and decrypt data. With this feature, you can also use DFSMSShsm commands to encrypt and decrypt data. See "DFSMSdss Encryption" on page 2.

Encryption Services

The Encryption Services feature supports both the System z format (originally introduced in Encryption Facility for z/OS, V1.1) and the OpenPGP format (part of Encryption Facility for z/OS, V1.2). The System z format supports hardware-accelerated compression before encryption.

The Encryption Services feature includes:

- Batch programs CSDFILEN to encrypt z/OS data and CSDFILDE to decrypt z/OS data
- Encryption Facility for OpenPGP to support OpenPGP standards as described in Internet Standard RFC 2440

IBM Encryption Facility for z/OS Client

The Encryption Facility for z/OS Client is a no-cost, separately licensed program that is offered as is, with no warranty, and is designed to enable the exchange of encrypted data between z/OS systems that have the Encryption Facility installed and systems running on z/OS and other platforms that require the supported functions. The Encryption Facility for z/OS Client consists of the Java-based Client and the Decryption Client for z/OS.

DFSMSdss Encryption

The DFSMSdss Encryption feature enables the encryption of DFSMSdss DUMP data sets. You can decrypt the data through the RESTORE command. With this feature you can also use the DFSMSshm dump class settings to encrypt data dumped through the BACKVOL DUMP command and automatic dump processing and decrypt the data through the RECOVER command. This feature supports hardware-accelerated compression before encryption to tape.

Comparison of Encryption Facility features and functions

Table 2 on page 3 summarizes the functions for Encryption Services CSDFILEN and CSDFILDE batch programs, Encryption Facility for OpenPGP, Encryption Facility for z/OS Client, and DFSMSdss Encryption:

Table 2. Comparison of Encryption Facility Features and Functions

Encryption Services CSDFILEN and CSDFILDE	Encryption Facility for OpenPGP	Encryption Facility for z/OS Client	DFSMSdss Encryption
<ul style="list-style-type: none"> • Allows encryption and compression of z/OS format data through CSDFILEN • Supports decryption and decompression of z/OS format data through CSDFILDE • Makes use of z/OS centralized key management and access authentication • Uses IBM mainframe server cryptographic and compression capabilities • Can use either public/private key pairs or passwords to create secure exchange between partners. 	<ul style="list-style-type: none"> • Allows encryption and decryption of z/OS format data as well as message files based on Internet Standard RFC 2440 • Makes use of z/OS centralized key management and IBM mainframe cryptographic functions and compression • Can encrypt or decrypt data and message files with the OpenPGP format on z/OS and other platforms that support OpenPGP-compliant applications • Can use either public/private key pairs or passwords to create secure exchange between partners. 	<p>Java-based Client:</p> <ul style="list-style-type: none"> • Allows client systems to encrypt and decrypt data files created on a z/OS system through the Encryption Services CSDFILEN and CSDFILDE batch programs • Can be used on any Java-enabled system • Can use either public/private key pairs or passwords to create secure exchange between partners. • Does not support compression of data for encryption and cannot process encrypted data that has been compressed using Encryption Services. <p>Decryption Client for z/OS:</p> <ul style="list-style-type: none"> • Allows decryption and decompression of data files created on a z/OS system through the Encryption Services CSDFILEN batch program • Can compress the data that is to be processed for performance benefits and to require less media for exchange purposes • Does not support data encryption • Can use either public/private key pairs or passwords to create secure exchange between partners. 	<ul style="list-style-type: none"> • Allows encryption and compression of dump data sets created by DFSMSdss or DFSMSHsm • Supports decryption and decompression during RESTORE process for DFSMSdss and RECOVER for DFSMSHsm • Makes use of z/OS centralized key management and IBM mainframe cryptographic functions and compression • Can use either public/private key pairs or passwords to create secure exchange between partners.

Security Server RACF enhancements

Enhancements to Security Server RACF (RACF), an element of z/OS, allow you to store internal RSA public and private keys for encrypted data in the ICSF public key data set (PKDS) and specify the PKDS labels for security certificates associated with encrypted data. Although you can use your own programs to load the PKDS, RACF provides a command (RACDCERT) to load the keys.

Encryption Services CSDFILEN and CSDFILDE

Encryption Services batch programs CSDFILEN and CSDFILDE can make use of a symmetric key that is randomly generated to protect System z format data. CSDFILEN encrypts this data key and stores it with the data in a header record. Encryption Services allows you to protect the data key by using public key architecture, by providing an RSA key label, or, if an RSA public/private key pair is not available, by using a key that is generated from a password that you provide.

The encrypted data contains a header with enough information to use for decryption. You can use the CSDFILDE batch program to decrypt the z/OS data.

Decryption Client for z/OS, which is part of the Encryption Facility for z/OS Client, can also make use of CSDFILDE to decrypt data on a z/OS platform. Or you can send the encrypted data to a Java platform where you can use the Java-based Client, also part of Encryption Facility for z/OS Client, to decrypt the data.

You code job control statements (JCL) to control CSDFILEN encryption and CSDFILDE decryption services. You can optionally specify that the data is to be compressed before encryption and that the encrypted data, which is an output sequential file, is to be sent to tape for archiving or transfer.

For detailed information, see Chapter 3, “Encrypting files through CSDFILEN of Encryption Services,” on page 17 and Chapter 4, “Decrypting files through CSDFILDE of the Encryption Services,” on page 33.

Encryption Facility for OpenPGP

Encryption Facility for OpenPGP can encrypt and decrypt z/OS-type data for use with OpenPGP-compliant systems as well as OpenPGP-compliant messages and files. It includes but is not limited to the following support:

- Passphrase base encryption of session key
- Digital signatures of data
- Importing/exporting of OpenPGP certificates (V3 and V4 for importing, only export V4, unless exporting an imported V3 key)
- RSA (1), ElGamal, and DSA (1) key generation
- Use of partial data packets
- ASCII Armor for OpenPGP certificates
- Data encryption with a randomly generated symmetric session key using AES 128 (1), 192, and 256 bit keys, Triple-DES (1), and Blowfish algorithms (2)
- Symmetric encryption of randomly generated symmetric session key using AES 128 (1), 192, and 256 bit keys, Triple-DES (1), and Blowfish algorithms (2)
- Asymmetric encryption of randomly generated symmetric keys using RSA (1) and ElGamal algorithms
- Compression using ZIP and ZLIB algorithms
- Digest/Hash using SHA-1 (1), MD5 (1), MD2 (1), SHA-256 (1), SHA-384, SHA-512 algorithms
- Digital Signature using DSA with SHA1 (1) and RSA (with all supported hashes listed above) (1) algorithm.

For complete information, see *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*.

Encryption Facility for z/OS Client

Encryption Facility for z/OS Client consists of the following services:

- Java-based Client
- Decryption Client for z/OS

The IBM Encryption Facility for z/OS Client (Web download program) includes the Java-based Client that can decrypt the contents of tapes encrypted by the Encryption Services feature running on z/OS and encrypt tapes to be decrypted using the Encryption Services feature on z/OS. This allows the exchange of encrypted data between z/OS systems and other operating systems.

The Decryption Client for z/OS component is designed to decrypt data that has been created through the Encryption Services feature on z/OS. The decryption component can decrypt data that has been processed using compression.

For complete information, see <http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>.

How CDSFILEN and CDSFILDE and the Encryption Facility for z/OS Client work

Figure 1 shows how the encryption process works with Encryption Services CDSFILEN and CDSFILDE and Encryption Facility for z/OS Client. You can archive the encrypted data and decrypt the data through CDSFILDE, the Java-based Client, or the Decryption Client for z/OS:

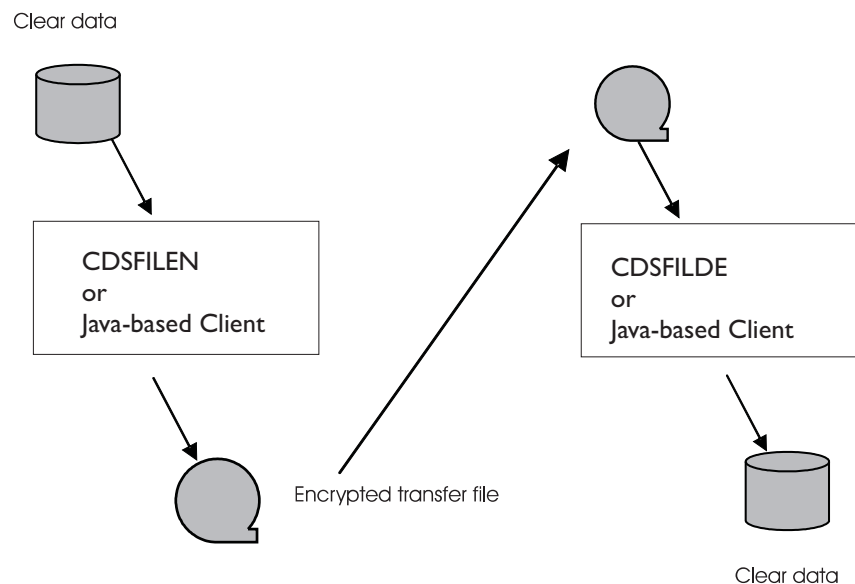


Figure 1. Encrypting and decrypting data with Encryption Services and the Java-based Client of Encryption Facility for z/OS Client

How Encryption facility for OpenPGP works

Figure 2 shows the general processing of OpenPGP format data by Encryption Facility for OpenPGP. It shows how Encryption Facility for OpenPGP encrypts and decrypts z/OS-type data, generates X.509 certificates for OpenPGP messages, imports or exports OpenPGP certificates, decrypts or verifies OpenPGP messages, and processes public and private keys for OpenPGP data:

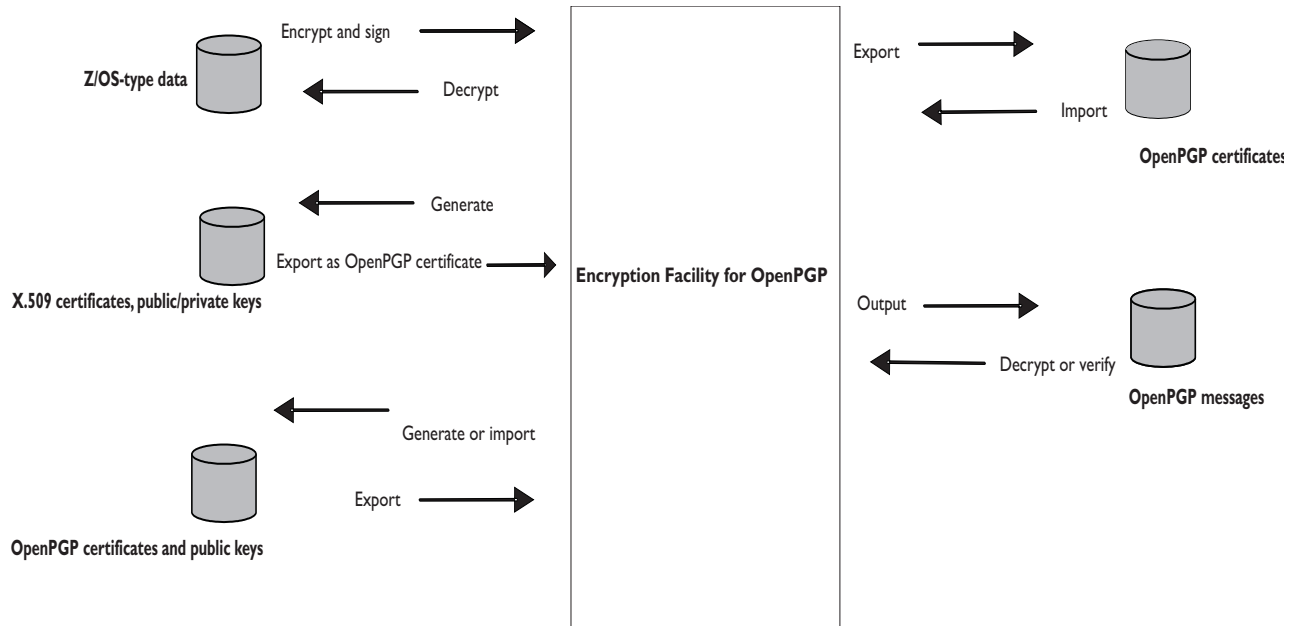


Figure 2. Encrypting and decrypting data and processing certificates with Encryption Facility for OpenPGP

How DFSMSdss Encryption works

The encryption/decryption process for DFSMSdss Encryption is similar to that of Encryption Services. The DFSMSdss DUMP command and the DFSMSHsm DEFINE DUMPCLASS commands provide some of the same options as those for the CSDFILEN batch program of Encryption Services to encrypt data. The DFSMSdss RESTORE command and the DFSMSHsm RECOVER commands provide some of the same options as those for the CSDFILEN batch program of Encryption Services to decrypt the data. You can encrypt data to tape or DASD.

Figure 3 on page 7 shows how the encryption process works with DFSMSdss Encryption. You use DFSMSdss to encrypt data through the DUMP command where you can archive the data. You can then decrypt the data through the RESTORE command:

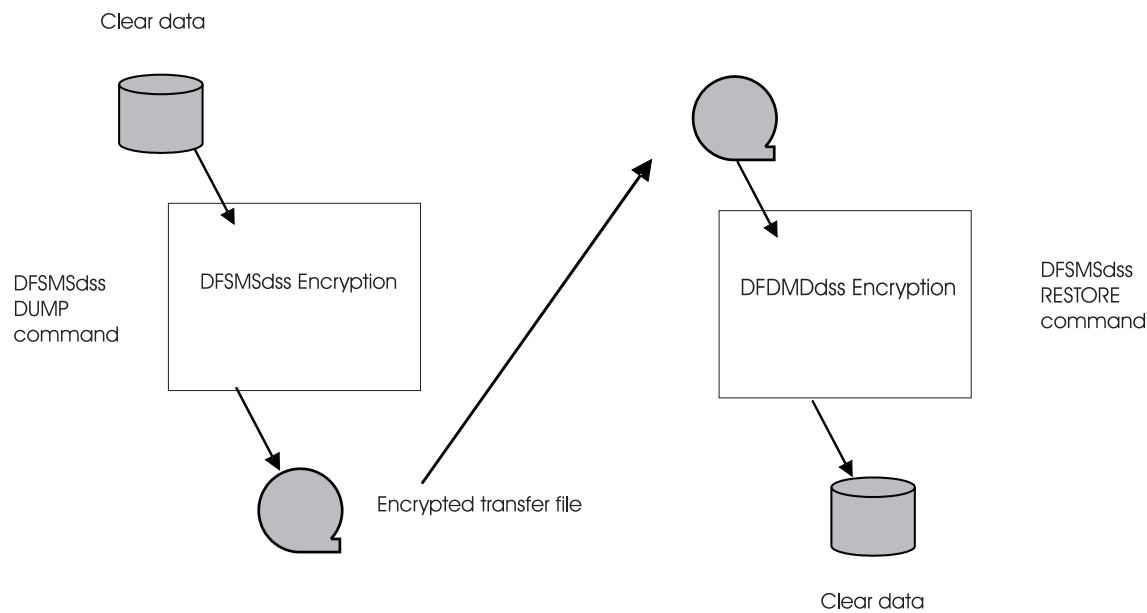


Figure 3. Encrypting and decrypting data with DFSMSdss Encryption

Hardware and software requirements

The following topics describe hardware and software requirements for Encryption Facility.

Hardware requirements

The options that you specify for encryption and decryption including cryptographic keys depend on the processor type and the cryptographic hardware that you have installed on the system. The Encryption Services and the DFSMSdss Encryption features of the Encryption Facility for z/OS run on the following IBM servers:

- System z9 BC or z9 EC, or equivalent
- zSeries z900 or z990, or equivalent
- zSeries z800 or z890, or equivalent

The cryptographic options for Encryption Facility V1.1 and higher have the following requirements:

- For the PASSWORD option, use one of the following:
 - CPACF only
 - CCF
- For the Clear-TDES and Clear-AES128 (no ENCTDES), use one of the following:
 - CPACF only, or CPACF with PCIXCC / CEX2C
 - CCF, or CCF with PCICC
- For 2048-bit keys, use one of the following:
 - CEX2C
 - PCIXCC
 - PCICC with PCI Crypto 2048 bit Enablement Feature 0867

- For RSA keys generated through RACF using ICSF or directly through ICSF, use one of the following:
 - CEX2C
 - PCIXCC
 - PCICC
- For 1024-bit ME keys generated through RACF BSAFE and imported into ICSF, a CCF is required.

Performance for secure key (ENCTDES option) is slower than clear key (CLEAR-TDES or CLEAR-AES128). IBM recommends the use of clear key for encrypting large volumes of data.

