

Linux on Z and LinuxONE

*Enterprise Key Management
for Pervasive Encryption of Data Volumes*



Note

Before using this document, be sure to read the information in [“Notices” on page 43](#).

This edition applies to the zkey utility as included in s390-tools version 2.15.1, and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2021.**

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

Contents

About this document.....	V
Who should read this publication.....	v
Terms and abbreviations.....	v
Other publications.....	vi
Chapter 1. Introduction.....	1
Installing EKMF Web.....	2
How EKMF Web works with zkey.....	2
Software and hardware prerequisites.....	3
Chapter 2. Setting up key templates.....	5
Creating pervasive encryption key templates.....	5
Creating an identity key template.....	7
Registering EKMF Web key templates with zkey.....	9
Chapter 3. Setting up zkey.....	11
Connecting zkey with EKMF Web.....	11
Configuring a key management system plug-in.....	11
Chapter 4. Using zkey with EKMF Web.....	17
Generating keys.....	17
Changing key properties.....	18
Listing keys.....	19
Renaming a key.....	20
Refreshing keys.....	21
Chapter 5. Sharing an EKMF Web key with another system.....	23
Chapter 6. Changing the CCA master key.....	27
Chapter 7. Recovering a secure key repository.....	29
Appendix A. zkey kms - Managing secure keys with a KMS plug-in.....	31
Accessibility.....	41
Notices.....	43
Trademarks.....	43
Index.....	45

About this document

Learn how to integrate the keys used for pervasive encryption in the EKMF Web enterprise key management system by using the zkey utility in a Linux instance on Z or LinuxONE.

This publication describes how to manage the HSM-protected keys used with protected-key dm-crypt in an enterprise setting. Using the zkey utility, you can generate and manage keys in EKMF Web enterprise key management system and import these keys into the zkey repository on your Linux instance. The publication describes the setup required on both the Linux instance and in EKMF Web. It also describes how to use EKMF Web to generate a key for a new volume, recovering a zkey repository, and how to efficiently process a master key replacement.

You can find the latest version of this document on the IBM® Knowledge Center at:

https://www.ibm.com/support/knowledgecenter/en/linuxonibm/liaaf/lnz_r_ck.html

Who should read this publication

For the EKMF Web administrator, this publication explains how to set up key templates for use by zkey.

For the security administrator who is responsible for pervasive encryption with zkey and EKMF Web, this publication gives insights into how keys are managed, and what you can do to optimize your workflow.

Further, this publication is interesting for Linux administrators who are responsible for storage configuration, as well as security engineers who design an enterprise key management process.

This publication assumes that you:

- Have Linux® on Z or LinuxONE administrator skills.
- Are familiar with security concepts.

Terms and abbreviations

EKMF Web for Linux on Z uses the terms and abbreviations listed here.

CA

Certificate authority. A trusted entity, external or internal to your organization, that issues digital certificates. Digital certificates are data files used to prove the identity of a website, person, or device by certifying the ownership of a public key by the named subject of the certificate.

CBC

Cipher block chaining. A method of reducing repetitive patterns in ciphertext by performing an exclusive-OR operation on each 8-byte block of data with the previously encrypted 8-byte block before it is encrypted.

CCA mode

Common Cryptographic Architecture. Crypto Express adapters can work in different modes, whereof CCA is one.

CSR

Certificate signing request. An electronic message that an organization sends to a CA to obtain a certificate. The request includes a public key and is signed with a private key; the CA returns the certificate after signing with its own private key.

dm-crypt

dm-crypt is the device mapper crypto target of the Linux kernel crypto target. It is a disk encryption subsystem, and is part of the device mapper infrastructure.

EKMF

IBM Enterprise Key Management Foundation. EKMF provides centralized key management for IBM cryptographic products on multiple platforms.

EKMF Web

IBM Enterprise Key Management Foundation Web is a web application that you use to manage keys on IBM Z and LinuxONE systems.

HSM

Hardware security module. A tamper-protected cryptographic device that protects master keys from being inspected. IBM Crypto Express CCA coprocessors and EP11 coprocessors are certified as HSMs. Each domain of an IBM Crypto Express coprocessor constitutes a virtual HSM and maintains a domain-specific master key or a set of master keys.

HSM master key

Each domain of a Crypto Express cryptographic coprocessor can contain active master keys which are used to generate secure keys.

LUKS2

Linux Unified Key Setup version 2 is used for disk encryption management.

protected key

A protected key is a key encrypted by a firmware master key.

secure key

A secure key is a key encrypted by an HSM master key.

TLS

Transport Layer Security. A set of encryption rules that uses verified certificates and encryption keys to secure communications over the Internet. TLS is an update to the SSL protocol.

zkey repository

An access-controlled list of secure keys managed by the zkey utility.