OpenPGP support and hardware cryptography

For AES or TDES symmetric encryption use one of the following:

- CPACF only
- CPACF with PCIXCC / CEX2C
- CCF
- CCF with PCICC

For signatures or session key encryption using 2048-bit keys or 2048-bit RSA key generation, use one of the following:

- CEX2C
- PCIXCC
- PCICC with PCI Crypto 2048 bit Enablement Feature 0867

For signatures or session key encryption using RSA 1024-bit ME keys generated through RACF BSAFE, imported into ICSF, and prepared for OpenPGP use, a CCF is required.

For signatures or session key encryption using RSA keys generated through RACF using ICSF or directly through ICSF and prepared for OpenPGP use, use one of the following:

- CEX2C
- PCIXCC
- PCICC

Software requirements

Software requirements for Encryption Facility Version 1.2 and later are as follows:

Encryption Services CSDFILEN and CSDFILDE: Encryption Services CSDFILEN and CSDFILDE batch programs require the following:

- z/OS (5694-A01) or z/OS.e (5655-G52) V1.6 or later release
- PTF for z/OS DFSMS APAR OA09868 and QSAM APAR OA13571
- Integrated Cryptographic Services Facility (ICSF) Web deliverable (FMID HCR7720) or later

Encryption Services for OpenPGP: The Encryption Services feature of the Encryption Facility for z/OS also requires the following for OpenPGP format:

- z/OS (5694-A01) or z/OS.e (5655-G52) V1.6 or higher
- For z/OS (5694-A01) or z/OS.e (5655-G52) V1.6, PTF for z/OS APAR OA10229

- Integrated Cryptographic Services Facility (ICSF) Web deliverable (FMID HCR7720) or later
- IBM 31-bit SDK for z/OS, Java 2 Technology Edition, Version 5, product 5655-N98 at Service Refresh level SDK5 SR4 or later
- PTF for z/OS ICSF APAR OA19177

You cannot use System z format data that has been processed through CSDFILEN, CSDFILDE, and Encryption Facility for z/OS Client in Encryption Facility Version 1.1 and Version 1.2 with Encryption Facility for OpenPGP Version 1.2.

Encryption Facility for z/OS Client (optional):

The Java-based Client of Encryption Facility Client for z/OS requires the following:

- To run on z/OS, one of the following is required:
 - IBM SDK for z/OS, Java™ 2 Technology Edition, 5655-I56, at PTF UQ90449 or later (SDK1.4.2)
 - IBM Developer Kit for OS/390®, Java 2 Technology Edition, 5655-D35, at PTF UQ88094 or later (SDK1.3.1)
- To run on other platforms, one of the following is required:
 - Sun SDK 5.0
 - An IBM JVM at SDK1.4.2
 - A JVM with a JCE cryptographic provider installed that supports all the required algorithms. See Encryption Facility for z/OS Client documentation for details on the algorithms, modes, and padding schemes needed.

Decryption Client for z/OS requires the following:

- z/OS (5694-A01) or z/OS.e (5655-G52) V1.4 or higher (Note: The Decryption Client for z/OS only runs on z/OS and is supported with both Encryption Facility for z/OS, V1.1 and V1.2.)
- PTF for z/OS DFSMS APAR OA09868
- z/OS Cryptographic Services - Integrated Cryptographic Services Facility with z990 Cryptographic Support Web deliverable (FMID HCR770A) or later. Some hardware features require the z990 and z890 Enhancements to Cryptographic Support Web deliverable (FMID HCR770B) or later.

Decryption Client for z/OS runs only on the z/OS platform and only provides decryption services.

For complete information on Encryption Facility for z/OS Client, see the following Web site: <http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>. For the Java-based Client, including PTF requirements for iSeries or other platforms, see the README file in the Java reference code for Encryption Facility for z/OS Client on the same Website.

DFSMSdss Encryption (optional): DFSMSdss Encryption feature of Encryption Facility for z/OS requires the following:

- z/OS (5694-A01) or z/OS.e (5655-G52) V1.6 or higher
- Integrated Cryptographic Services Facility (ICSF) Web deliverable (FMID HCR7720) or later
- Either the DFSMSHsm/DFSMSdss combination priced feature or the DFSMSdss priced feature of z/OS or z/OS.e V1.6 or higher
- PTF for z/OS DFSMS APARs OA13300, OA13453, and OA13687

On any processors with CCFs (for example, z900 and z800), the cryptographic key data set (CKDS) must be populated with SYSTEM keys (NOCVKEYS, ANSI, ESYS). See *z/OS Cryptographic Services ICSF Administrator's Guide*.

ICSF: z/OS Cryptographic Services requires:

- ICSF 64-bit Virtual Support for z/OS V1.6 and z/OS.e V1.6 (HCR7720)
- ICSF in z/OS 1.7 base (HCR7720) or z/OS 1.8 base (HCR7740)
- Cryptographic Support for z/OS V1R6/R7 and z/OS.e V1R6/R7 (HCR7730)
- Cryptographic Support for z/OS V1R8 and z/OS.e V1R8 (HCR7731)

Security Server (RACF) (optional): The optional PTF for Security Server (RACF) APAR OA13030 to perform the following:

- Use the RACF RACDCERT command to allow the storage of RSA public keys in the ICSF PKDS
- Specify the PKDS labels to be used when storing public or private keys in the PKDS
- List the PKDS labels of existing certificates

Chapter 2. Getting started

This chapter describes installation tasks and considerations for getting started using IBM Encryption Facility for z/OS:

- “How do I install IBM Encryption Facility for z/OS?”
- “Getting started with Encryption Services”
- “Getting started with ICSF” on page 12
- “Getting started with Encryption Facility for z/OS Client” on page 14
- “Getting started with DFSMSdss Encryption” on page 14
- “Getting started with RACF” on page 14

For information about Encryption Facility for OpenPGP, see *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*.

How do I install IBM Encryption Facility for z/OS?

You can install Encryption Facility and use any of the following optional features:

- Encryption Services
- DFSMSdss Encryption

You can download Encryption Facility for z/OS Client from the World Wide Web.

The following steps are a summary of how to install Encryption Facility. These steps provide only a broad description. For installation information, see *IBM Encryption Facility for z/OS: Program Directory*. For hardware and software requirements for Encryption Facility, see “Hardware and software requirements” on page 7.

1. Order and install program product, 5655-P97 (IBM Encryption Facility for z/OS).
2. Ensure that you have the required software installed on your z/OS system.
3. Apply any service from PSP Buckets.
4. Optional: Download Encryption Facility for z/OS Client through click through license from the following Web site:<http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>

Getting started with Encryption Services

Encryption Services includes the CSDFILEN and CSDFILDE batch programs and the Encryption Facility for OpenPGP. Encryption Services is an SMP/E installable program.

You can use Encryption Services CSDFILEN and CSDFILDE to encrypt and decrypt data on z/OS and perform other functions..

SYS1.SAMPLIB: You can find DDDEF and ALLOC jobs for the Encryption Services in the following SYS1.SAMPLIB members:

- CSDDDEF for DDDEF
- CSDALLOC for ALLOC

For information about Encryption Services CSDFILEN and CSDFILDE, see Chapter 3, “Encrypting files through CSDFILEN of Encryption Services,” on page 17 and Chapter 4, “Decrypting files through CSDFILDE of the Encryption Services,” on page 33.

As of Encryption Facility Version 1.2 and later, you can use Encryption Services for OpenPGP to process data according to OpenPGP standards as described in Internet Standard RFC 2440. Encryption Facility for OpenPGP allows you to encrypt and decrypt z/OS-type data for use with OpenPGP compliant systems and manage both X.501 certificates and OpenPGP certificates. For complete information, see *IBM Encryption Facility for z/OS: Using Encryption Facility for OpenPGP*.

Getting started with ICSF

If you have ICSF installed, see “Software requirements” on page 8 to ensure that you are using the required level for Encryption Facility.

If you need information about installing, planning, and implementing ICSF, see the following publications:

- *z/OS Cryptographic Services ICSF Overview*
- *z/OS Cryptographic Services ICSF System Programmer's Guide*
- *z/OS Cryptographic Services ICSF Administrator's Guide*

Encryption Facility makes use of ICSF to manage cryptographic keys for encrypted data.

ICSF supports the following cryptographic standards and architectures:

- IBM Common Cryptographic Architecture (CCA) that is based on the ANSI Data Encryption Standard (DES)
- Advanced Encryption Standard (AES).

Cryptographic keys: In the secret key cryptography system based on DES, two parties share secret keys that are used to protect data and keys that are exchanged on the network. Sharing secret keys establishes a secure communications channel. The only way to protect the security of the data in a shared secret key cryptographic system is to protect the secrecy of the secret key.

ICSF also supports triple DES encryption for data privacy. TDES triple-length keys use three, single-length keys to encipher and decipher the data. This results in a stronger form of cryptography than that available with single DES encipherment.

With AES, data can be encrypted and decrypted using 128-bit, 192-bit, and 256-bit clear keys. CBC and ECB encryption are also supported.

For public key cryptography, ICSF supports both the Rivest-Shamir-Adelman (RSA) algorithm 1, and the NIST Digital Signature Standard algorithm. RSA is one of the most widely used public key encryption algorithms. In this system, each party establishes a pair of cryptographic keys, which includes a public key and a private key. Both parties publish their public keys in a reliable information source, and maintain their private keys in secure storage.

Cryptographic keys and Encryption Facility: Encryption Facility makes use of TDES triple-length keys and 128-bit AES keys for data encryption. On a system with secure cryptographic hardware, you can use Encryption Facility to generate TDES and AES keys and encrypt them for protection through RSA public keys. On

systems without secure cryptographic hardware, a password allows the generation of clear TDES and AES keys. The use of these cryptographic keys with Encryption Facility depends on the kind of processor and the type of cryptographic hardware that you have installed.

Generating and placing an RSA key in the PKDS: RSA public and private keys for encryption can be stored in the ICSF public key data set (PKDS). These RSA keys are used by Encryption Facility to protect the symmetric keys that protect the data. You can specify multiple RSA keys as input to Encryption Services or Encryption Facility for z/OS Client and copy and distribute the resulting output file to multiple recipients. You can also use ICSF callable services to generate RSA keys and place them in the PKDS. The required ICSF callable services are CSNDPKB PKA key token build and CSNDPKG PKA key generate.

CSNDPKB builds a skeleton PKA token. The principal parameters are as follows:

- Rule array
- Key Value Structure (KVS)
- Generated Key Token (KeyToken) .

For example, the parameters for the generation of a skeleton key token for a 1024 bit RSA private key are as follows:

- PKB_RULE = "RSA-PRIVKEY-MGMT"
- PKB_KVS = "0400000000030000010001"
- PKB_KeyToken = (generated)

CSNDPKG generates key values for the PKA token. The principal parameters are as follows:

- Rule array
- Skeleton key identifier (SkelKey)
- Generated key identifier (GenKey)

For example, the parameters for a 1024 bit RSA private key are as follows:

- PKG_RULE = "MASTER"
- PKG_SkelKey = PKB_KeyToken
- PKG_GenKey = "THIS.CAN.BE.A.PKDS.LABEL"

If you specify a PKDS key label for GenKey, ICSF writes the token to the PKDS.

Using the ICSF utility panels to create or delete PKDS records and import or export RSA keys: You can use ICSF utility panels to create or delete PKDS records and export or import RSA keys to an x.509 certificate. You use x.509 certificates to certify the transmission of the RSA public keys between senders and receivers of encrypted data. For information about using digital certificates, see "Using RACF to store keys, manage PKDS labels, and send digital certificates" on page 49.

Coprocessor Requirements for using the ICSF utility panels: To use the full function of the ICSF utility panels, you must have a PCICC, PCIXCC, or a CEX2C cryptographic coprocessor. If you do not have one of these coprocessors, you cannot generate key pairs using the panels.

For information about using the ICSF utility panels, see "Using ICSF utilities panels for PKDS key management" on page 61. For complete information about using ICSF utility panels and services, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

ICSF uses the following ICSF callable services to create or delete PKDS records and export or import RSA keys to x.509 certificates:

- CSNDPKB (builds the skeleton key token)
- CSNDKRC (creates the PKDS record)
- CSNDKRD (deletes the PKDS record)
- CSNDKRR (reads the record from the PKDS)
- CSNDPKX (extracts only the public key from the record)
- CSNBOWH (hashes the to-be-signed portion of the generated certificate)
- CSNDDSG (signs the hash)

If you are using RACF or similar security product, ensure that the security administrator authorizes ICSF to use these services and any cryptographic keys that are input. For information about ICSF callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

Getting started with Encryption Facility for z/OS Client

Encryption Facility for z/OS Client is a separately licensed, no-charge, web download that includes the Java-based Client and the Decryption for z/OS Client. You can use Encryption Facility for z/OS Client on non-z/OS platforms to encrypt and decrypt z/OS format data that is created through Encryption Services CSDFILEN and CSDFILDE on z/OS. You can use the Decryption for z/OS Client to decrypt data on z/OS platforms that is encrypted through Encryption Services CSDFILEN.

You can download Encryption Facility for z/OS Client from the following Web site: <http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>.

Be sure to read the README file for complete information about software requirements including any PTFs for iSeries or other platforms.

For overview information, see Chapter 5, “Using Encryption Facility for the Java-based Client,” on page 41.

Getting started with DFSMSdss Encryption

You can use DFSMSdss Encryption of Encryption Facility to perform the encryption of output data sets from the DFSMSdss DUMP command to tape or DASD. You can decrypt the data through the DFSMSdss RESTORE command. With this feature you can also use the DFSMSShsm dump class settings to encrypt data dumped through the BACKVOL DUMP command and automatic dump processing. You can use the DFSMSShsm RECOVER command to decrypt the data.

See Chapter 6, “Using DFSMSdss Encryption,” on page 45.

Getting started with RACF

You can use RACF to help you store RSA public and private keys for encryption in the ICSF public key data set (PKDS). You can also specify the PKDS labels to use when you store public or private keys in the PKDS and can list PKDS labels of public/private key pairs from existing certificates that reside in the RACF database.

The certificate management services of RACF allow you to establish a limited scope certificate authority for your internal and external users, issuing and administering digital certificates in accordance with your own organization's policies.

For information about using RACF to store keys and generate labels, see Chapter 7, "Using RACF with Encryption Facility," on page 49.

Chapter 3. Encrypting files through CSDFILEN of Encryption Services

This chapter presents information about using the CSDFILEN batch program of Encryption Services to encrypt data.

CSDFILEN is the batch program of Encryption Services that supports System z file formats and encrypts input data on z/OS. Depending on the kind of processor and cryptographic coprocessor, you can specify options to use clear key or secure key encryption, specify options for key protection, and indicate if you want to compress the data before it is to be encrypted.

JCL DD statements for CSDFILEN

CSDFILEN supports the following DD statements or their dynamic allocation equivalents:

Table 3. JCL DD Statements for CSDFILEN

DD statement	Description
SYSPRINT	<p>Specifies the name of the data set to which CSDFILEN writes encryption statistics and diagnostics information. It can be a sequential data set as follows:</p> <ul style="list-style-type: none">• Typically a sysout data set• DASD or tape data set• PDS or PDSE member <p>CSDFILEN sets the following values:</p> <ul style="list-style-type: none">• RECFM=FBA• LRECL=133 <p>The system selects an optimal value for BLKSIZE unless you choose to code BLKSIZE. BLKSIZE must be a multiple of 133 unless it is a sysout DD statement, but coding BLKSIZE on a sysout DD statement is not beneficial.</p>
SYSIN	<p>Specifies the source from which CSDFILEN reads control statements. It can be a sequential data set as follows:</p> <ul style="list-style-type: none">• Typically a spooled system input data set• ONE of the following specifications:<ul style="list-style-type: none">– DASD or tape data set– PDS or PDSE member <p>CSDFILEN requires the following specifications:</p> <ul style="list-style-type: none">• RECFM=F or FB• LRECL=80

Table 3. JCL DD Statements for CSDFILEN (continued)

DD statement	Description
SYSUT1	<p>Specifies the name of the data set that contains data to be encrypted. It can be a sequential data set as follows:</p> <ul style="list-style-type: none"> • DASD or tape data set • PDS or PDSE member • z/OS UNIX Systems Services file <p>The input to CSDFILEN can also be a system input data set, that is if you code * or DATA on the DD statement. Unlike the other types of input, a dynamic allocation equivalent of this data set does not exist.</p> <p>CSDFILEN can read data sets created with BSAM, QSAM, BPAM or EXCP.</p> <p>You must first unload VSAM data in order to encrypt it.</p> <p>For z/OS UNIX Systems Services files, consider the following coding practices:</p> <ul style="list-style-type: none"> • Code the PATH keyword and PATHOPTS=ORDONLY. • Do not code FILEDATA=TEXT so that CSDFILEN can read the data without character conversion. • Do not code variable block (VB) for BLKSIZE. <p>For all types of input consider the following for record format and DCB information:</p> <ul style="list-style-type: none"> • The data set or file can have any record format (RECFM). • If the data set label (whether on DASD or tape) contains the DCB information, you do not have to code RECFM, LRECL or BLKSIZE on the DD statement. • If you plan to decrypt the file to a z/OS UNIX Systems Services file or send the file to a non-z/OS system, code RECFM=U so that you get the maximum length of records for better efficiency. <p>Normally DASD data sets and standard labelled tapes have the correct DCB information. If the record format is not available in the data set label and you do not code RECFM on the DD statement, CSDFILEN assumes RECFM=U (undefined format). If the block size is not available from the data set label and you do not code BLKSIZE on the DD statement, CSDFILEN assumes the maximum block size supported by the device. If this block size value is much larger than the sizes of the real blocks, the program might run more slowly than you expect.</p> <p>IBM suggests that if the input is an unlabeled dump tape created by DFSMSdss, you code BLKSIZE=60000, which is a little larger than the largest block expected.</p> <p>If CSDFILEN is using a record format of fixed or variable and the record length is not available from the data set label and you do not code LRECL on the DD statement, CSDFILEN fails.</p>

Table 3. JCL DD Statements for CSDFILEN (continued)

DD statement	Description
SYSUT2	<p>Specifies the name of the data set that is to contain the encrypted data. It can be a sequential data set as follows:</p> <ul style="list-style-type: none"> • DASD or tape data set • PDS or PDSE member <p>CSDFILEN forces RECFM=U. You can specify the maximum block size by coding the BLKSIZE or BLKSZLIM keyword. With either keyword, if the value exceeds the maximum supported by the device, CSDFILEN reduces the value. If you do not code a value, CSDFILEN assumes the optimal value for the device. For information about these values, see the INFO=AMCAP option of the DEVTYPE macro in <i>z/OS DFSMSdfp Advanced Services</i>.</p> <p>If you want to copy the file containing the encrypted data to another device (for example, you specify a tape data set on SYSUT2 but later copy the data set to DASD), you might want to code BLKSIZE to reflect a block size that is applicable to the other device (for example, 23476 for 3380 DASD or 27998 for 3390 DASD). In a PDSE or z/OS UNIX Systems Services file, CSDFILDE simulates an optimal block size, which typically is 32760.</p> <p>The following list is a summary of default block sizes depending on the kind of specified device:</p> <ul style="list-style-type: none"> • If the device is DASD, the default block size is a half track. • If the device is an IBM 3590 or a newer tape device, the default block size is 256 KB. • If the device is an IBM 3480 or 3490 Tape Subsystem or an IBM Virtual Tape Server, the default block size is 65535. Note that this data set cannot be an ANSI/ISO standard labelled tape or a tape for which OPTCD=Q is specified. In either case, the system requires the data to be character data and performs character conversion, which thereby destroys encrypted data. <p>DO NOT specify DISP=MOD for the SYSUT2 data set on CSDFILEN. You might encounter an error when you try to decrypt the data through CSDFILDE.</p>

Control statement keywords for CSDFILEN SYSIN DD

You can specify the following options in the control statement data set (identified by SYSIN DD) to control encryption of the input files. All keywords must start in column 1, and you cannot code a continuation statement. The program treats an asterisk (*) in column 1 as a comment. If you specify the same keyword multiple times, the program uses the last specification:

Table 4. Control Statement Keywords for CSDFILEN SYSIN DD

Description	JCL keyword
Descriptive text	<p>DESC='text'</p> <p>Specifies 1 to 64 EBCDIC character bytes of descriptive <i>text</i> to be placed in the header record. CSDFILEN places the information in the header record. The information is used to assist in identifying the source of the encrypted data in the output. You must enclose the text in single quotation marks. Imbedded blanks are allowed. All text must be included on one control statement line. DESC is optional.</p>

Table 4. Control Statement Keywords for CSDFILE SYSIN DD (continued)

Description	JCL keyword
Encryption type	<p>Specifies information about which encryption key you want Encryption Services to generate. You can specify one of the following types. If you do not specify an option, the default is CLRTDES:</p> <p>CLRTDES Specifies that the input file is to be encrypted with a clear TDES triple-length key.</p> <p>CLRAES128 Specifies that the input file is to be encrypted with a clear 128-bit AES key.</p> <p>ENCTDES Specifies that the input file is to be encrypted with a secure TDES triple-length key.</p>
Method to generate and protect the data encrypting key	<p>Specifies the method to be used to generate and protect the data encrypting key. RSA and PASSWORD are mutually exclusive. One of the following keywords is required:</p> <p>RSA=label Specifies the 64-byte <i>label</i> of an existing RSA public key that is in the ICSF PKDS. The program uses this key to encrypt the data-encrypting key. The corresponding RSA private key must be present at the recovery site when you decipher the data. RSA= can point to an RSA key that contains both a private and a public key, or you can specify the name of the corresponding RSA private key when you invoke CSDFILDE or Encryption Facility for z/OS Client at the recovery site.</p> <p>For Encryption Services you can use from 1 to 16 RSA= keywords to specify from 1 to 16 public key labels. Depending on the number of multiple RSA labels, you can send the encrypted file to up to 16 individual recipients.</p> <p>For how to specify multiple RSA key labels, see “Specifying multiple RSA keys” on page 23.</p> <p>PASSWORD=password Specifies the 8- to 32- EBCDIC character <i>password</i> to be used to generate a clear TDES triple-length key or a clear 128-bit AES key. Leading and trailing blanks and tab characters are removed; imbedded blanks and tab characters are allowed. Passwords are case sensitive.</p> <p>You can specify this option on systems that do not have secure cryptographic hardware (for example, for z890, z990 or z9-109 processors that only use CPACF).</p> <p>In order to minimize problems because of code page differences at the encrypting and decrypting sites, IBM suggests that you use only the upper and lower-case letters A through Z, numerals 0 – 9 and the underscore character (_).</p>

Table 4. Control Statement Keywords for CSDFILEN SYSIN DD (continued)

Description	JCL keyword
Number of iterations	<p>ICOUNT=nnnnn</p> <p>When you specify a password, specifies the number of iterations that the SHA-1 hash algorithm is to be performed in the generation of the data key and the initial chaining vector (ICV) for encryption. nnnnn is an integer between 1 and 10000. If you do not specify ICOUNT, the default is 16.</p> <p>ICOUNT allows you to strengthen security when you use PASSWORD. If you specify a robust password (32 random characters), the default is sufficient. See “Specifying the ICOUNT value” on page 23.</p>
Compression option	<p>COMPRESSION=NO YES</p> <p>Specifies whether you want compression of the clear input before encryption of the data occurs. COMPRESSION=NO indicates that compression does not occur. COMPRESSION=YES causes compression to be performed before encryption. If you do not specify the COMPRESSION keyword, the default is NO.</p>

User guidelines and samples for encrypting data

Guidance information includes the following topics:

- “When should I use CLRTDES or ENCTDES?”
- “Using PASSWORD and RSA options” on page 22
- “Specifying multiple RSA keys” on page 23
- “Using RSA keys and digital certificates” on page 23
- “Specifying the ICOUNT value” on page 23
- “When should I compress data for encryption?” on page 23
- “Verifying encryption files when you archive” on page 24
- “Using Encryption Facility and UNIX pipes” on page 24

Reference information includes the following topics and samples:

- “Format of the header record for the CSDFILEN output file” on page 24
- “Format of the statistics report file for CSDFILEN” on page 26
- “Return codes for CSDFILEN” on page 29
- “JCL Examples for CSDFILEN” on page 30

When should I use CLRTDES or ENCTDES?

The decision on whether to use CLRTDES or ENCTDES key values depends on the kind of cryptographic hardware you have, the level of security you want, and the level of performance.

For Encryption Facility, a CLRTDES key is a triple-length TDES key that the service generates dynamically. Unlike the ENCTDES key value the CLRTDES key value can appear in application storage. If Encryption Facility is running on a z890, z990, or System z9 109, CSDFILEN encrypts the data using the clear TDES key on the CPACF. This usually results in better performance than if you are using the ENCTDES key value.

The ENCTDES key is a triple-length TDES key that the service generates within the secure boundary of the cryptographic hardware (CCF, PCICC, PCIXCC, or CEX2C), and it uses the ICSF symmetric master key to encrypt the triple-length TDES key that the service generates. The clear value of an ENCTDES key never leaves the boundary of the secure cryptographic hardware. Encryption and decryption of data using an ENCTDES key requires secure cryptographic hardware to be available.

Each type of key is equally secure in regards to the data that appears in the output file of the CSDFILEN statistics report file; that is, CSDFILEN does not write any clear key information to the file.

Using PASSWORD and RSA options

PASSWORD and RSA options for the encrypted data depend on the processor and cryptographic hardware that you have installed. Base your decisions on the security requirements of the data and on your available hardware.

PASSWORD option: Generally, if you do not have secure cryptographic hardware installed, you can specify the PASSWORD keyword. Passwords are case sensitive.

RSA option: Consider using the RSA keyword that makes use of public/private keys for encryption and the exchange of digital certificates. You specify the label of the public key that is stored in the ICSF PKDS on the RSA option when you encrypt the data. The corresponding RSA private key must be present at the recovery site when you decipher the data. A recipient on another site can decrypt the data through the private key that is specified on the RSA option for CSDFILDE. Optionally, the recipient can decrypt the data through the private key that is specified by the **-keyStoreCertificateAlias** argument on the Java-based Client of the Encryption Facility for z/OS Client; see “Using RSA keys and certificates” on page 41. You can specify multiple RSA keys as input.

See “Control statement keywords for CSDFILDE SYSIN DD” on page 34.

RSA private tokens: When you use the RSA option with CSDFILEN to encrypt the data-encrypting key, you must consider the cryptographic hardware that exists at the site that decrypts the data. Table 5 summarizes the RSA private tokens and required cryptographic hardware for decryption. For details, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*:

Table 5. RSA private tokens and required cryptographic hardware

RSA private key token (internal)	Required cryptographic hardware
RSA private key token 1024 Modulus-Exponent Internal form	One of the following: <ul style="list-style-type: none"> • Cryptographic Coprocessor Feature • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor
RSA private key token 1024 Chinese Remainder Theorem Internal form	One of the following: <ul style="list-style-type: none"> • PCI Cryptographic Coprocessor • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor

Table 5. RSA private tokens and required cryptographic hardware (continued)

RSA private key token (internal)	Required cryptographic hardware
RSA private key token 2048 Chinese Remainder Theorem Internal form	<p>One of the following:</p> <ul style="list-style-type: none"> • PCI Cryptographic Coprocessor with LIC January 2005 or later and z/OS ICSF HCR770B or later • PCI X Cryptographic Coprocessor • Crypto Express2 Coprocessor

Specifying multiple RSA keys

You can specify multiple RSA control statements. Each control statement identifies the label of one RSA public key in the ICSF PKDS. CSDFILEN supports up to 16 public key labels.

For each RSA public key, Encryption Services creates an identifier that is to be associated with the key. The RSA identifier allows CSDFILDE to associate an RSA private key with the correct RSA key information in the header record of the encrypted output file. To decrypt the file that uses the RSA option, you must specify the **RSA=** control statement with the label of the RSA private key on CSDFILDE or specify the **-keyStoreCertificateAlias** parameter for the RSA private key on the Java-based Client of Encryption Facility for z/OS Client. CSDFILDE supports only one **RSA=** control statement. The statistics report file for CSDFILEN includes any **RSA=** control statements that you specify as input.

Using RSA keys and digital certificates

When you use RSA for cryptographic key management instead of PASSWORD (derived key option), digital certificates form the basis of the key exchange. The "public keys" that are stored in the ICSF PKDS are almost always derived from digital certificates. You can use your own program to store public and private RSA keys and manage certificates, but if you use RACF, the RACDCERT command allows you to perform the following functions:

- Store internal public/private RSA keys in the ICSF PKDS
- Manage PKDS labels for the keys
- Establish a limited scope certificate authority for your users

See Chapter 7, "Using RACF with Encryption Facility," on page 49.

Specifying the ICOUNT value

The iteration count (ICOUNT) in Password Based Encryption (PBE) is intended to strengthen weak passwords. The default of 16 provides reasonable security and performance if the password is robust (that is, 32 random characters). Most PBE schemes assume that you choose a weak password; thus, iteration counts of 1000 or higher are often normal.

When should I compress data for encryption?

When you plan to archive large amounts of encrypted data, you might consider compressing the data (for example, to reduce the number of tape volumes you might need).

Some tape devices make use of their own compression when you store data. By definition, data that is encrypted should not be compressible; if you can compress

encrypted data it is probably not well encrypted. You might want to compress your data before you encrypt it using the COMPRESSION option of Encryption Services.

If you plan to use the Java-based Client of the Encryption Facility for z/OS Client to decrypt z/OS data, note that it cannot decrypt data that has been compressed from the CSDFILEN batch program. On any supported z/OS platform, you can use the Decryption for z/OS Client of Encryption Facility for z/OS Client to decompress and decrypt data that has been encrypted through CSDFILEN.

CSDFILEN tries to compress input data that is 64 KB or more. If you try to compress less than that amount, the statistics report output for CSDFILEN indicates that 0 bytes of data have been compressed. If you try to decompress the data through CSDFILDE, the statistics report output for CSDFILDE indicates that 0 bytes have been expanded.

Verifying encryption files when you archive

Before you send any files that you encrypt to other systems for archiving, verify that you can decrypt the file on the same system where you encrypted it. Also, be sure to retain your keys for encrypted data especially if you plan to archive the data for a long time. See Chapter 4, “Decrypting files through CSDFILDE of the Encryption Services,” on page 33.

Using Encryption Facility and UNIX pipes

You can utilize a UNIX pipe to improve performance of Encryption Services if you have a utility that writes data to a data set and the data set is to be input to Encryption Services. Instead of using an intermediate file or data set, you can make use of a UNIX pipe to pipe the data directly to the Encryption Services. As a result, you might obtain significant performance improvements for UNIX Systems Services applications.

UNIX pipes are supported wherever you use either a fixed or undefined record format file for the pipe data set, as long as the JCL provides appropriate values for LRECL, BLKSIZE, and RECFM on the DD statement. For example, the following DD statement defines a fixed record format for a UNIX pipe that another application has created and opened for writing:

```
//SYSUT1 DD PATH='/tmp/temppipe',  
//          LRECL=4160,BLKSIZE=4160,RECFM=FB,  
//          DSNTYPE=PIPE,  
//          FILEDATA=BINARY,  
//          PATHOPTS=(ORDONLY),  
//          PATHMODE=SIRWXU
```

For complete information about UNIX pipes, see *z/OS V2R2.0 UNIX System Services User's Guide*.

Format of the header record for the CSDFILEN output file

The output of the CSDFILEN program contains the encrypted data with a header record that contains the information you need for the CSDFILDE program or Encryption Facility for z/OS Client to decrypt the data. Table 6 on page 25 shows the format of the header record:

Table 6. Format of the Header Record for the CSDFILEN Output File

Offset (Decimal)	Name of Header field	Type of data	Description
0	HEADER_EYE	Character	Eyecatcher: "HEADER" in EBCDIC.
6	HDR_VERSION	Character	Version of the header record for Encryption Facility.
8	HDR_DESC	Character	EBCDIC description (DESC keyword) of CSDFILEN input file.
72	HDRLEN	Integer	Length of entire header record (integer format).
76	HDRSALT	Character	8-byte field (salt value) used with password.
84	HDRICNT	Integer	Iteration count (ICOUNT keyword), integer format from 1 - 10000 to be used with password.
88	HDRKEYLN	Character	Modulus length (hexadecimal format from 512 – 2048) in bits of the RSA public/private key taken from the RSA keyword information.
90	HDRRSA	Character	64-byte label (RSA keyword) of the RSA public/private key in ICSF PKDS.
154	HDRICV	Character	Initialization chaining vector to be used with encryption/decryption.
170			Reserved for IBM use.
174	HDAESDES	Bit	Type of key to be used to encrypt/decrypt data: <ul style="list-style-type: none"> • x'01' use a clear TDES triple-length key • x'02' use a clear 128-bit AES key • x'03' use a secure TDES triple-length key .
175	HDRFLAGS	Bit	Bit string that indicates type of output, compression options, and format of encrypted data: <ul style="list-style-type: none"> • Bit 0 = '1'b: unused • Bit 1 = '1'b: indicates output data compressed • Bit 2 = '1' b: indicates compression dictionary is present in the encrypted data • Bit 3 = '1'b: indicates clear data is binary • Bit 3 = '0'b: indicates clear data is text (not used by z/OS) .
176	HDR_COMPVER	Character	Version of Encryption Facility compression used.
178	HDRIRECF	Bit	Input file record format: <ul style="list-style-type: none"> • Bit 0 = '1'b, Bit 1 = '0'b: Fixed • Bit 0 = '0'b, Bit 1 = '1'b: Variable • Bit 0 = '1'b, Bit 1 = '1'b: Undefined • Bit 3 = '1'b: Blocked records • Bit 5 = '1'b: ASA control character • Bit 6 = '1'b: Machine control character.
179	HDRIRECL	Integer	Input file logical record length
181	HDRIBLKS	Integer	Input file block size
185	HDRORECF	Bit	Output file record format: <ul style="list-style-type: none"> • Bit 0 = '1'b, Bit 1 = '0'b: Fixed • Bit 0 = '0'b, Bit 1 = '1'b: Variable • Bit 0 = '1'b, Bit 1 = '1'b: Undefined • Bit 3 = '1'b: Blocked records • Bit 5 = '1'b: ASA control character • Bit 6 = '1'b: Machine control character.