Other publications

These publications might be of interest.

Publications for Linux distributions

For Linux on Z documents that are adapted to a particular distribution, see one of the following web pages:

- SUSE Linux Enterprise Server documents at

www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_suse.html

- Red Hat® Enterprise Linux documents at

www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_rh.html

- Ubuntu Server documents at

www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_ubuntu.html

These publications are available on IBM Knowledge Center at www.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_lib.html

- *Device Drivers, Features, and Commands*
- *Using the Dump Tools*
- *KVM Virtual Server Quick Start*, SC34-2753
- *KVM Virtual Server Management*, SC34-2752
- *How to use FC-attached SCSI devices with Linux on z Systems®*, SC33-8413
- *Introducing IBM Secure Execution for Linux*, SC34-7721
- *libica Programmer's Reference*, SC34-2602
- *Exploiting Enterprise PKCS #11 using openCryptoki*, SC34-2713

- *Secure Key Solution with the Common Cryptographic Architecture Application Programmer's Guide*, SC33-8294
- *Pervasive Encryption for Data Volumes*, SC34-2782
- *How to set an AES master key*, SC34-7712
- *Troubleshooting*, SC34-2612
- *Kernel Messages*, SC34-2599
- *How to Improve Performance with PAV*, SC33-8414
- *How to Set up a Terminal Server Environment on z/VM®*, SC34-2596

EKMF Web publications

The following EKMF Web publications are available:

- *IBM Enterprise Key Management Foundation - Web Edition: EKMF Web UI Configuration and Operation Guide*, SC28-2022
- *IBM Enterprise Key Management Foundation - Web Edition: EKMF Web UI User's Guide*, SC28-2023
- *IBM Redbooks: Key Management Deployment Guide using the IBM Enterprise Key Management Foundation*, SG24-8181

For details about EKMF Web, see the web page at:

<https://www.ibm.com/products/enterprise-key-management-foundation-web>

Chapter 1. Introduction

Key management is an essential aspect of managing secure and resilient systems. The IBM Enterprise Key Management Foundation (EKMF) provides real-time, centralized secure management of keys and certificates.

Data protection is often driven by industry regulations. However, compliance with regulations is only the minimum protection level. One of the most effective ways to protect data is to encrypt it. With the encryption of data, encryption keys become the most sensitive pieces of data. Therefore, special care must be taken in managing such keys since both their disclosure and their loss can have harmful effects on the enterprise. The control of such keys is assigned to special experts (security officers) and they must follow guidelines that are imposed by the enterprise, or other regulatory bodies.

Linux on IBM Z® and IBM LinuxONE offers pervasive encryption for data volumes for data at-rest through **dm-crypt** using secure keys that are managed with the help of the **zkey** utility.

Key management on the enterprise level

On Linux, a secure key repository controlled by the **zkey** utility can be used to manage secure keys to encrypt Linux volumes. To encrypt volumes, you use **dm-crypt** with the protected-key cipher paes. The encryption keys are protected by a Crypto Express adapter. The **zkey** command manages those keys locally on the system it runs.

To manage keys across your enterprise from a central location, not separately on each system, you can use the **zkey** utility with the key-management system plug-in interface. Key-management systems can provide a plug-in and thus can integrate themselves into the **zkey** utility. The **zkey** utility does not need to know any details about the key-management system, it just uses the plug-in to interface with the key-management system.

Using an enterprise-wide key management system includes these benefits:

- You can centrally manage the full lifecycle of cryptographic keys, including creation, access, maintenance, decryption, and destruction.
- It helps ensuring industry compliance. As compliance rules change, a centralized system that adheres to a standardized key management policy that is implemented across the organization can simplify updates.
- You can share keys across Linux instances, if these instances share encrypted volumes.

EKMF Web plug-in for Linux on IBM Z and LinuxONE

A key-management system plug-in for EKMF Web is available for Linux on IBM Z and LinuxONE. This allows the **zkey** utility to integrate with EKMF Web. Thus, EKMF Web can manage keys used by multiple Linux instances to encrypt volumes. The Linux instance must have access to a cryptographic adapter.

With EKMF Web for Linux, security is emphasized. Keys are never exposed in plain text, and are only available to authorized parties, where both the key protection and the authentication of authorized parties is secured by HSMs.