Table 6. Format of the Header Record for the CSDFILEN Output File (continued)

Offset (Decimal)	Name of Header field	Type of data	Description
186	HDRORECL	Integer	Length of output file logical record.
188	HDROBLKS	Integer	Output file block size.
192	HDR_KEYVAL	Integer	Encrypted data-encrypting key.
448			Reserved for IBM use.
464	HDR_RSA_CNT	Integer	Applies only when the "HEADER" is version X'0002' or greater: Number of RSA= control statements.
468	HDR_RSA	Character	Applies only when the "HEADER" is version X'0002' or greater: An array consisting of information for multiple RSA= control statements. The length is variable based on the number of RSA= control statements with each entry 344 bytes in length.
	HDR_RSA_LAB	Character	An element of HDR_RSA consisting of a 64-byte label of one of the RSA public/private keys in ICSF PKDS.
532	HDR_KEY_LN	Character	An element of HDR_RSA consisting of Modulus length (hexadecimal format from 512 - 2048) in bits of the RSA public/private key in this entry.
534		Character	Two-byte placeholder of HDR_RSA.
536	HDR_KEY_VAL	Character	An element of HDR_RSA consisting of the hexadecimal encrypted data-encrypting key. This value is encrypted by the RSA key in this entry.
792	HDR_RSA_TAG	Character	An element of HDR_RSA consisting of a hexadecimal value used for validation.
812			End of Header record.

Format of the statistics report file for CSDFILEN

The output of the statistics report file depends on whether the encrypted data has been compressed.

COMPRESSION=NO: The following example shows the output of the statistics report file from CSDFILEN when compression is not specified (COMPRESSION=NO):

```

CSDFILEN Encryption Utility 09/28/2005 (MM/DD/YYYY) 12:43:38 (HH:MM:SS)
INPUT:  DESC='DATA TO SEND TO PARTNER'
INPUT:  RSA=CCA.PVT06.INT.ENC.1024S0F
INPUT:  COMPRESSION=NO
CSDFILEN:  RSA-PUB : CCA.PVT06.INT.ENC.1024S0F
INPUT:  LRECL  121 BLKSIZE    484 RECFM FB
OUTPUT: BLKSIZE  32760
ENCRYPTION OF DATA: CLEAR      TDES KEY USING CPACF
RECORDS READ:      22,653 WRITTEN:      88
BYTES READ:        2,741,013
BYTES WRITTEN:     2,877,408 WITH HEADER AND PAD
CIPHER TIMES (IN SECONDS): HIGH:    0.001969 DATA:  306704 LOW:    0.000747 DATA: 116600
TOTAL CIPHER TIME (IN SECONDS):    0.018274 CIPHERS:      10
TOTAL ELAPSED TIME:   0:00:04.12

```

COMPRESSION=YES: The following example shows the output of the statistics report file from CSDFILEN when compression is specified (COMPRESSION=YES):

```

      CSDFILEN Encryption Utility 09/28/2005 (MM/DD/YYYY) 12:51:20 (HH:MM:SS)
INPUT: DESC='COMPRESSED DATA FOR FILE XYZ'
INPUT: CLRAES128
INPUT: PASSWORD=*****
INPUT: COMPRESSION=YES
CSDFILEN:      :
INPUT:  LRECL   121 BLKSIZE      484 RECFM FB
OUTPUT: BLKSIZE   32760
ENCRYPTION OF DATA: CLEAR      AES  KEY USING CSNBSYE
RECORDS READ:      22,653 WRITTEN:      25
BYTES READ:        2,741,013
BYTES WRITTEN:      789,376 WITH HEADER AND PAD
CIPHER TIMES (IN SECONDS): HIGH:   0.003655 DATA:  123376 LOW:   0.000880 DATA:  29968
TOTAL CIPHER TIME (IN SECONDS):  0.022709 CIPHERS:      10
TOTAL COMPRESS TIME (IN SECONDS):  0.232279 TOTAL COMPRESS BYTES:      723,320
TOTAL ELAPSED TIME:  0:00:02.43

```

If you try to compress less than 64 K of data, CSDFILEN does not compress the data, and the statistics report indicates the following for COMPRESS BYTES:

COMPRESS BYTES: 0

See “When should I compress data for encryption?” on page 23.

Multiple RSA control statements: The following example shows the output of the statistics report file from CSDFILEN that includes the maximum of 16 RSA label definitions:

```

CSDFILEN Encryption Utility 02/15/2006 (MM/DD/YYYY) 18:21:
INPUT: *-----
INPUT: *      ENC4: Multiple RSA + CLRTDES Encryption + Compression
INPUT: *-----
INPUT: DESC='Multiple RSA + CLRTDES Encryption + Compression'
INPUT: RSA=RSA.ME02.512.PRIV.KEYMGMT.CLEAR
INPUT: RSA=RSA.514.PUBLIC
INPUT: RSA=RSA.799.INTERNAL.PRIVATE.TOKEN
INPUT: RSA=RSA.ME06.513.PRIV.KMONLY
INPUT: COMPRESSION=YES
INPUT: RSA=RSA.ME06.651.PRIV.KEYMGMT
INPUT: RSA=RSA.ME06.1023.PRIV.KEYMGMT
INPUT: RSA=RSA.ME06.1024.PRIV.KMONLY
INPUT: RSA=RSA.CRT08.512.PUB.KEYMGMT.CLEAR
INPUT: RSA=RSA.CRT08.675.PRIV.KMONLY.CLEAR
INPUT: CLRTDES
INPUT: RSA=RSA.CRT08.1024.PRIV.KEYMGMT
INPUT: RSA=RSA.CRT08.1243.PRIV.KMONLY.CLEAR
INPUT: RSA=RSA.CRT08.1666.PRIV.KEYMGMT.CLEAR
INPUT: RSA=RSA.CRT08.2048.PUB.KEYMGMT.CLEAR
INPUT: RSA=RSA.CRT08.PRIV.1536.BIT.MODULUS
INPUT: RSA=RSA.ME06.PRIV.830.BIT.MODULUS
INPUT: RSA=RSA.ME06.PRIV.833.BIT.MODULUS
CSDFILEN:  RSA-PUB : RSA.ME02.512.PRIV.KEYMGMT.CLEAR
CSDFILEN:  RSA-PUB : RSA.514.PUBLIC
CSDFILEN:  RSA-PUB : RSA.799.INTERNAL.PRIVATE.TOKEN
CSDFILEN:  RSA-PUB : RSA.ME06.513.PRIV.KMONLY
CSDFILEN:  RSA-PUB : RSA.ME06.651.PRIV.KEYMGMT
CSDFILEN:  RSA-PUB : RSA.ME06.1023.PRIV.KEYMGMT
CSDFILEN:  RSA-PUB : RSA.ME06.1024.PRIV.KMONLY
CSDFILEN:  RSA-PUB : RSA.CRT08.512.PUB.KEYMGMT.CLEAR
CSDFILEN:  RSA-PUB : RSA.CRT08.675.PRIV.KMONLY.CLEAR
CSDFILEN:  RSA-PUB : RSA.CRT08.1024.PRIV.KEYMGMT
CSDFILEN:  RSA-PUB : RSA.CRT08.1243.PRIV.KMONLY.CLEAR
CSDFILEN:  RSA-PUB : RSA.CRT08.1666.PRIV.KEYMGMT.CLEAR
CSDFILEN:  RSA-PUB : RSA.CRT08.2048.PUB.KEYMGMT.CLEAR
CSDFILEN:  RSA-PUB : RSA.CRT08.PRIV.1536.BIT.MODULUS
CSDFILEN:  RSA-PUB : RSA.ME06.PRIV.830.BIT.MODULUS
CSDFILEN:  RSA-PUB : RSA.ME06.PRIV.833.BIT.MODULUS
INPUT:  LRECL   252 BLKSIZE   32760 RECFM FB
OUTPUT: BLKSIZE   32760
ENCRYPTION OF DATA: CLEAR      TDES KEY USING CCF
  RECORDS READ:      679  WRITTEN:      4
  BYTES READ:      171,108
  BYTES WRITTEN:      100,564  WITH HEADER AND PAD
CIPHER TIMES (IN SECONDS): HIGH:   0.000164 DATA:   94592 LOW:   0.000164 DATA:   94592
TOTAL CIPHER TIME (IN SECONDS):   0.000164 CIPHERS:      1
TOTAL COMPRESS TIME (IN SECONDS):   0.092435 TOTAL COMPRESS BYTES:      29,047
TOTAL ELAPSED TIME:   0:00:20.82

```

Understanding the statistics report

In the statistics report, the line that starts CIPHER TIMES (IN SECONDS): shows the longest time (HIGH) that the program takes to encipher a chunk of data and the number of bytes of clear data that are in that chunk. CIPHER TIMES (IN SECONDS) also shows the shortest time (LOW) that the program takes to encipher a chunk of data, and the number of bytes in that chunk. For example if the statistics report contains the following line:

```

CIPHER TIMES (IN SECONDS): HIGH:   0.003655 DATA:   123376 LOW:   0.000880 DATA:   29968

```

the longest amount of time taken for a single encipher is .003655 seconds to encipher a block of 123376 bytes. The shortest amount of time taken for a single encipher is .000880 seconds to encipher a block of 29968 bytes.

CSDFILEN diagnostics

CSDFILEN might write diagnostic information to the statistics report file.

The following error line begins with the characters ****ERROR**** and indicates that more than 16 **RSA=** keywords are specified. CSDFILEN discontinues processing:

```
RSA SPECIFIED MORE TIMES THAN THE ALLOWED MAXIMUM
```

Each of the following error lines begins with the characters ****ERROR**** and ends the processing of CSDFILEN:

```
DISP=MOD NOT ALLOWED
DESCRIPTION TEXT MUST BE ENCLOSED IN SINGLE QUOTES
SPECIFY ONE OF RSA/PASSWORD ONLY
ONE OF RSA/PASSWORD MUST BE SPECIFIED
ENCTDES NOT VALID WITH PASSWORD
ITERATION COUNT NOT VALID
UNEXPECTED ERROR DURING COMPRESSION (RETCODE=nn)
SYSUT1 FILE FAILED TO OPEN
SYSUT2 FILE FAILED TO OPEN
MULTIPLE ENCRYPTION TYPES SPECIFIED
PASSWORD ENTERED NOT VALID
RECORD FORMAT FOR SYSUT2 NOT VALID. RECFM(U) IS REQUIRED.
CONFLICTING LRECL, BLKSIZE, AND RECFM
PRODUCT DEREGISTRATION FAILED
```

CSDFILEN might also write the following diagnostic information (beginning with the characters ****INFO****) to the statistics report file:

```
CSNBOWH possible SAF authority violation
```

Compression warnings and errors: The following error lines occur together and begin with the characters ****WARNING****. They indicate that because of the randomness of the input data, compression does NOT reduce the size of the output file. This is a minor error. CSDFILEN disables compression, but encryption of the data continues:

```
**WARNING** MINOR ERROR OCCURRED BUILDING THE COMPRESSION DICTIONARY.
**WARNING** ENCRYPTION CONTINUING WITHOUT COMPRESSION.
```

The following error lines occur together and begin with the characters ****ERROR****. They indicate that an unrecoverable error occurred during compression. This is a major error, and CSDFILEN cannot continue processing. Turn off compression and rerun the job:

```
**ERROR** CATASTROPHIC ERROR WHILE BUILDING THE COMPRESSION DICTIONARY.
**ERROR** PROCESSING HALTED, RERUN WITHOUT COMPRESSION.
```

ICSF callable services and diagnostics: If CSDFILEN invokes an ICSF callable service and that service encounters a failure, CSDFILEN writes the following diagnostic information to the statistics report file. In this example, the service CSNDSYI was invoked and returned a return code of 8 and a reason code of X'271C'. For information about ICSF return codes and reason codes, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

```
**ERROR**CSNDSYI 08 00271C
```

Return codes for CSDFILEN

CSDFILEN can issue the following return codes (decimal values in general register 15):

Table 7. Return Codes for CSDFILEN

Return code	Meaning
0	Successful operation
8	User error
20	Physical error occurred on input file
24	Physical error occurred on output file
28	SYSPRINT file failed to open
32	SYSUT1 file failed to open
36	SYSUT2 file failed to open
50	Product registration/deregistration failed

CSDFILEN can also return the following return codes that the ICSF callable service passes back to the routine. In this case, the return code and reason codes from the ICSF service are also recorded in the statistics report file:

Return code	Meaning
4	Warning
8	Application error
12	CSF error
16	Terminating error

JCL Examples for CSDFILEN

Example 1: In this example CSDFILEN reads data from a DASD data set named LAB.MASTER DATA. It creates an encrypted version in a DASD data set named LAB.MASTER.DATA.SAFE. The SYSIN options are for a z800 or z900 processor with CCF or a z890 or z990 with PCIXCC or CEX2C, or z9 109 processor with CEX2C. The options include a description for the encrypted data, a request to generate a secure triple-length TDES key (ENCTDES) protected in the header with a public RSA key:

```
//ENCRYPT1 EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=LAB.MASTER.DATA,DISP=SHR
//SYSUT2 DD DSN=LAB.MASTER.DATA.SAFE,
// UNIT=SYSDA,DISP=(NEW,CATLG),
// SPACE=(1024,(60,10)),AVGREC=K
//SYSIN DD *
DESC='My Secure Data'
ENCTDES
RSA=ICSFEHN.RSAPUB
/*
//
```

Example 2: In this example CSDFILEN reads data from a DASD data set named LAB.MASTER DATA. It creates an encrypted version in a DASD data set named LAB.MASTER.DATA.SAFE. The SYSIN options are for a z800 or z900 processor with CCF or a z890 or z990 with PCIXCC or CEX2C, or z9 109 processor with CEX2C. The options include a description for the encrypted data, a request to generate a secure triple-length TDES key (ENCTDES) protected in the header with multiple (5) RSA keys:

```
//ENCRYPT1 EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=LAB.MASTER.DATA,DISP=SHR
//SYSUT2 DD DSN=LAB.MASTER.DATA.SAFE,
```

```
// UNIT=SYSDA,DISP=(NEW,CATLG),
// SPACE=(1024,(60,10)),AVGREC=K
//SYSIN DD *
DESC='My Secure Data'
ENCTDES
RSA=ICSFHEHN.RSAPUB
RSA=RSA.ME02.512.PRIV.KEYMGMT.CLEAR
RSA=RSA.514.PUBLIC
RSA=RSA.799.INTERNAL.PRIVATE.TOKEN
RSA=RSA.ME06.513.PRIV.KMONLY
/*
//
```

Example 3: In this example CSDFILEN reads data from an IBM standard labeled tape data set named ADRDSSU.DUMPFIL, which is a cataloged data set. In the case of multiple volumes, CSDFILEN allocates two drives to improve performance. It creates an encrypted version on another tape whose volume serial is ARCHIV. The system adds the name of the resulting data set to the system catalog. The SYSIN options are for a z800 or z900 processor with CCF or a z800, z900, or z9 109 with CPACF. The options include a description for the encrypted data, a request to generate a clear triple-length TDES key with a password. An iteration count is also specified and the data is to be compressed.

```
//ENCRYPT2 EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=(,2),DSN=ADRDSSU.DUMPFIL,DISP=OLD
//SYSUT2 DD UNIT=3590-1,DISP=(,CATLG),VOL=SER=ARCHIV,
// DSN=FILE.ARCHIVE
//SYSIN DD *
DESC='My Secure Data'
CLRTDES
PASSWORD=1509TY6E
ICOUNT=3200
COMPRESSION=YES
/*
//
```

Example 4: In this example CSDFILEN reads data from a z/OS UNIX Systems Services file named /u/Lab/experiments/test3. It creates an encrypted version on scratch tapes to be retained. The output data set name is not to be added to the system catalog. The volume serial number for the data set is indicated in the messages for the job. The SYSIN options are for a z800 or z900 with CCF or a z890, z990, or z9 109 processor with CPACF. For hardware encryption, 128-bit AES must be enabled on the processor; otherwise, software encryption occurs. For a z800 or z900 processor ICSF software performs the AES key processing. The options include a description for the encrypted data, a request to generate a clear 128-bit AES key with a password. An iteration count is also specified:

```
//ENCRYPT3 EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD PATH='/u/Lab/experiments/test3',PATHOPTS=ORDONLY
//SYSUT2 DD UNIT=3490,DISP=(,KEEP),DSN=LAB.EXP.TEST3,
// VOL=(RETAIN,,,3)
//SYSIN DD *
DESC='My Secure Data'
CLRAES128
PASSWORD=1509TY6E
ICOUNT=3200
/*
//
```

ICSF callable services for CSDFILEN

The Encryption Services invokes the following ICSF callable services for CSDFILEN. If you are using RACF or similar security product, ensure that the security administrator authorizes the Encryption Services to use the following services and any cryptographic keys that are input. For information about the ICSF callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

- CSFCKM Multiple clear key Import
- CSFENC Encipher
- CSFRNG Generate a random number
- CSFSYE Encipher using clear DES/AES key
- CSFPKE Public key encrypt
- CSFSYG Generate and wrap a symmetric key
- CSFSYX Export a symmetric key
- CSFOWH One-way hash

Encryption Services might use the following ICSF callable service without the need for authorization:

- CSFXBC Convert binary string to character

Chapter 4. Decrypting files through CSDFILDE of the Encryption Services

This chapter presents information about using the CSDFILDE batch program of the Encryption Services to decrypt data.

CSDFILDE is a batch program of the Encryption Services that supports System z file formats and decrypts encrypted file output from CSDFILEN or the Java-based Client of Encryption Facility for z/OS Client. You can specify the same options for key protection that were used when CSDFILEN or the Java-based Client of the Encryption Facility for z/OS Client encrypted the file. You can also specify an option (INFO) to provide information on the original clear key file information and the keys to be used by decryption processing.

JCL DD statements for CSDFILDE

CSDFILDE supports the following DD statements or their dynamic allocation equivalents:

Table 8. JCL DD Statements for CSDFILDE

DD statement	Description
SYSPRINT	<p>Specifies the name of the data set to which CSDFILDE writes encryption statistics and diagnostics information. It can be a sequential data set as follows:</p> <ul style="list-style-type: none">• Typically a sysout data set• DASD or tape data set• PDS or PDSE member <p>CSDFILDE sets the following values:</p> <ul style="list-style-type: none">• RECFM=FBA• LRECL=133 <p>The system selects an optimal value for BLKSIZE unless you choose to code BLKSIZE. BLKSIZE must be a multiple of 133 unless it is a sysout DD statement, but coding BLKSIZE on a sysout DD statement is not beneficial.</p>
SYSIN	<p>Specifies the source from which CSDFILDE reads control statements. It can be a sequential data set as follows:</p> <ul style="list-style-type: none">• Typically a spooled system input data set• ONE of the following specifications:<ul style="list-style-type: none">– DASD or tape data set– PDS or PDSE member <p>CSDFILDE requires the following specifications:</p> <ul style="list-style-type: none">• RECFM=F or FB• LRECL=80
SYSUT1	<p>Specifies the name of the data set that contains the encrypted data to be decrypted. This is the encrypted data from CSDFILEN or the Encryption Facility for z/OS Client. It can be a sequential data set as follows:</p> <ul style="list-style-type: none">• DASD or tape data set• PDS or PDSE member <p>If you copy the file that is the output of CSDFILEN to another z/OS file, DO NOT change the LRECL or RECFM DCB parameters.</p>

Table 8. JCL DD Statements for CSDFILDE (continued)

DD statement	Description
SYSUT2	<p>Specifies the name of the output data set that contains the decrypted data. It can be a sequential data set as follows:</p> <ul style="list-style-type: none"> • DASD or tape data set • PDS or PDSE member • z/OS UNIX Systems Services file <p>The output from CSDFILDE can also be a sysout data set (SYSOUT=x).</p> <p>DISP: You can code any of the following values for DISP on SYSUT2:</p> <ul style="list-style-type: none"> • DISP=(NEW,CATLG). The data set did not exist before this job step. After the job step, the system is to catalog and keep the data set. • DISP=(NEW,KEEP). The data set did not exist before this job step. If the new data set is not SMS-managed, the system is to keep the data set but does not catalog it so you must keep track of the volume where the data set is to reside, including disk and tape. If the new data set is SMS-managed, the system treats the data set as if you had coded CATLG. On z/OS DASD data sets are usually SMS-managed. • DISP=(NEW,PASS). The data set did not exist before this job step and a following job step determines the final disposition of the data set. • DISP=(MOD,xxxx) where the xxxx is CATLG, KEEP or PASS as described above. If the data set did not exist before this job step, the system is to allocate the space for the data set. The result is the same as if you had coded DISP=(NEW,xxxx). In this case, you must also code SPACE. <p>If the data set existed before this job step, the system uses it and QSAM adds the new records to the logical end of the existing data set. In this case, SPACE has no effect, but if more space is needed, the system uses the secondary space amount, which is a temporary override for the space. A subsequent program that appends more records if needed uses the originally-coded secondary space amount.</p> <p>RECFM and LRECL: You do not need to code RECFM or LRECL for SYSUT2. CSDFILDE assumes that you want to use the original values for RECFM and LRECL for the clear data set.</p> <p>BLKSIZE and BLKSZLIM: You do not need to specify BLKSIZE for SYSUT2. Consider the following situations:</p> <ul style="list-style-type: none"> • If the input device to CSDFILEN is a tape without a standard header label (RECFM=U, BLKSIZE=0), the output device for decryption determines the recovered block size (32K or 64K for tape or 27998 K for disk). In this case, the record format and block size of the input to decryption (RECFM=U) is unrelated to the output record format, record length, and block size. • If the input device has a standard header label and specifies a non-zero value for BLKSIZE, that value is used by CSDFILDE, even if the output does not result in optimized processing. For example, if the input device specifies a 32K tape, and the data is decrypted on disk, CSDFILDE uses 32K, even though for optimum utilization, 27998 is a better BLKSIZE value for disk.

Control statement keywords for CSDFILDE SYSIN DD

You can specify the following options in the control statement data set (identified by SYSIN DD) to control decryption of the input files. All keywords must start in column 1, and you cannot code a continuation statement. The program treats an asterisk (*) in column 1 as a comment.

All of the following keywords are optional unless the following conditions apply:

- You specified PASSWORD= as input to CSDFILEN. In this case you must supply the correct password on the SYSIN control statement.
- The label of the RSA private key to be used to decrypt the data-encrypting key is different from the label of the RSA public key used by CSDFILEN. In this case you must supply the label of the RSA private key on the JCL statement for CSDFILDE.

Table 9. Control Statement Keywords for CSDFILDE SYSIN DD

Description	JCL keyword
Method to generate and protect the data encrypting key	<p>Specifies the method that is used to generate and protect the data encrypting key. RSA and PASSWORD are mutually exclusive.</p> <p>RSA=label</p> <p>Specifies the 64-byte <i>label</i> of an RSA private key that is in the ICSF PKDS. This RSA private key that corresponds to the public key is used to decrypt the data-encrypting key that is present in the header record of the encrypted data from CSDFILEN or Encryption Facility for z/OS Client.</p> <p>If you are decrypting data that is encrypted with a single RSA= control statement and the RSA private key label is the same as the RSA public key label used to protect the data-encrypting key, the RSA keyword is optional because the RSA key label is stored in the header record. For data encrypted with multiple RSA= control statements, you must specify one RSA keyword for decryption on CSDFILDE. CSDFILDE does not allow multiple RSA keywords. See “Specifying multiple RSA keys” on page 23.</p> <p>PASSWORD=password</p> <p>Specifies the 8- to 32-EBCDIC character password to be used to regenerate the clear TDES triple-length key or the clear 128-bit AES key that is used for the data encryption. This password must match that specified as input to CSDFILEN or Encryption Facility for z/OS Client. Passwords are case sensitive.</p> <p>In order to minimize problems because of code page differences at the encrypting and decrypting sites, IBM suggests that you use only the upper and lower-case letters A through Z, numerals 0 – 9 and the underscore character (_) .</p>
Information only	<p>INFO</p> <p>Specifies that file decryption is not to be performed, but that information about the defaults that CSDFILDE establishes is to be recovered and written to the SYSPRINT file. When the information is written, CSDFILDE processing ends. This option is useful when you want to determine the original clear text file DCB information and to ensure that a specified RSA key is present in the current ICSF system.</p>

User reference information for decrypting data

Reference information includes the following topics and samples:

- “Format of the statistics report file for CSDFILDE”
- “Return codes for CSDFILDE” on page 38
- “JCL examples for CSDFILDE” on page 39

Format of the statistics report file for CSDFILDE

The output of the statistics report file depends on whether the encrypted data has been compressed.

COMPRESSION=NO: The following example shows the output of the statistics report file from CSDFILDE when compression has not been specified for the encrypted data:

```
CSDFILDE Decryption Utility 09/28/2005 (MM/DD/YYYY) 14:41:07 (HH:MM:SS)
CSDFILDE:      :
INPUT: DESC = DATA TO SEND TO PARTNER
INPUT: LRECL   121 BLKSIZE   484 RECFM FB
INPUT: PASSWORD=*****
RECORDS READ:      88 WRITTEN:      22,653
  BYTES READ:      2,877,408 BYTES RECOVERED:      2,741,013
CIPHER TIMES (IN SECONDS): HIGH:   0.001949 DATA: 294840 LOW:   0.001462 DATA: 229320
TOTAL CIPHER TIME (IN SECONDS):   0.018218 CIPHERS:      10
TOTAL ELAPSED TIME:   0:00:02.23
```

COMPRESSION=YES: The following example shows the output of the statistics report file from CSDFILDE when compression has been specified for the encrypted data:

```
CSDFILDE Decryption Utility 09/28/2005 (MM/DD/YYYY) 14:47:50 (HH:MM:SS)
CSDFILDE:  RSA-PUB : CCA.PVT06.INT.ENC.1024S0F
INPUT: DESC = DATA TO SEND TO PARTNER
INPUT: LRECL   121 BLKSIZE   484 RECFM FB
INPUT: RSA=CCA.PVT06.INT.ENC.1024S0F
RECORDS READ:      25 WRITTEN:      22,653
  BYTES READ:      789,368 BYTES RECOVERED:      2,741,013
CIPHER TIMES (IN SECONDS): HIGH:   0.001854 DATA: 294376 LOW:   0.001431 DATA: 229320
TOTAL CIPHER TIME (IN SECONDS):   0.005120 CIPHERS:      3
TOTAL EXPAND TIME (IN SECONDS):   0.037375 TOTAL EXPANDED BYTES:      2,760,344
TOTAL ELAPSED TIME:   0:00:09.57
```

If the report indicates 0 for EXPANDED BYTES, an attempt to compress data that is less than 64 K bytes probably occurred with CSDFILEN. See “When should I compress data for encryption?” on page 23

If one or more RSA keywords have been specified for the encrypted data input, and if the INFO keyword is specified as input to CSDFILDE, the output of the statistics report file from CSDFILDE includes all of the RSA key labels (in the previous example, CSDFILDE: RSA-PUB : CCA.PVT06.INT.ENC.1024S0F). If the PASSWORD keyword has been specified for the encrypted data input, and if the INFO keyword is specified as input to CSDFILDE, the output from CSDFILDE in the statistics report file shows blanks in the second line as follows:

```

CSDFILDE Decryption Utility 09/15/2005 (MM/DD/YYYY) 10:51:27 (HH:MM:SS)
CSDFILDE:      :
INPUT: DESC = UR0.B32760 INPUT
INPUT: LRECL      80 BLKSIZE   32720 RECFM FB
INPUT: *-----*
INPUT: *      INFO KEYWORD ONLY      *
INPUT: *-----*
INPUT: INFO

```

Understanding the statistics report

In the statistics report, the line that starts CIPHER TIMES (IN SECONDS) shows the longest time (HIGH) that the program takes to decipher a chunk of data and the number of bytes of clear data that are in that chunk. CIPHER TIMES (in seconds) also shows the shortest time (LOW) that the program takes to decipher a chunk of data, and the number of bytes in that chunk. For example if the statistics report contains the following line:

```

CIPHER TIMES (IN SECONDS): HIGH:   0.001854 DATA:  294376 LOW:   0.001431 DATA:  229320

```

the longest amount of time taken for a single decipher is .001854 seconds to decipher a block of 294376 bytes. The shortest amount of time taken for a single decipher is .001431 seconds to decipher a block of 229320 bytes.

In the line that starts TOTAL CIPHER TIME (IN SECONDS):, CIPHERS: indicates the number of times CSDFILDE invokes an ICSF callable service to perform decryption. In the following example, CSDFILEN invoked callable service CSNBDEC 17647 times:

```

TOTAL CIPHER TIME (IN SECONDS):   32.842984 CIPHERS:           17,647

```

The value is not the same as the number of hardware instructions executed because the hardware instruction processes a CPU-determined number of blocks , each of which are a multiple of the algorithm length (8 bytes for TDES or 16 bytes for AES). The system might require multiple invocations of the hardware instruction to process one "chunk" of data that ICSF processes.

CSDFILDE diagnostics

CSDFILDE might write diagnostic information to the statistics report file. Each of the following error lines begins with the characters **ERROR** and ends the processing of CSDFILDE:

```

CONFLICTING OUTPUT LRECL
CONFLICTING OUTPUT RECFM
EOF REACHED ON INPUT FILE BEFORE END OF HEADER
SPECIFY ONE OF RSA/PASSWORD ONLY
PASSWORD MUST BE SPECIFIED
PASSWORD ENTERED NOT VALID
INCORRECT PASSWORD ENTERED
HEADER RECORD NOT VALID
UNEXPECTED ERROR DURING EXPANSION (RETcode=nn)
SYSUT1 FILE FAILED TO OPEN
SYSUT2 FILE FAILED TO OPEN
UNSUPPORTED VERSION OF ENCRYPTED DATA
OUTPUT MUST SUPPORT QSAM LARGE BLOCK INTERFACE (LBI)
RECORD FORMAT FOR SYSUT1 NOT VALID. RECFM(U) IS REQUIRED.
PASSWORD NOT ALLOWED WITH RSA OPTION
PRODUCT DEREGISTRATION FAILED

```

The following diagnostic information (beginning with the characters ****WARNING****) indicates that the output data set (SYSUT2) of CSDFILDE has a different BLKSIZE(nnnnnn) from the input data set (SYSUT1) specified on CSDFILEN. CSDFILDE processing continues:

NEW OUTPUT BLKSIZE. REQUESTED: nnnnnn

CSDFILDE might also write the following diagnostic information (beginning with the characters ****INFO****) to the statistics report file:

CSNBOWH possible SAF authorization violation

ICSF callable services and diagnostics: If CSDFILDE invokes an ICSF callable service and that service encounters a failure, CSDFILDE writes the following diagnostic information to the statistics report file. In this example, the service CSNDPKD was invoked and returned a return code of 8 and a reason code of X'271C'. For information about ICSF return codes and reason codes, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

****ERROR****CSNDPKD 08 00271C

Corrupted compression dictionary: A system abend code of 0C7 usually means that the compressed data has not been decompressed because either the compression dictionary or the data is corrupted.

Return codes for CSDFILDE

CSDFILDE can issue the following return codes (decimal values in general register 15):

Table 10. Return Codes for CSDFILDE

Return code	Meaning
0	Successful operation
4	Warning
8	User error
20	Physical error occurred on input file
24	Physical error occurred on output file
28	SYSPRINT file failed to open
32	SYSUT1 file failed to open
36	SYSUT2 file failed to open
40	Error reading header from input file
44	Bad data found during expansion of compressed data
48	No expansion dictionary in decrypted input data
50	Product registration/deregistration failed

CSDFILDE can also return the following return codes that the ICSF callable service passes back to the routine. In this case, the return and reason codes from the ICSF service are also recorded in the statistics report file:

Return code	Meaning
4	Warning
8	Application error
12	CSF error
16	Terminating error

JCL examples for CSDFILDE

Example 1: In this example CSDFILDE reads data from a DASD data set named PARTNER.XYZ.ENCDATA. It writes the decrypted data to a DASD data set named PARTNER.XYZ.INVENTORY, which did not previously exist. The system creates and catalogs the data set. All of the defaults are used for the SYSIN options. Because the original data has been encrypted with a CLRTDES key that is protected with an RSA private key and the RSA key label is available in the header, you do not need to specify RSA= if the PKDS of the decrypting system contains the same RSA private key stored with the same label.

```
//DECRYPT EXEC PGM=CSDFILDE
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=PARTNER.XYZ.ENCDATA,DISP=SHR
//SYSUT2 DD DSN=PARTNER.XYZ.INVENTORY,
//          UNIT=SYSDA,DISP=(NEW,CATLG),
//          SPACE=(1024,(60,10)),AVGREC=K
//SYSIN DD *
/*
//
```

Example 2: In this example CSDFILDE reads encrypted data from an IBM standard labeled tape data set that is named XYZ.FILE.ARCHIV. The data set is also a cataloged data set. CSDFILDE writes the decrypted data to a new tape data set XYZ.DATA, on volume serial ARCHIV. This data set is added to the system catalog. Because the original clear data was encrypted using PASSWORD, you must specify the same password as input to the CSDFILDE program. The program retrieves the iteration count and salt value used by CSDFILDE from the header record that is contained in the SYSUT1 input file. It uses those values along with the password to recover the clear key for decryption:

```
//DECRYPT2 EXEC PGM=CSDFILDE
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=3590,DSN=XYZ.FILE.ARCHIV,DISP=OLD
//SYSUT2 DD UNIT=3590,DISP=(NEW,CATLG),VOL=SER=ARCHIV,
//          DSN=XYZ.DATA
//SYSIN DD *
PASSWORD=ASK NOT FOR WHOM THE BELL TOLLS
/*
//
```

Example 3: In this example CSDFILDE reads data from a z/OS UNIX Systems Services file named /u/Lab/encdata/partner3. It writes the decrypted data to an existing data set, PARTNER3.INVENTORY, on a 3390 DASD volume named CSDUS1. The original clear data has been encrypted with a key that is protected by an RSA public key. The RSA private key that corresponds to the public key is stored in the PKDS of this system with the label MY.PRIV.2048.KEY. Because this label is different from the label of the RSA public key that has been used for encryption, you must specify the label of the private key for RSA as input to the CSDFILDE program.

```
//DECRYPT3 EXEC PGM=CSDFILDE
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD PATH='/u/Lab/encdata/partner3',PATHOPTS=ORDONLY
//SYSUT2 DD DSN=PARTNER3.INVENTORY,
//          UNIT=3390,VOL=SER=CSDUS1,DISP=OLD
//SYSIN DD *
RSA=MY.PRIV.2048.KEY
/*
//
```

ICSF callable services for CSDFILDE

The Encryption Services invokes the following ICSF callable services for CSDFILDE. If you are using RACF or similar security product, ensure that the security administrator authorizes Encryption Services to use the following services and any cryptographic keys that are specified as input to CSDFILDE. For information about the ICSF callable services, see *z/OS Cryptographic Services ICSF Application Programmer's Guide*.