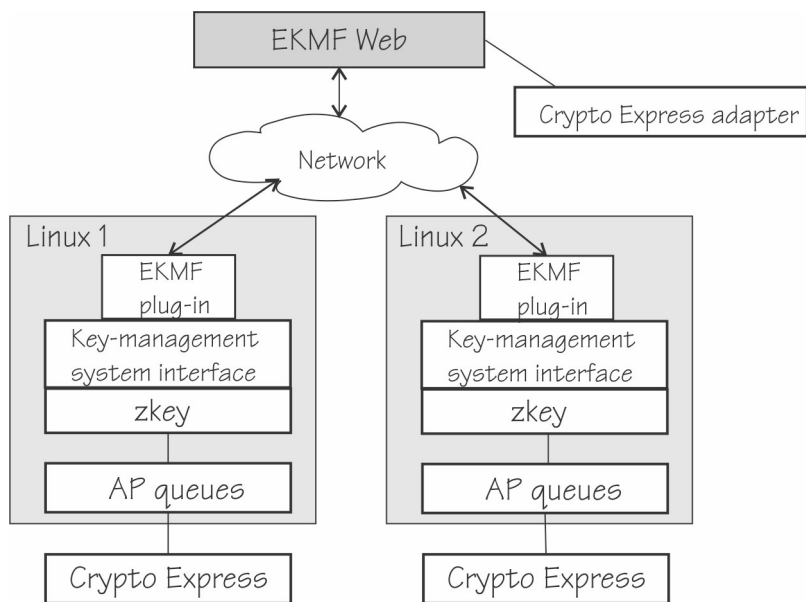


Figure 1. EKMf Web integrates through a plug-in with zkey on Linux

Figure 1 on page 2 illustrates how key management on multiple Linux instances, as z/VM guests, KVM virtual machines, or in LPAR mode, can be handled from a single key-management system.

Installing EKMF Web

Installing entails setting up a Web server for the EKM Web interface, a database for the keys, cryptographic adapters, and user accounts.

For details about the installation, see the *EKMF Web UI Configuration and Operation Guide*, SC28-2022.

How EKMF Web works with zkey

On EKM Web, templates for zkey keys must be prepared and then registered with zkey.

Cryptographic keys are the most important objects in a key management system. Apart from being the managed objects, keys are also used to establish a secure communication channel between the secure key repository and the EKMF Web server.

To use different key types for different purposes, these key types must be declared as *key templates* on the EKMF Web server.

On zkey, you must bind the zkey instance to EKMF Web.

These concepts are illustrated in Figure 2 on page 3.

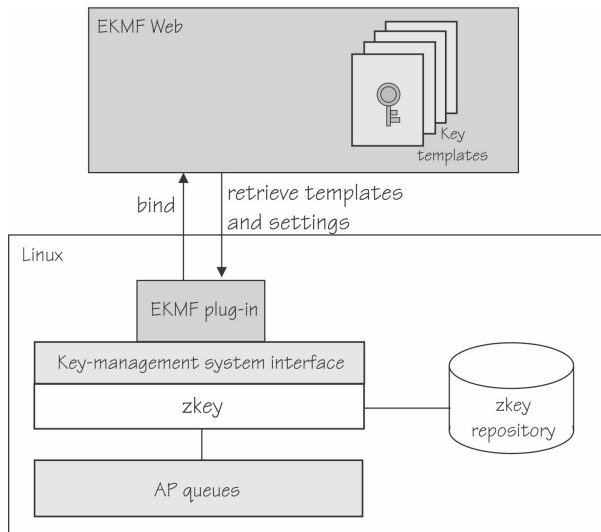


Figure 2. zkey and EKMF Web

Software and hardware prerequisites

Deploying an EKMF Web and pervasive encryption solution on Linux on Z or LinuxONE requires minimum levels of hardware and software on Linux.

For software and hardware prerequisites for EKMF Web, see the *EKMF Web UI Configuration and Operation Guide*, SC28-2022.

Hardware prerequisites

- IBM Z hardware as of IBM z13, or any LinuxONE system with the CPACF feature installed.
- A Crypto Express6S or later configured in CCA coprocessor mode.
- Volumes to be encrypted (for example, SCSI or DASD volumes). For DASD volumes, you can encrypt partitions only, not the complete DASD.
- The AES and APKA master keys must be set using the TKE (or the `panel.exe` program in a test environment).

Software prerequisites

- Linux kernel upstream version 5.4 or later for the support of secure keys of type CCA-AESCIIPHER. Older versions where the required modules have been back-ported might also work.
- The **cryptsetup** utility version 2.0.3 or later is required to configure an encrypted volume.
- The **zkey** utility from the s390-tools package, as of upstream version 2.15.1, that contains the enhancements for EKMF Web.
- The CCA 6.0 package or later from the [software-package selection page](#).