- CSFCKM Multiple clear key import
- CSFDEC Decipher
- CSFSYD Decipher using clear DES/AES key
- CSFOWH One-way hash
- CSFPKD Public key decrypt
- CSFSYI Import a symmetric key

Encryption Services might use the following ICSF callable service without the need for authorization:

- CSFXBC Convert binary string to character

Chapter 5. Using Encryption Facility for the Java-based Client

You can use the Java-based Client of Encryption Facility for z/OS Client to encrypt and decrypt System z format data on supported Java platforms. You can also use the Decryption for z/OS Client of Encryption Facility for z/OS Client on a z/OS platform to decrypt data that has been encrypted through CSDFILEN. This chapter presents information about using the Java-based Client of Encryption Facility for z/OS Client. For information about the Decryption Client for z/OS, see <http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>.

Encryption and decryption functions

The Java-based Client works with the CSDFILEN and CSDFILDE programs to perform the following functions:

- Decrypts data that is encrypted through the CSDFILEN program on z/OS
- Encrypts data that the CSDFILDE program is able to decrypt on z/OS

Installing the Java code

Before you use Encryption Facility for z/OS Client to encrypt or decrypt data on a z/OS system, ensure that you perform the following steps:

1. Download Encryption Facility for z/OS Client zip file from the Java Web site to where you want to run the program. See "Getting started with Encryption Facility for z/OS Client" on page 14.
2. Extract the zip files and place the Java source on the z/OS system.
3. Compile the source code on z/OS.

Java classes: The Java-based Client includes the following Java classes:

- EncryptionFacility (application with options to encrypt or decrypt a binary file)
- Messages (class that contains the messages for the Java-based Client)

For complete information about these classes and the use of the Java-based Client, see the documentation in the JavaDocsPublic directory within the downloadable zip file.

Using RSA keys and certificates

Using RSA keys: When you encrypt data through the Java-based Client, you can specify multiple RSA public keys through the **-keyStoreCertificateAlias** parameter.

For example, you can use two RSA **-keyStoreCertificateAlias** parameters for certificate1 and certificate2 as input for encryption:

```
-keyStoreCertificateAlias="certificate1 alias with imbedded blanks"  
-keyStoreCertificateAlias="certificate2 alias with imbedded blanks"
```

You can specify up to 16 RSA key labels for encryption.

When you use the Java-based Client to decrypt data, use the **-keyStoreCertificateAlias** parameter to specify RSA labels. The decryption process uses the public key portion of the RSA key that is specified by

-keyStoreCertificateAlias to identify the appropriate encrypted data encryption key in the header. The Java-based Client then uses the RSA private key to unwrap the data encryption key.

To use RSA keys and certificates with the Java Client you must store your keys in a Java keystore. The most common way to do this is with a utility called `keytool`. For the documentation on the Java `keytool` utility, see the following Web site:

<http://java.sun.com/j2se/1.4.2/docs/tooldocs/solaris/keytool.html>

Using certificates: Generally speaking, what you need to do with `keytool` depends on how you intend to use the Java Client.

- To send encrypted data to another system, you need the certificate of the other system. For example, the following code shows how to import a certificate (`Acertfile.cer`) from another system into your keystore:

```
keytool -import -alias sysA -file Acertfile.cer
```

- To receive encrypted data from another system, the other system needs your certificate. For example, the following code shows how to create a keypair and certificate in your keystore:

```
keytool -genkey -dname "CN=System B, OU=MyUnit, O=MyOrg, L=MyLocal, S=MyState, C=MyCountry" -alias sysB
```

The following code shows how to export your certificate (`Bcertfile.cer`) from your keystore to another system:

```
keytool -export -alias sysB -file Bcertfile.cer
```

Considerations using the Java-based Client

Consider the following uses of the Java-based Client for encrypting or decrypting z/OS format data:

Java policy files: To run the Java-based Client you need to copy the unrestricted Java policy files to your Java Virtual Machine (JVM). This allows you to use large key sizes.

Data compression: The Java-based Client cannot compress data for encryption. Also, the Java-based Client cannot process encrypted data that has been compressed by Encryption Facility on z/OS.

Encrypted data from CSDFILEN: The Java-based Client cannot decrypt data that is encrypted through encrypted keys (that is, if you have specified the ENCTDES keyword on CSDFILEN when you encrypted the data).

Data conversions between systems: Encryption Facility does not handle data conversions that need to take place between z/OS and non-z/OS systems. You must handle all data conversions (for example, adding new line characters or making ASCII, EBCDIC, and other kinds of code conversions).

Editing encrypted files on z/OS: Do not use an editor on z/OS to access a file that has been encrypted with the Java-based Client, or you might corrupt the file format.

Record format and size: You can use the Java-based Client of Encryption Facility for z/OS Client to encrypt data that meets the requirements of z/OS fixed or undefined record formats (but not variable length format). The encrypted file always has an undefined record format. Unlike CSDFILEN, the Java-based Client

does not recognize the record format attribute of a file unless you specify it. Use **-recordFormat** and **-recordSize** as follows:

- To indicate a fixed record format, specify **-recordFormat** as **FIXED** and a valid value for **-recordSize**. Allowable record sizes for **FIXED** record format are from 1 to 32760. The Java-based Client encrypts the data according to the specified record size and sets the header information to reflect that the data is in fixed record format. If the file size is not a multiple of the record size, the Java-based Client pads the last record with binary zeroes.
- To indicate an undefined record format, use the default value or specify **UNDEFINED** for **-recordFormat**. If you use the default value or specify **UNDEFINED** for **-recordFormat**, you cannot specify a value for **-recordSize**. If you specify a value for **-recordSize**, the Java-based Client indicates an exception.

Exchanging files between operating systems: If you are exchanging data between the Java file system and z/OS, you must be aware of the file characteristics (record format and length) for z/OS data. If you encrypt a file from the Java-based Client, the Java-based Client stores the specified record format information in the header. However, because the Java file system does not recognize record boundaries, the result is a data stream of bytes. If you re-encrypt the data on the Java file system and send it back to z/OS, the z/OS system cannot restore the original file characteristics because the Java-based Client does not include that information. To use **CSDFILDE** to decrypt the data from the Java-based Client, you must specify the record format and length on the **JCL**.

If **CSDFILEN** has encrypted the data, you can use **INFO** on **CSDFILDE** to display the original **LRECL** and **BLKSIZE** values. (These values are in the header of the encrypted file). If you decrypt that data on **CSDFILDE**, you do not need to specify the record format or length.

Chapter 6. Using DFSMSdss Encryption

You can use DFSMSdss Encryption to encrypt and decrypt data through DFSMSdss and DFSMSHsm commands. For complete information, see *z/OS DFSMSdss Storage Administration*.

DFSMSHsm documentation

For complete information about using DFSMSHsm commands to encrypt and decrypt data, see the following DFSMSHsm publications:

- *z/OS DFSMSHsm Storage Administration*
- *z/OS DFSMSHsm Implementation and Customization Guide*

For DFSMSdss you can use the DUMP command to encrypt an output data set and specify that the encrypted data is to reside on tape or DASD. You can specify the following options on the DUMP command:

Table 11. Options for the DUMP Command

Description	DUMP option
Encryption type	ENCRYPT Specifies information about which encryption key you want to generate. You can specify one of the following types. CLRTDES Specifies that the input file is to be encrypted with a clear TDES triple-length key in the DFSMSdss address space CLRAES128 Specifies that the input file is to be encrypted with a clear 128-bit AES key ENCTDES in the DFSMSdss address space ENCTDES Specifies that the input file is to be encrypted with a secure TDES triple-length key in the DFSMSdss address space

Table 11. Options for the DUMP Command (continued)

Description	DUMP option
Method to generate and protect the data encrypting key	<p>Specifies the method to be used to generate and protect the data encrypting key. RSA and PASSWORD are mutually exclusive. One of the following keywords is required:</p> <p>RSA(label) Specifies the 64-byte <i>label</i> of an existing RSA public key that is present in the ICSF cryptographic key data set (PKDS).</p> <p>KEYPASSWORD(password) Specifies a password between 8 and 32 characters that is used to generate a data key to encrypt the user data. If you specify KEYPASSWORD on the DUMP command, you must also specify the same KEYPASSWORD on the RESTORE command.</p> <p>IBM suggests that you use only the upper and lower-case letters A through Z, numerals 0 – 9 and the underscore character (_) .</p>
Compression option	<p>HWCOMPRESS Specifies whether you want compression of the clear input before encryption of the data occurs. If you want compression, specify the keyword HWCOMPRESS. Omit the keyword if you do not want compression.</p>

To decrypt the data from the DUMP command, you can use the RESTORE command with the following options:

Table 12. Options for the RESTORE Command

Description	RESTORE option
Method used to generate and protect the data encrypting key	<p>Specifies the method to be used to generate and protect the data encrypting key. RSA and PASSWORD are mutually exclusive:</p> <p>RSA(label) Specifies the 64-byte <i>label</i> of an existing RSA private key that is present in the ICSF PKDS. The RSA option on the RESTORE command is optional. Use RSA if you want to specify a different label for an RSA key. If you do not specify the RSA keyword on the RESTORE command, DFSMSdss uses the original label specified on the DUMP command.</p> <p>KEYPASSWORD(password) Specifies a password between 8 and 32 characters that is used to generate a data key to encrypt the user data. If KEYPASSWORD has been specified on the DUMP command, you must also specify the same KEYPASSWORD on the RESTORE command.</p> <p>IBM suggests that you use only the upper and lower-case letters A through Z, numerals 0 – 9 and the underscore character (_) .</p>

JCL examples for DFSMSdss Encryption

Example 1: This example shows the JCL for encrypting data for a full volume dump to tape:

```
//SYSIN DD *
DUMP -
    FULL -
        INDD ( -
            DC9SS01 -
        ) -
        ENCRYPT(CLRAES128) -
        HWCOMPRESS -
        RSA(CCA.CRT08.INT.ENC.1024S0F) -
        OUTDDNAME (DDTAPE1)
/*
```

Example2: This example shows the JCL for encrypting a logical dump data set and decrypting the data through the RESTORE command:

```
//SYSIN DD *
DUMP -
    DS(INCLUDE(SOURCE.**)) -
        LOGINDD ( -
            DC9SS01 -
            DC9SS02 -
        ) -
        ENCRYPT(CLRDES) -
        KEYPASSWORD(MYPASSWORD) -
        HWCOMPRESS -
        ALLDATA(*) -
        ALLEXCP -
        OUTDDNAME (DDTAPE1)
/*
//SYSIN DD *
RESTORE -
    DS(INCLUDE(SOURCE.**)) -
        OUTDYNAM ( -
            (T9SS01) -
            (T9SS02) -
            (T9SS03) -
        ) -
        KEYPASSWORD(MYPASSWORD) -
        RENAMEU(TARGET) -
        REPLACEU -
        STORCLAS(SC9TG016) -
        INDDNAME (DDTAPE1)
/*
```

Chapter 7. Using RACF with Encryption Facility

You can use RACF to help you store RSA public and private keys for encryption in the ICSF public key data set (PKDS). You can also specify the PKDS labels to use when you store public or private keys and can list PKDS labels of existing security certificates as well as establish a PKI to manage the digital certificate authority for users.

This chapter includes the following topics:

- “Using RACF to store keys, manage PKDS labels, and send digital certificates”
- “Using the RACDCERT command” on page 50

Using RACF to store keys, manage PKDS labels, and send digital certificates

Encryption Facility encrypts data for archival and recovery purposes. Long term storage of archived encrypted data helps with disaster recovery. Moreover, Encryption Facility allows you to safely transport encrypted data to interested parties on other sites where it can be decrypted or stored for future use. RACF provides public/private key support and the management of PKDS labels associated with the keys for both archiving and recovery of the encrypted data. Through RACF, you can also set up a limited scope certificate authority that allows you to exchange key information for the encrypted data.

For data archival and recovery, the encryption and decryption process can make use of the RSA public/private key pair. The public part is meant for encrypting data that can be archived, and the private part for decrypting or recovering the data.

For data transit, the sending side needs to use the recipient's public key to encrypt the data, while the receiving side uses the corresponding private key of the public/private pair to decrypt the data. The sender expects to receive the public key of the recipient in the form of an x.509 certificate, and the sender needs to store the public key, without the equivalent private key, in the ICSF PKDS.

If you are using RSA private/public keys to encrypt or decrypt data, you can make use of RACF to allow you to exchange keys with the senders or recipients of the data. This exchange involves using digital certificates to identify users and their keys. You can also use ICSF utility panels to create or delete PKDS records and export or import RSA keys to x.509 certificates. For a scenario, see “Using ICSF utilities panels for PKDS key management” on page 61. For complete information, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

RACF and certificates: RACF lets you create certificates with a private/public RSA key pair for the decryption of data sent to you by another party. To accomplish this, the sender must encrypt the data using your public key. Before encryption, this public key needs to be sent to the other party enclosed in the certificate created by RACF.

This certificate identifies you as the owner of the key pair. The other party can receive the certificate and use RACF to create a PKDS label for the public key in the other party's ICSF PKDS. In addition, if you want to encrypt data to return to

the other party, a second key exchange must occur, where the other party creates the key pair and sends you the public key certificate from that system.

Figure 4 shows how this exchange of certificates works with RSA keys that are stored in the PKDS of the sending and receiving systems. On z/OS you can use RACF to generate a digital certificate and key pair. You (the z/OS customer) send the digital certificate to a receiver (the business partner) on a remote system who can also use RACF or another program or service to create a digital certificate and key pair and send the certificate to you.

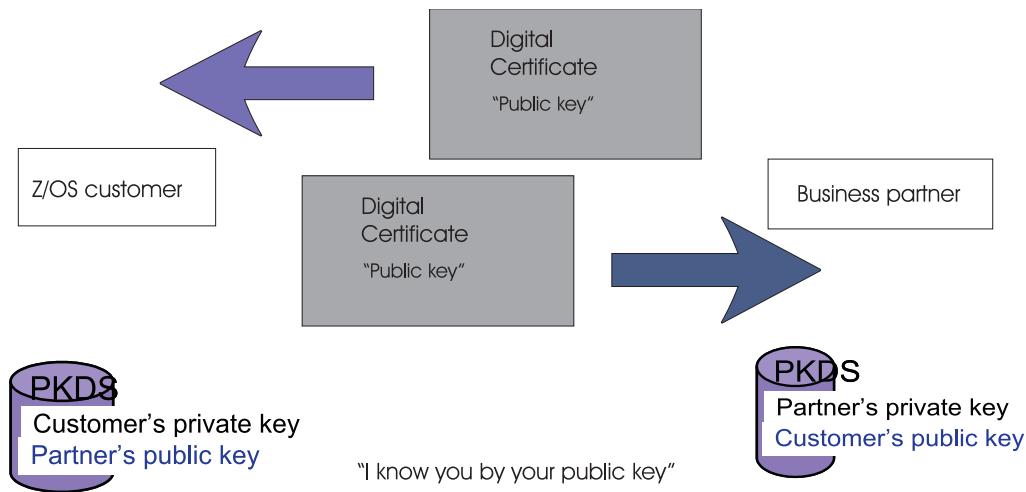


Figure 4. Establishing a trusted exchange through digital certificates

Using the RACDCERT command

RACDCERT is used to install and maintain digital certificates, key rings, and digital certificate mappings in RACF. RACDCERT should be used for all maintenance of the DIGTCERT, DIGTRING, and DIGTNMAP class profiles. For complete information, see *z/OS Security Server RACF Command Language Reference*.