Access rights

The zkey user ID that is to be used for generating keys requires the following access rights:

- EKMF Web roles keys:export, keys:generate, keys:pre_activation:activate, keys:write, keys:active:install. See *EKMF Web UI Configuration and Operation Guide*, SC28-2022 for a full list of roles.
- In RACF, role-specific profiles must be defined. For each role, in the EJBROLE class and all user IDs must have READ access to the profiles corresponding to their required level of access. See *EKMF Web UI Configuration and Operation Guide*, SC28-2022 for examples.

Chapter 2. Setting up key templates

As an EKMF Web administrator, you need to set up key templates for use with zkey on EKMF Web for your organization, and register those templates with zkey.

About this task

A one-time setup is needed for key templates. The templates can then be used by any number of systems with zkey for key management.

You must define four key templates for zkey on EKMF Web:

- non-XTS - For volume encryption with an AES key in CBC mode.
- XTS part 1 - For volume encryption with AES in XTS mode, first part of an XTS key
- XTS part 2 - For volume encryption with AES in XTS mode, second part of an XTS key
- Identity key - Identifies the zkey system. This type of key is used for verifying the identity of zkey to EKMF Web.

By creating your own templates, you can decide on naming schemes and key properties such as the size of the key. For each template, define the algorithm, the size, the type, the state, and allow the key to be exported.

Creating pervasive encryption key templates

As an EKMF Web administrator, you must set up key templates for use with zkey on EKMF Web for your organization.

About this task

For pervasive encryption, you need to define three key templates:

- non-XTS - For volume encryption with an AES key in CBC mode.
- XTS part 1 - For volume encryption with AES in XTS mode, first part of an XTS key.
- XTS part 2 - For volume encryption with AES in XTS mode, second part of an XTS key.

The two XTS templates must have the same properties.

Procedure

To create non-XTS, XTS1, and XTS2 key templates, follow these steps:

1. Log in to EKMF Web.
2. Go to **Administration** in the left navigation bar.
3. Click **Key templates**.
4. On the panel that opens, click the **Create** button on the right.
5. Select key type **Pervasive Encryption** from the drop-down menu.
6. In the **Name** field, enter the template name.

The templates names can consist of up to 30 uppercase alphabetic characters, numerals, and hyphens. For example, assuming you want to remember that these key templates are for non-XTS zkey keys, the name can be ZKEY-NONXTS. An example of the first part of a template for the first part of an XTS key is show in [Figure 3 on page 6](#)

Figure 3. Create new key template panel, part 1

7. In the **Key label** field, enter the pattern of the key names.

All keys that are generated with this template have a name that follows this pattern.

For example, assuming you want the keys to be named as the template and then have sequential numbering, enter:

ZKEY . NONXTS . <seqno>

8. Select AES for the key algorithm. Only AES keys are eligible for pervasive encryption.
9. Select 256 for the key size.
10. Select Cipher for key type. For zkey, only cipher keys are possible.
11. For **Key state**, select Active.
12. Set **Allow key export** to on.

This setting allows you to transfer the key to zkey. An example for the second part of a template for a key is shown in [Figure 4](#) on [page 7](#).

IBM Enterprise Key Management Foundation

Key details

Key algorithm

☒ AES

Key size

256

Key type

☒ Cipher *i* ☐ Data *i*

Key state

☒ Active ☐ Pre-activation

Allow key export (optional)

☒ On

Figure 4. Create new key template, part 2

13. Optional: Set the key's active period.
14. Click **Save**.

Results

The template is created and shown on the **Key Template** page. The example template would be listed as ZKEY-NONXTS.

What to do next

Repeat the steps to create the non-XTS, XTS1, and XTS2 key templates. For information on how to create the identity key template, see [“Creating an identity key template” on page 7](#).

Creating an identity key template

As an EKMF Web administrator, you need to set up an identity key template for use with zkey on EKMF Web for your organization.

Procedure

To create an identity key template, follow these steps:

1. Log in to EKMF Web.
2. Go to **Administration** in the left navigation bar.
3. Click **Key templates**
4. On the window that opens, click the **Create** button on the right.
5. Select the key type **Identity** from the drop-down menu.
6. In the **Name** field, enter the template name.
The templates names can consist of up to 30 uppercase alphabetic characters, numerals, and hyphens. For example, assuming you want to remember that these identity key templates are for zkey keys, the name can be ZKEY-ID.
7. In the **Key label** field, enter the pattern of the key names. In contrast to the name, the label can contain full stops, but no hyphens.

All keys that are generated with this template have a name that follows this pattern.

For example, assuming you want the keys to be named similar to the name, remember that it is an elliptic curve key, and have sequential numbering, enter:

```
ZKEY.ID.EC.<seqno>
```

An example for the first part of a template for an identity key is shown in [Figure 5 on page 8](#).