For Encryption Facility, RACF provides the following support for the RACDCERT command:

- RACDCERT ADD that includes the keyword PCICC in addition to the ICSF keyword to allow user-defined PKDS labels.
- RACDCERT GENCERT PCICC, ICSF, and DSA keywords to indicate how RACF should generate the key pair and how the key should be stored for future use.
- RACDCERT REKEY PCICC and ICSF keywords to indicate how RACF should generate the key pair and how the private key should be stored for future use if the key algorithm of the original certificate is RSA.
- RACDCERT LIST that displays the digital certificate information, including certificate authority and site certificate information.

The following sample is output from the RACDCERT LIST command that shows the PKDS Label IRR.DIGTCERT.GEORGEM.SY1.BD7103108611F42F:

```
Digital certificate information for user GEORGEM:
Label: New Cert Type - Ser # 00
Certificate ID: 2QfHxdbZx8XU1YWmQMOFmaNA46iXhUBgQOKFmUB7QPDw
Status: TRUST
Start Date: 1996/04/18 03:01:13
End Date: 1998/02/13 03:01:13
Serial Number:
    >00<
Issuer's Name:
    >OU=Internet Demo CertAuth.O=TheCert Software Inc.<
Subject's Name:
    >OU=Internet Demo CertAuth.O=TheCert Software Inc.<
Private Key Type: ICSF
Private Key Size: 1024
PKDS Label: IRR.DIGTCERT.GEORGEM.SY1.BD7103108611F42F
Ring Associations:
Ring Owner: GEORGEM
Ring:
    >GEORGEMsNewRing01<
Ring Owner: GEORGEM
Ring:
    >GEORGEMsRing<
```

For a RACF scenario, see “Using the RACDCERT command for key and certificate management of encrypted data” on page 60.

For complete command syntax information, see *z/OS Security Server RACF Command Language Reference*.

Considerations using RACDCERT

RACF database and PKDS in a sysplex: If you are sharing a RACF database in a sysplex environment, you must ensure that you are also sharing the PKDS where Encryption Facility resides. For example, if you use RACDCERT GENCERT to generate an RSA key pair and you try to load the public key into the PKDS of another partition that does not share the RACF database, RACF fails the command.

Archiving RSA key data: If you use RACF to create private key entries in the PKDS for use with Encryption Facility, do not issue the following RACF commands that might delete the PKDS entries:

- RACDCERT DELETE deletes a certificate from RACF and its corresponding entry in the PKDS.
- DELUSER deletes any certificates owned by the user and any corresponding entries in the PKDS.
- RACDCERT ROLLOVER command retires a certificate's private key and deletes its entry in the PKDS

As a precaution, backup the certificate and its private key **before** you place it in the ICSF PKDS as follows:

1. Generate the certificate and key pair.
2. Export the pair to a data set in PKCS12 format (password protected).
3. Migrate the key to ICSF.

The following example shows the RACF commands you can use:

```

RACDCERT GENCERT... /* Do not specify the ICSF, PCICC, or DSA keywords */
RACDCERT EXPORT(LABEL('cert-label')) DSN(backup-data-set-name) FORMAT(PKCS12DER) PASSWORD('secret-password')
RACDCERT ADD(backup-data-set-name) PASSWORD('secret-password') ICSF(pkds-label) | PCICC(pkds-label)

```

ICSF and RACDCERT: RACDCERT processing makes use of ICSF services. If your installation has established access control over ICSF services, the issuers of RACDCERT need to be granted READ authority to ICSF services as follows:

Table 13. RACDCERT command authority and ICSF services

RACDCERT command	Keywords	ICSF Service	Comments
GENCERT or REKEY		<ul style="list-style-type: none"> CSFRNG 	Only required if ICSF is active
GENCERT or REKEY	PCICC	<ul style="list-style-type: none"> CSFPKG CSFPKX 	
GENCERT or REKEY	ICSF or PCICC	<ul style="list-style-type: none"> CSFPKRC CSFPKRR CSFPKRW 	
GENCERT	SIGNWITH	<ul style="list-style-type: none"> CSFDSG 	Only required if the signing certificate has an ICSF or PCICC key
DELETE		<ul style="list-style-type: none"> CSFPKRD 	Only required if the certificate being deleted has an ICSF or PCICC key

Chapter 8. User scenarios

The following scenarios show some ways that you can use Encryption Facility to encrypt and decrypt files and RACF to perform certificate exchange.

- “Encrypting data using z/OS and decrypting using the Java-based Client”
- “Using the RACDCERT command for key and certificate management of encrypted data” on page 60
- “Using ICSF utilities panels for PKDS key management” on page 61

Also, see “JCL Examples for CSDFILEN” on page 30, “JCL examples for CSDFILDE” on page 39, and “JCL examples for DFSMSdss Encryption” on page 47.

Encrypting data using z/OS and decrypting using the Java-based Client

The following scenarios describe encrypting files on z/OS and decrypting the files on the Java-based Client.

Scenario 1

In this scenario the JCL for CSDFILEN on z/OS specifies that the data is to be encrypted using a password and a clear 128-bit AES key (CLRAES128). An iteration count (ICOUNT) of 9 is also specified. On z/OS, the device type is a 3390, the data set to be encrypted is a member of a partitioned data set (PO) with a record format of FB, record length of 80, and a block size of 23440. The user saves the encrypted file in the hierarchical file system (HFS) that an xSeries system can access.

```

//TESRENC JOB ((ENCRYPT,UC),'ZEF.TEST',
//          NOTIFY=,MSGLEVEL=(1,1),MSGCLASS=H,
//          REGION=4M
//*)
//*****
//*
//* FOLLOWING JOB TAKES DATA FROM ONE DSN AND ENCRYPTS IT
//* AND PLACES THE RESULT IN ANOTHER DATASET
//*
//*
//*****
//*-----
//*          ENCRYPT
//*          (ONLY MANDATORY ENTRY IS EITHER RSA OR PASSWORD)
//*
//*-----
//ENC      EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//*-----
//STEPLIB DD DSN=ICSF7730.LINKLIB,DISP=SHR
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
DESC='ENCRYPTION, PW length of 9 Mixedcase, fb data to java x-series'
PASSWORD=pw0FNINE9
ICOUNT=9
CLRAES128
/*
//SYSUT1   DD DSN=TSTFLR6.FXT.CLIST(AMIOUT),DISP=SHR
//SYSUT2   DD DSN=PAYROLL.G118.FB80.T0JEFF.SEP29T1,
//          UNIT=SYSDA,DISP=(NEW,CATLG),
//          SPACE=(1024,(60,10))
//*)

```

The user on the z/OS system copies the encrypted file to the HFS with the file name PAYROLL.ZSERIES.ENCRYPTD (not shown).

On the Java statements for the Java-based Client, the user on the xSeries system specifies the same password used to encrypt the data to perform decryption as follows. In this scenario Encryption Facility for z/OS Client automatically detects the key used for the encryption and the iteration count:

```

java -Djava.encryption.facility.debuglevel=0 \
com.ibm.encryptionfacility.EncryptionFacility \
-mode decrypt \
-password "pw0FNINE9" \
-inputFile PAYROLL.ZSERIES.ENCRYPTD \
-outputFile PAYROLL.XSERIES.DECRYPTD

```

Scenario 2

In this scenario, two files are encrypted through CSDFILEN on a z/OS system. The files are encrypted using a CLRTDES data-encryption key. One file protects the data-encryption key with an RSA public key extracted from the ICSF PKDS. The other file uses a password to generate the data-encryption key used. The encrypted files are sent to a system where the Java-based Client decrypts the files.

Using RSA to encrypt on z/OS: CSDFILEN encrypts a file using a CLRTDES key data-encrypting key. The data-encrypting key is then encrypted with an existing RSA public key that is retrieved from ICSF PKDS with the label specified by RSA=.

```
//*-----
//ENC      EXEC PGM=CSDFILEN
//*-----
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*-----
//SYSPRINT DD DSN=USERID.ZEF.LISTING,DISP=(MOD,PASS)
//*-----
//SYSUT1   DD DSN=USERID.ZEF.CLR,DISP=SHR
//SYSUT2   DD DSN=USERID.ZEF.RSA,ENC,DISP=SHR
//SYSIN    DD *
*-----*
*   RSA key encryption on z                               *
*-----*
DESC='TDES ENCRYPTED FILES USING 1023 BIT RSA KEY '
RSA=IRR.DIGTCERT.ENICHEN.SYS1.BDA7284CD8F33AB2
CLRTDES
/*
```

Using PASSWORD to encrypt on z/OS: CSDFILEN encrypts a file using a CLRTDES data-encrypting key. The data-encrypting key is generated using the specified password.

```
//*-----
//ENC      EXEC PGM=CSDFILEN
//*-----
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*-----
//SYSPRINT DD DSN=USERID.ZEF2.LISTING,DISP=(MOD,PASS)
//*-----
//SYSUT1   DD DSN=USERID.ZEF2.CLR,DISP=SHR
//SYSUT2   DD DSN=USERID.ZEF2.PW.ENC,DISP=SHR
//SYSIN    DD *
*-----*
*   PASSWORD encryption on a z                             *
*-----*
DESC='TDES ENCRYPTED FILES USING Password '
PASSWORD=ABCD1234
CLRTDES
/*
```

Using RSA keys and certificates on the Java-based Client: To use RSA keys and certificates with the Java-based Client, you must store your keys in a Java keystore. The most common way to do this is with a utility called `keytool`. To transfer the RSA public key between systems, you must use X.509 certificates. For information about using the Java-based Client, see the README file documentation from the following Web site: <http://www.ibm.com/servers/eserver/zseries/zos/downloads/#asis>.

Decrypting the file protected with RSA: The recipient of the encrypted file copies the file into `zef.rsa.enc`. Encryption Facility for z/OS Client is used to decrypt the RSA protected file. The data-encrypting key is recovered using an RSA private key from the Java keystore. The `KEYPW` on the export command specifies a password used to recover the RSA key:

```

export MODE='decrypt'
export KEYSTORETYPE='jks'
export KEYSTORENAME='/home/g1a5445/edar/test/keystores/keystore_jks'
export ALIAS_1='peggyrac'
export KEYPW='xyz12abc'

java com.ibm.encryptionfacility.EncryptionFacility \
-mode $MODE \
-password $KEYPW \
-keyStoreType $KEYSTORETYPE \
-keyStoreName $KEYSTORENAME \
-keyStoreCertificateALIAS $ALIAS_1 \
-inputFile zef.rsa.enc \
-outputFile zef.rsa.dec

```

Decrypting the file protected with PASSWORD: The recipient of the encrypted file copies the file into zef.pw.enc. Encryption Facility for z/OS Client is used to decrypt the password protected file. The data-encrypting key is regenerated using the password provided in the Java option: -password \$PASSWORD. The password is the same password that was used to encrypt the original file.

```

export MODE='decrypt'
export PASSWORD='ABCD1234'

java com.ibm.encryptionfacility.EncryptionFacility \
-mode $MODE \
-password $PASSWORD \
-inputFile zef.pw.enc \
-outputFile zef.pw.dec

```

Scenario 3

In this scenario, a z/OS system that has access to a UNIX Systems Services (USS) file system creates an encrypted data set on tape and copies the encrypted data set to a USS file. USS then invokes a batch job for Encryption Facility for z/OS Client to decrypt the file to another USS file and copies the decrypted USS file to a z/OS data set.

Encrypting data using z/OS and placing the encrypted data in a tape data set: The following JCL for CSDFILEN encrypts the z/OS data set HLQ.INPUT.DATASET and places the output into a tape data set. CSDFILEN uses a password (TEST1PASSWORD) to encrypt the data and places the encrypted data in the data set HLQ.ENCRYPTED.TAPE.DATASET:

```

//ESE2TAPE JOB 'JOB INFORMATION','TEST',MSGLEVEL=(1,1),
// MSGCLASS=H,CLASS=A,NOTIFY=system_id,REGION=4M
//*****
/** This is sample JCL which can be used to encrypt data and place *
/** it into a tape dataset using the Encryption Services provided *
/** by the IBM Encryption Facility for z/OS *
/** *
/** This job uses the CSDFILE batch program of the Encryption *
/** Services to encrypt the data to tape. The SYSUT1 DD statement *
/** specifies the name of the dataset that contains the data to *
/** be encrypted. The SYSUT2 DD statement contains the name of the *
/** tape dataset that will contain the encrypted data. *
/** *
/** Refer to the IBM Encryption Facility for z/OS: User's Guide for *
/** additional information. *
//*****
//ENC EXEC PGM=CSDFILEN
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
PASSWORD=TEST1PASSWORD
//SYSUT1 DD DSN=HLQ.INPUT.DATASET,DISP=SHR
//SYSUT2 DD UNIT=3590-1,DSN=HLQ.ENCRYPTED.TAPE.DATASET,
// DISP=OLD,VOL=SER=ABCDEF
/*

```

Copying the data to USS and using batch services to invoke Encryption Facility for z/OS Client: The JCL performs the following steps:

1. In STEP1 the IEBGENER copy program on z/OS copies the encrypted data set HLQ.ENCRYPTED.TAPE.DATASET from the tape and places it into a USS file called /filesys/input.encrypt.file on the USS file system.
2. In STEP2 a BPXBATCH program on USS runs the shell script javadecrypt_pw.sh to invoke Encryption Facility for z/OS Client that decrypts the data in the USS file /filesys/input.encrypt.file. The JCL for BPXBATCH specifies that the decrypted file is to be placed into another USS file /filesys/output.decrypt.file.
For the sample script that Encryption Facility for z/OS Client uses to decrypt the data, see Running the Encryption Facility for z/OS Client sample script.
3. In STEP3 IEBGENER copies the decrypted USS file /filesys/output.decrypt.file from STEP2 to a z/OS data set HLQ.DECRYPT.OUTPUT.

```

//JCD2DSN JOB 'JOB INFORMATION',MSGLEVEL=(1,1),
// MSGCLASS=H,CLASS=A,NOTIFY=system_id
//*****
/** This is sample JCL which can be used to receive an encrypted dataset on a tape and perform *
/** decryption using the Encryption Facility for z/OS JAVA client. The resulting decrypted data *
/** is placed into a z/OS dataset. *
/** *
/** Overview of job steps *
/** - STEP1 - this will use IEBGENER to copy an encrypted dataset from a tape and place it *
/** into a file within the USS file system. *
/** - STEP2 - this will use the USS file created by STEP1 and invoke BPXBATCH to run a shell *
/** script named javadecrypt_pw.sh to decrypt the data into another USS file within a file *
/** system. *
/** - STEP3 - this will use IEBGENER to copy the decrypted file from the USS file system and *
/** place it into an z/OS data set. *
/** *
/** File, dataset and JOB considerations: The originating encrypted dataset is assumed to be *
/** of Fixed Block (FB) Record Format (RECFM). *
/** *
/** - All steps within this job specify the REGION= parameter. *
/** This is critical to STEP2 to ensure that JAVA has sufficient memory to execute and decrypt *
/** the file. *
/** *
/** - The user must ensure that there is sufficient space available within the USS file system *
/** to contain the encrypted file (which came from tape) and the decrypted file (which was the *
/** output of the EF JAVA decryption client). *
/** *
/** - The customer's client (receiver of the encrypted data) must know the data set attributes *
/** (BLKSIZE,RECFM,LRECL) of the original (clear) source dataset. This information is required *
/** for STEP3 of this job. It is specified on STEP3 SYSUT1 DD in this sample. *
/** *
/** - The customer's client (receiver of the encrypted data) must also know the size of the *
/** original (clear) source dataset. This information is required for STEP3 of this job. It is *
/** specified on STEP3, SYSUT2 DD for the SPACE value. *
/** *
/** Note: USS refers to Unix System Services *
/** *
//*****
//*****
/** STEP1 - This jobstep will copy an encrypted file from a tape dataset to an HFS file within *
/** Unix System Services. The HFS is automatically created via the OCREAT specified on the *
/** PATHOPTS keyword. The PATHDISP keyword will keep the file active across to the next step. *
/** Refer to the Unix System Services User's Guide for additional explanation on PATHOPTS, *
/** PATHMODE and PATHDISP. *
//*****
//STEP1 EXEC PGM=IEBGENER,REGION=6M
//SYSUT1 DD UNIT=3590-1,DSN=HLQ.ENCRIPTED.TAPE.DATASET,
// DISP=(SHR),VOL=SER=ABCDEF
//SYSUT2 DD PATH='/filesys/input.encrypt.file',
// PATHOPTS=(OWRONLY,OCREAT),
// PATHMODE=SIRWXU,
// PATHDISP=KEEP
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
/**