The screenshot shows the 'Create new template' form in the IBM Enterprise Key Management Foundation. The left sidebar contains a navigation menu with 'Key management', 'Datasets', 'Administration', 'Key templates' (selected), 'Settings', 'Audit log', 'About', and 'API'. The main form area is titled 'Create new template' and contains 'Template details'. The 'Keystore type' is set to 'Identity'. The 'Name' field contains 'ZKEY-ID'. The 'Key Label (optional)' field contains 'ZKEY.ID.EC.<seqno>'. The 'Description (optional)' field contains 'Additional information about this key'.

Figure 5. Create new identity key template, part 1

8. Select elliptic curve (ECC) or RAS for the key algorithm.
9. Select 521 for identity keys to use a prime 521 curve (ECC), or one of the supported RSA key sizes.
10. For **Key state**, select Active.

An example for the second part of a template for an identity key is shown in [Figure 6 on page 8](#).

The screenshot shows the 'Key details' form in the IBM Enterprise Key Management Foundation. The left sidebar contains a navigation menu with 'Key management', 'Datasets', 'Administration', 'Key templates', 'Settings', 'Audit log', 'About', and 'API'. The main form area is titled 'Key details' and contains 'Key algorithm' (ECC selected), 'Key size' (521 selected), 'Key state' (Active selected), and 'Allow key export (optional)' (Off selected).

Figure 6. Create new identity key template, part 2

11. Keep **Allow key export** off.
12. Optional: Set the key's active period.
13. Click **Save**.

Registering EKMF Web key templates with zkey

The key templates that you created must be made known to zkey.

Procedure

1. On EKMF Web, go to **Administration** in the left navigation bar.
2. Click **Settings** in the left navigation
3. Enter the template names.

Scroll down to find entry fields for:

- XTS Key Template Name (Key1)
- XTS Key Template Name (Key2)
- Non-XTS Key Template Name
- Identity Key Template Name

Using the example from the key templates, you enter ZKEY-NONXTS in the **Non-XTS Key Template Name** field.

4. Click **Save**.

Example

In the screen capture that is shown in [Figure 7 on page 9](#) all four templates are registered.

IBM Enterprise Key Management Foundation

Key management

Datasets

Administration

Key templates

Settings

Audit log

About

API

EKMF.WEB.SECRETS.KEY

EKMF Web HMAC Key Label

EKMF.WEB.HMAC.KEY

EKMF Web Identity Key Label

EKMF.WEB.IDENTITY.KEY

EKMF Web Adapter Serial Number

813BDB64

XTS Key Template Name (Key 1)

ZKEY-XTS1

XTS Key Template Name (Key 2)

ZKEY-XTS2

Non-XTS Key Template Name

ZKEY-NONXTS

Identity Key Template Name

ZKEY-ID

Save

Figure 7. Entering the template names on the **Settings** page

Chapter 3. Setting up zkey

As a zkey user, you need to connect zkey to EKMF Web and configure the EKMF plug-in.

Connecting zkey with EKMF Web

After the EKMF Web administrator set up key templates and registered them for use with zkey, you can connect zkey to EKMF Web.

Before you begin

You need a Linux instance with the zkey utility installed. This instance of zkey must be able to connect to EKMF Web.

You need access to EKMF Web, that is, a user ID must be set up for you. For more details about access rights, see [“Access rights” on page 3](#).

About this task

On your Linux system, use the **zkey** command to bind to the EKMF Web plug-in and configure it.

Procedure

1. Optional: Check which plug-ins are available.
Issue the **zkey kms plugins** command, for example:

```
# zkey kms plugins
KMS-Plugin          Shared library
-----
EKMFWeb             zkey-ekmfweb.so
```

The example shows that the EKMF Web plug-in is available. It also shows the shared library that implements the plug-in.

2. To bind to the EKMF Web plug-in, issue the following command:

```
# zkey kms bind EKMFWeb
```

The local secure key repository is bound to a key management system of type EKMF Web.

Note: A secure key repository can be bound to only one key management system.

Configuring a key management system plug-in

After binding a key management system to the repository, the EKMF plug-in must be configured.

Before you begin

You require an EKMF Web user ID. For more details about access rights, see [“Access rights” on page 3](#).

APKA and AES master keys must be in place on the AP queues that your Linux instance uses. Master keys are set through the TKE.

About this task

Use a command of this form to configure the plug-in:

```
# zkey kms configure <config_option>
```

The **zkey kms configure** command supports plug-in-specific options that the plug-in provides. To see which plug-in-specific options a plug-in provides or requires, use the command:

```
# zkey kms configure --help
```

You can supply all configuration options at once, or use the **zkey kms configure** command several times, supplying only one or a few configuration options each time. A useful command might be **zkey kms info**, which displays information about the key-management system to which zkey is bound. For example:

```
# zkey kms info
KMS-Plugin:      EKMFWeb
Supported key types: CCA-AESGCM
APQNs:          (configuration required)
....
```

In the configuration phase, use the **info** command to see what must still be configured. The preceding example shows that APQNs, that is, AP queues, must be configured.