```

```

/*****
/* STEP2 - This jobstep will use BPXBATCH to invoke a USS shell script which invokes the EF JAVA *
/* decryption client.The JAVA decryption client uses STDIN as input to specify the USS file *
/* containing the encrypted data. The JAVA decryption client uses STDOUT as the output USS file *
/* into which decrypted data will be placed. *
/* *
/* The shell script is named javadecrypt_pw.sh and the names of the files specified in STDIN and *
/* STDOUT DDs must match the input and output file specifications in the shell script. The PARM *
/* specified on BPXBATCH assumes that the shell script resides in the USS filesystem in *
/* a path specified by /path. *
/* *
/* STDERR specifies a USS file into which error messages from the JAVA decryption *
/* client will be placed. *
/* *
/* Repeating from the comments above, REGION=0M is critical on STEP2 to ensure that JAVA has *
/* sufficient storage to decrypt the input file. *
*****/
//STEP2 EXEC PGM=BPXBATCH,REGION=0M,
// PARM='SH /path/javadecrypt_pw.sh'
//STDIN DD PATH='/filesys/input.encrypt.file',
// PATHOPTS=(ORDONLY),
// PATHMODE=SIRWXU,
// PATHDISP=KEEP
//STDOUT DD PATH='/filesys/output.decrypt.file',
// PATHOPTS=(OWRONLY,OCREAT),
// PATHMODE=SIRWXU,
// PATHDISP=KEEP
//STDERR DD PATH='/path/stderr.log',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
/*****
/* STEP3 - This jobstep will copy a decrypted file from a USS file within the file system to a *
/* z/OS dataset via IEBGENER. The BLKSIZE, LRECL, and RECFM on SYSUT1 will drive the attributes *
/* that will be used on SYSUT2 when it is created. *
/* *
/* The space required to handle the decrypted output in the z/OS dataset must be consistent with *
/* the original (clear) source dataset that came from the originator partner. In this sample, *
/* the dataset required 300 cylinders on a 3390. *
/* *
*****/
//STEP3 EXEC PGM=IEBGENER,REGION=6M
//SYSUT1 DD PATH='/filesys/output.decrypt.file',
// BLKSIZE=6144,LRECL=1024,RECFM=FB,
// PATHOPTS=(ORDONLY),
// PATHMODE=SIRWXU,
// PATHDISP=KEEP
/*
//SYSUT2 DD DSN=HLQ.DECRYPT.OUTPUT,
// DISP=(NEW,CATLG),
// SPACE=(CYL,(2,1)),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY

```

Running the Encryption Facility for z/OS Client sample script: STEP2 of the JCL specifies the shell script javadecrypt_pw.sh that invokes Encryption Facility for z/OS Client. The following sample script shows how Encryption Facility for z/OS Client uses the password TEST1PASSWORD to decrypt the USS file /filesys/input.encrypt.file:

```
# Sample script which may be invoked via z/os Batch process, IE: BPXBATCH
# PATH statement needs to specify the absolute path to the java 'bin' directory.
# CLASSPATH must include the java encryption java client's absolute path and name.
# IE: /path/efclient.zip
# it is important to 'cd' to the directory where the efclient.zip file resides.
# Please note: This example utilizes decryption via PASSWORD.
# This example uses '/filesys/input.encrypt.file' as it's encrypted data source,
# which is also specified in BPXBATCH job as STDIN.
# This example uses '/filesys/output.decrypt.file' for expected decrypted output,
# which is also specified in BPXBATCH job as STDOUT.
# Important: Ensure the 'REGION=0M', parm is included in the BPXBATCH job step,
# to ensure the java virtual machine
# has sufficient storage available.
#
export PATH=/usr/lpp/java/cur13secure/bin:.$PATH
export CLASSPATH=/feu/efclient.zip:.$CLASSPATH
cd /path
java -Djava.encryption.facility.debuglevel=0 \
com.ibm.encryptionfacility.EncryptionFacility \
-mode decrypt \
-password TEST1PASSWORD \
-inputFile /filesys/input.encrypt.file \
```

Using the RACDCERT command for key and certificate management of encrypted data

In this scenario you want to create a new certificate with a 2048 bit RSA public/private key pair for an encrypted data set that is sent from a remote site. You plan to send the certificate to the remote site so the recipient can create a PKDS label and store the certificate in the ICSF PKDS on the remote system:

1. Use the RACDCERT command on your site to create the PKDS label for the key pair:

```
RACDCERT GENCERT SUBJECTSDN(CN('Sally's Data Encryption')) WITHLABEL('Sally's Data Encryption') SIZE(2048)
PCICC NOTAFTER(DATE(2020/08/10))
```

2. Use the RACDCERT LIST command to view the name of the PKDS label that RACF has created:

```
RACDCERT LIST(LABEL('Sally's Data Encryption'))
```

Digital certificate information for user SALLY:

```
Label: Sally's Data Encryption
Certificate ID: 2QfHxdbZx8XUaqweQM0FmaNA46iXhUBgQ0KFmUB7QPDw
Status: TRUST
Start Date: 2005/08/11 00:00:00
End Date: 2020/08/10 23:59:59
Serial Number:
>00<
Issuer's Name:
>CN=Sally's Data Encryption<
Subject's Name:
>CN=Sally's Data Encryption<
Private Key Type: PCICC
Private Key Size: 2048
PKDS Label: IRR.DIGTCERT.SALLY.SY1.BD7103108611F42F
```

3. To send the certificate you need to extract it from RACF. Use the RACDCERT EXPORT command to extract it from RACF:

```
RACDCERT EXPORT(LABEL('Sally's Data Encryption')) DSN(FOR.JACOB.CRT)
```

4. Send the certificate to the remote site so the recipient can return encrypted data. (You can use e-mail, FTP, or whatever program your installation uses to send the certificate to the remote site. Note that you do not send the private key to the recipient so the certificate does not need to be protected.)
5. The recipient at the remote site receives the certificate that you sent in step 4 into a data set (SALLY.CRT in this example). The recipient needs the public key from this certificate to send you encrypted data. The recipient adds the certificate to the RACF data base as a SITE certificate, and gives it the name "Sally". The command also creates a PKDS label with the same value "Sally."

```
RACDCERT SITE ADD(SALLY.CRT) WITHLABEL('Sally') ICSF(*)
```

6. From the remote site the recipient uses the following RACDCERT command to list out the certificate that has been added to RACF. Note that RACF changes PKDS label text to uppercase:

```
RACDCERT SITE LIST(LABEL('Sally'))
```

Digital certificate information for SITE:

```
Label: Sally
Certificate ID: egljcv8XUaqweQMOfmaNA46iXhUBgQ0KFmUB7QPDw
Status: TRUST
Start Date: 2005/08/11 00:00:00
End Date: 2020/08/10 23:59:59
Serial Number:
>00<
Issuer's Name:
>CN=Sally's Data Encryption<
Subject's Name:
>CN=Sally's Data Encryption<
Private Key Type: None
PKDS Label: SALLY
```

Using ICSF utilities panels for PKDS key management

This scenario shows how to use the ICSF utilities panels to manage PKDS keys.

1. From the ICSF primary options utility panel, select option 5, UTILITY and press ENTER. You receive the ICSF utilities panel CSFUTL00:

```
----- ICSF - Utilities -----

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
 6 PKDSKEYS    - Manage keys in the PKDS

Press ENTER to go to the selected option.
Press END to exit to the previous menu.

OPTION ==>
```

2. From CSFUTL00 select option 6: PKDSKEYS and press enter. You receive the ICSF PKDS keys panel CSFPKY00.

```

----- ICSF - PKDS Keys -----
Enter the PKDS record's label for the actions below
==>

Select one of the following actions then press ENTER to process:

- Generate a new PKDS key pair record
  Enter the key length ==>          512, 1024, or 2048
  Enter Private Key Name (optional)
  ==>

- Delete the existing public key or key pair PKDS record

- Export the PKDS record's public key to a certificate data set
  Enter the DSN ==>
  Enter desired subject's common name (optional)
  CN=

- Create a PKDS public key record from an input certificate.
  Enter the DSN ==>

COMMAND ==>

```

CSFPHY00 allows you to perform the following PKDS key management:

- Generate an RSA key pair PKDS record
- Delete an existing PKDS record
- Export an existing public key to an x.509 certificate
- Import a public key from an x.509 certificate

Coprocessor Requirements for using the ICSF utility panels: To use the full function of the ICSF utility panels, you must have a PCICC, PCIXCC, or a CEX2C cryptographic coprocessor. If you do not have one of these coprocessors, you cannot generate key pairs using the panels.

For example, to generate a new PKDS RSA key pair, enter the PKDS record label (MY.PKDS.KEY.PAIR) and the key length in bits. You can also specify an optional private key name that ICSF can imbed into the key token. This service creates an RSA public/private key PKDS record. You can use the key pair that ICSF generates to encrypt and recover archive data. You can also use it to recover encrypted data that another party transmits to you if you make the public key portion available to the partner:

```

----- ICSF - PKDS Keys -----
Enter the PKDS record's label for the actions below
==>MY.PKDS.KEY.PAIR

Select one of the following actions then press ENTER to process:

- Generate a new PKDS key pair record
  Enter the key length ==> 1024      512, 1024, or 2048
  Enter Private Key Name (optional)
  ==>

- Delete the existing public key or key pair PKDS record

- Export the PKDS record's public key to a certificate data set
  Enter the DSN ==>
  Enter desired subject's common name (optional)
  CN=

- Create a PKDS public key record from an input certificate.
  Enter the DSN ==>

COMMAND ==>

```

If the system successfully generates the new key pair record, you receive confirmation panel CSFPKY01:

```

----- ICSF - PKDS Key Request Successful -----

Label ==> MY.PKDS.KEY.PAIR

Key function completed successfully

Press ENTER or END to return to the previous menu.

COMMAND ==>

```

For complete information about using the ICSF utilities panels, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

Appendix. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (<http://www.ibm.com/support/knowledgecenter/SSLTBW/welcome>).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the "Contact us" web page for z/OS (<http://www.ibm.com/systems/z/os/zos/webqs.html>) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS V2R2 ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out

punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element

The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element

The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the

default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**

The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

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.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted

for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (<http://www.ibm.com/software/support/systemsz/lifecycle/>)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This book is a user's guide to Encryption Facility.

This book primarily documents information that is NOT intended to be used as a Programming Interface of Encryption Facility.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Caution: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

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

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

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

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

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

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

Index

Numerics

128-bit AES keys 1

A

accessibility 65
 contact IBM 65
 features 65
assistive technologies 65

C

certificates
 use with RACF 50
 use with the Java-based Client 41
CLRAES128 keyword
 for CSDFILEN encryption 20
 for DFSMSdss Encryption 45
CLRTDES keyword
 for CSDFILEN encryption 20
 for DFSMSdss Encryption 45
compressing data
 diagnostics 29
 guidelines 23
COMPRESSION keyword 21
contact
 z/OS 65
cryptographic keys
 determining which to use for
 encryption 21
 ICSF 12
 use with IBM Encryption Facility for
 z/OS 1, 12
CSDFILDE
 batch job for decryption 33
 callable services for 40
 DD statements for decrypting
 files 33
 diagnostics 37
 JCL examples 39
 JCL keywords 33
 keywords for decrypting files 34
 return codes 38
 statistics report file 36
CSDFILEN
 batch job for encryption 17
 callable services for 32
 DD statements for encrypting
 files 17
 diagnostics 29
 header file format 24
 JCL examples 30
 JCL keywords 19
 keywords for encrypting files 19
 return codes 29
 statistics report file 26

D

decryption
 CSDFILDE JCL 33
 DFSMSdss Encryption 45
 Encryption Services 4
 Java-based Client 41
 overview of CSDFILDE 4
 scenario 53
Decryption Client for z/OS
 overview 5
DESC keyword 19
DFSMSdss Encryption
 encryption and decryption for 45
 feature of IBM Encryption Facility for
 z/OS 1
 JCL examples 47
 JCL keywords 45
 overview 6
 software requirements 11
 using DFSMSHsm with 6
DFSMSHsm 1
diagnostics
 CSDFILDE 37
 CSDFILEN 29
DUMP command for encryption with
 DFSMSdss Encryption 45

E

ENCRYPT keyword 45
encrypting and decrypting data
 overview 1
 scenarios 53
encryption
 CSDFILEN JCL 17
 DFSMSdss Encryption 45
 Encryption Services 4
 Java-based Client 41
 overview of CSDFILEN 4
 scenario 56
Encryption Facility 1
Encryption Facility for OpenPGP
 overview 4
 software requirements 8
Encryption Facility for z/OS Client
 Decryption Client for z/OS 5
 encryption and decryption for 41
 function of IBM Encryption Facility
 for z/OS 5
 installing 41
 Java-based Client 5
 overview 2, 5
 software requirements 9
Encryption Services
 CSDFILDE batch program 4
 CSDFILEN batch program 4
 function of IBM Encryption Facility
 for z/OS 1
 installing 11
 overview 2

Encryption Services (*continued*)
 software requirements 8
ENCTDES keyword
 for CSDFILEN encryption 20
 for DFSMSdss Encryption 45

H

hardware requirements 7
HWCOMPRESS keyword 46

I

IBM Encryption Facility for z/OS
 cryptographic keys 12
 features 1
 hardware and software
 requirements 7
 installation 11
 overview 1
 UNIX pipes 24
 user scenarios 53
ICOUNT keyword
 description 21
 specifying value for 23
ICSF
 callable services and diagnostics for
 CSDFILDE 38
 callable services and diagnostics for
 CSDFILEN 29
 cryptographic keys 12
 getting started with 12
 utility panels 13
ICSF utility panels
 coprocessor requirements 13
 description 13
 scenario 61
INFO keyword 35
Integrated Cryptographic Service
 Facility 12

J

Java-based Client
 considerations using 42
 overview 5
 using RSA keys and certificates 41
JCL
 DD statements for decrypting files
 with CSDFILDE 33
 DD statements for encrypting files
 with CSDFILEN 17
 examples for DFSMSdss
 Encryption 47
 examples of DD statements for
 decrypting files with CSDFILDE 39
 examples of DD statements for
 encrypting files with CSDFILEN 30
 keywords for decrypting files with
 CSDFILDE 34

JCL (*continued*)

keywords for DFSMSdss

Encryption 45, 46

keywords for encrypting files with

CSDFILEN 19

K

keyboard

navigation 65

PF keys 65

shortcut keys 65

KEYPASSWORD keyword

for decryption with DFSMSdss

Encryption 46

for encryption with DFSMSdss

Encryption 46

keys 21

N

navigation

keyboard 65

Notices 69

P

PASSWORD keyword

determining when to use 22

for CSDFILDE decryption 35

for CSDFILEN encryption 20

R

RACDCERT command

considerations using 51

description 50

RACF

enhancements for IBM Encryption

Facility for z/OS 49

RACDCERT command 50

scenario to manage keys and

certificates 60

sending trusted certificates 50

software requirements 10

storing keys, managing PKDS labels,

and sending digital certificates 49

RESTORE command for decryption with

DFSMSdss Encryption 46

RSA keyword

determining when to use 22

for CSDFILDE decryption 35

for CSDFILEN encryption 20

for decryption with DFSMSdss

Encryption 46

for encryption with DFSMSdss

Encryption 46

private tokens and cryptographic

hardware for decryption 22

specifying multiple keywords for

CSDFILEN 23

using digital certificates with 23

RSA parameter for the Java-based

Client 41

S

scenarios

copying z/OS encrypted data to USS

and invoking Encryption Facility for

z/OS Client 57

encrypting and decrypting USS

files 56

encrypting data using z/OS and

decrypting using the Java-based

Client 53

encrypting data using z/OS and

placing the encrypted data in a tape

data set 56

running the Encryption Facility for

z/OS Client sample script to decrypt

data 59

using ICSF utilities panels for PKDS

key management 61

using the PASSWORD keyword 53

using the RACDCERT command for

key and certificate management of

encrypted data 60

using the RSA keyword 55

sending comments to IBM xi

shortcut keys 65

software requirements 8

DFSMSdss Encryption 11

Encryption Facility for OpenPGP 8

Encryption Facility for z/OS Client 9

IBM Encryption Facility for z/OS 8

RACF 10

statistics report file

CSDFILDE 36

CSDFILEN 26

storing keys, managing PKDS labels, and

sending digital certificates

using RACF 49

T

TDES triple-length keys 1

U

UNIX system services 24

user interface

ISPF 65

TSO/E 65

USS

scenario 56

using UNIX pipes with

CSDFILEN 24

V

verifying encryption files 24



Product Number: 5655-P97

Printed in USA

SA23-1349-04