You must associate AP queues with the key management plug-in. AP queues perform secure key operations for the plug-in. Keys that are generated with the key management plug-in are automatically associated with these AP queues. If a plug-in supports different key types, for example CCA and EP11 keys, then at least one of the matching AP queues with this type must be associated. The EKMF Web plug-in supports only CCA type keys.

An overview of the setup process is shown in [Figure 8 on page 12](#).

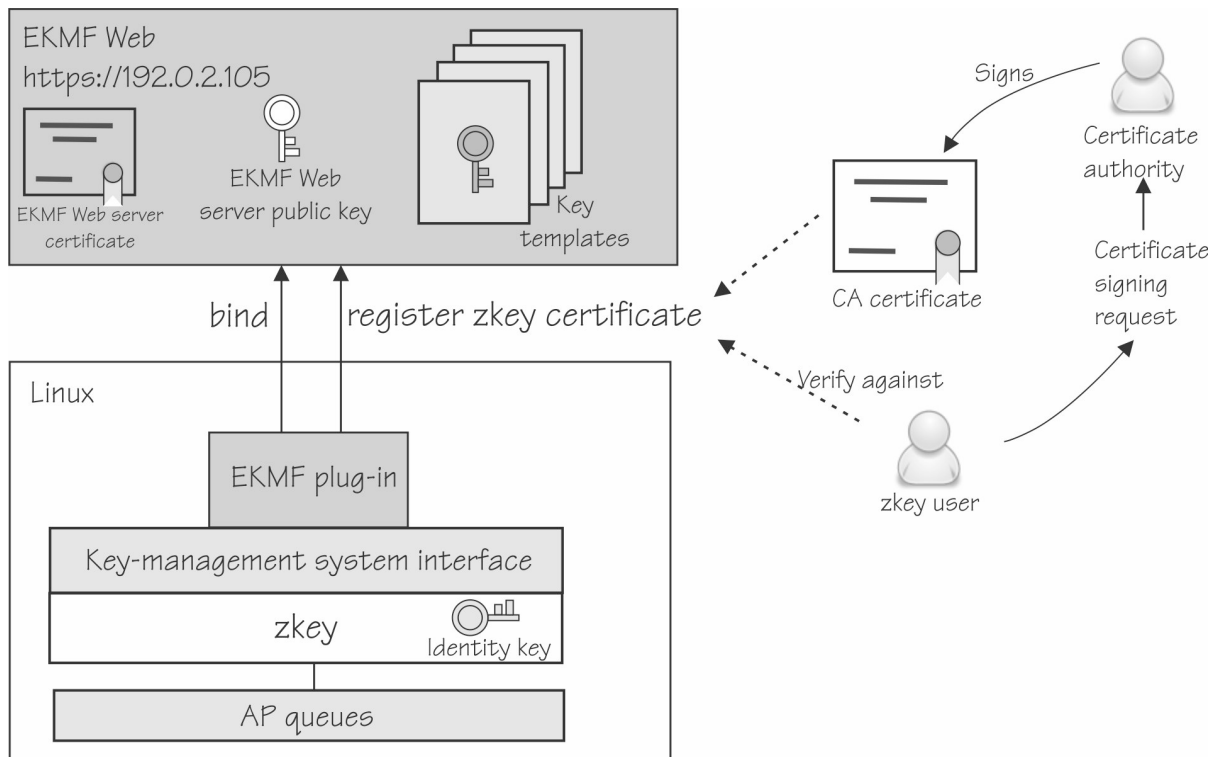


Figure 8. Exchanging configuration information between EKMF Web and zkey

Procedure

Specify the following settings.

1. To specify the AP queues to associate with the EKMF Web plug-in, use a command of the form:

```
# zkey kms configure --apqns <adapter1.domain1,adapter2.domain2,...>
```

The AP queue is defined by its adapter ID and domain ID. If you specify multiple AP queues, they must have the same master key.

For example, to add AP queues from adapter 08 and 09, both with domain 002f, issue:

```
# zkey kms configure --apqns 08.002f,09.002f
```

To see whether the AP queues are configured, use **zkey kms info**:

```
# zkey kms info
KMS-Plugin:      EKMFWeb
Supported key types: CCA-AESCIPIHER
APQNs:          08.002f
                09.002f
EKMF Web server: (configuration required)
```

In the next step, tell zkey which EKMF Web server to use.

2. Connect to the EKMF Web server.

Use a command of the form:

```
# zkey kms configure --ekmfweb-url <URL>
```

The URL must start with `https://`, and can contain a port number that is separated by a colon. If no port number is specified, 443 is used for HTTPS. You can specify TLS-specific options to control the behavior of the TLS protocol and the validation of the EKMF Web server's certificate, see [Appendix A, "zkey kms - Managing secure keys with a KMS plug-in,"](#) on page 31.

When the connection to the EKMF Web server is established, the EKMF Web settings, such as the EKMF Web server's public key and the key templates that are used by EKMF Web to generate keys are automatically retrieved from EKMF Web.

Tip: If the settings change in EKMF Web at a later time, use the `--refresh-settings` to refresh the settings in zkey.

```
# zkey kms configure --ekmfweb-url https://192.0.2.105:30140/
The EKMF Web server presented the following certificate to identify itself:
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 942492814 (0x382d4c8e)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=us, O=ibm, OU=ekmf-web, CN=localhost
    Validity
      Not Before: Mar 17 11:37:54 2021 GMT
      Not After : Mar 17 11:37:54 2022 GMT
    Subject: C=us, O=ibm, OU=ekmf-web, CN=localhost
    Subject Public Key Info:
  ....
zkey: Is this the EKMF Web server you intend to work with [y/N]? y
```

Carefully check the certificate to ensure that you are connected to the correct server. For example, have the administrator of the web server send you the server certificate through a different method, and compare that certificate with what the **zkey kms configure** command gives you.

3. Log in to the EKMF Web server with the user ID given to you by the administrator.

```
zkey: EKMF Web user ID:
```

Go to the web browser as instructed and collect a one-time passcode for login. Using passcodes avoids sending passwords in the clear over the connection.

Now the zkey instance knows which EKMF Web server to work with. Using **zkey kms info** again, the example configuration now recognizes the key templates for this zkey instance:

```
# zkey kms info
KMS-Plugin:          EKMFWeb
Supported key types: CCA-AESIPHER
APQNs:              08.002f
                   09.002f
EKMF Web server:    https://192.0.2.105:30140
CA-bundle:          System's CA certificates
Client certificate:  (none)
Client private key:  (none)
EBMF Web public key: ECC (secp521r1)
Key templates:
  Identity:         ZKEY-ID
  Label template:   ZKEY.ID.EC.<seqno>
  XTS-Key1:         ZKEY-XTS1
  Label template:   ZKEY.XTS1.<seqno>
  XTS-Key2:         ZKEY-XTS2
  Label template:   ZKEY.XTS2.<seqno>
  Non-XTS:          ZKEY-NONXTS
  Label template:   ZKEY.NONXTS.<seqno>
Identity key:       ECC (secp521r1)
Registered key label: (registration required)
```

Also, an identity key that is used to identify the zkey client to EKMF Web is automatically generated locally using the properties that are defined in the identity key template from EKMF Web.

In the next step, generate a certificate that identifies this zkey instance to EKMF Web.

4. Generate a certificate with which to register the zkey client with EKMF Web. The zkey utility automatically uses the identity key from the last step when generating the certificate.

The registration certificate is an X.509 certificate that is generated with the secure identity key as follows:

- a. Use the `--gen-csr` option to generate a certificate-signing request (CSR) with the identity key. For example:

```
# zkey kms configure --gen-csr csr.pem --cert-subject "CN=Example Name,O=Myorg,C=us"
```

- b. Pass the resulting CSR to a certificate authority (CA) to have it issue a CA-signed certificate for the EKMF Web plug-in.

Alternative for test setups: You can use the `--gen-self-signed-cert` option to generate a self-signed certificate with the identity key for the EKMF Web plug-in. Self-signed certificates should be used for testing only. Use the `--cert-subject` to specify the certificate subject name, and optionally the `--cert-extensions` option to specify extensions.

To renew an existing certificate, use the `--renew-cert` option. The subject name and extensions are then read from the certificate.

5. Register the zkey client with EKMF Web.

- a) Find out whether the identity key template uses label tags other than the `<seqno>` tag.

Registering the client automatically generates an identity key on EKMF Web with the public key from the certificate. If the identity key template in EKMF Web uses label tags other than the sequential numbering, `<seqno>`, you must specify them using the `-T` option when you register the client. To find out what the label tags are, use the following command:

```
# zkey kms info
```

- b) Register the client with EKMF Web.

You receive a certificate file from the CA or from the self-signed certificate process. To use this certificate to register with EKMF Web, use a command of the form:

```
# zkey kms configure --register <cert_file>
```

For example, assuming the certificate file is called `cert.pem`:

```
# zkey kms configure --register cert.pem
```

Using **zkey kms info** again, you can see that no more configuration steps are open, and this instance of zkey is registered with EKMF Web with the key in the **Registered key label** field, for example:

```
# zkey kms info
...
Registered key label: ZKEY.ID.EC.00001
```

This key identifies this instance of zkey to EKMF Web. The certificate contains the public key of the zkey instance so that EKMF Web now knows the public key.

Results

Now zkey and EKMF Web are set up, and you can generate keys.

Chapter 4. Using zkey with EKMF Web

With the zkey utility you can perform all the tasks on EKMF Web to manage your keys including generating, listing, changing properties, and removing keys.

Before you begin

The tasks described in the following assume that the zkey repository is bound to EKMF Web.

About this task

The following topics provide details about the tasks:

- [“Generating keys” on page 17](#)
- [“Changing key properties” on page 18](#)
- [“Listing keys” on page 19](#)
- [“Renaming a key” on page 20](#)
- [“Refreshing keys” on page 21](#)

Generating keys

Keys are generated in EKMF Web, and stored in the zkey repository. Properties that you define for a key, such as the description or the volume, are transferred to EKMF Web.

Before you begin

You need to know:

- The volumes that you want to encrypt.
- The type of key you want to generate.

About this task

EKMF Web cannot import existing zkey keys. Keys that were generated locally before the repository was bound to EKMF Web are marked as local, and can be used only on the Linux instance on which zkey runs.

Procedure

1. Optional: Find out whether the key template uses label tags other than <seqno>.

If the key template in EKMF Web uses label tags other than the sequential numbering, <seqno>, you must specify them using the -T option when you generate the key. To find out what the label tags are, use the following command:

```
# zkey kms info
```

2. Use the **zkey generate** command. You must specify a name. Issue a command of the form:

```
# zkey gen --name <name>
```

By default, the **zkey gen** command generates the new key in EKMF Web and imports it into the zkey repository.

To generate a key on the local zkey repository only, use the --local option. Local keys cannot be imported into EKMF Web.

Keys that are generated in EKMF Web are always of type CCA-AES/CIPHER. The cryptographic size of the keys depends on the underlying EKMF Web template.

For example, assuming you want to encrypt a volume `/dev/dasdb1` with the device mapper name `enc_disk` and generate an XTS key for this encryption, issue:

```
# zkey gen --name emkf-dasdb1 --xts --volumes /dev/dasdb1:enc_disk --description "XTS key for DASD B1"
```

Results

The key is saved in EKM Web with its properties. You can reuse the key for another system.

After the key is generated you can use the **kms list** command to see its properties, such as the two parts of the XTS key:

```
# zkey list
Key                                     : emkf-dasdb1
-----
Description                           : XTS key for DASD B1
Secure key size                       : 272 bytes
Clear key size                       : 512 bits
XTS type key                         : Yes
Key type                             : CCA-AESCIIPHER
Volumes                             : /dev/dasdb1:enc_disk
APQNs                                : 08.002f
                                     09.002f
Key file name                       : /etc/zkey/repository/emkf-dasdb1.skey
Sector size                         : (system default)
Volume type                         : LUKS2
Verification pattern                 : 709bc1e20e34f940362761141e094c65
                                     d15bc6cc177d88e7c704577df96d1484
KMS                                  : EKMWeb
KMS key label                       : ZKEY.XTS1.00002
                                     ZKEY.XTS2.00002
Created                             : 2021-03-17 17:31:14
Changed                             : (never)
Re-enciphered                       : (never)
```

Changing key properties

To change a property of a key, use the **zkey change** command

About this task

Properties that you change with the **zkey change** command are updated in EKM Web.

Some properties depend on the system, such the AP queues. These are not stored in EKM Web.

Other properties represent the same physical entity, but need different names on different systems, for example, the same physical volume can be mounted to two Linux instances under different names. The same key can be imported on another Linux instance, to another zkey repository, under a different name.

For details about the **zkey change** command, see the **zkey** command reference in *Pervasive Encryption for Data Volumes*, SC34-2782, or the man page.

Procedure

1. Optional: List the key properties.
For example, assuming the key generated before:

```
# zkey list
Key                               : emkf-dasdb1
-----
Description                       : XTS key for DASD B1
Secure key size                   : 272 bytes
Clear key size                    : 512 bits
XTS type key                      : Yes
Key type                         : CCA-AESGIPHER
Volumes                          : /dev/dasdb1:enc_disk
APQNs                            : 08.002f
                                : 09.002f
Key file name                     : /etc/zkey/repository/emkf-dasdb1.skey
Sector size                      : (system default)
Volume type                      : LUKS2
Verification pattern             : 709bc1e20e34f940362761141e094c65
                                : d15bc6cc177d88e7c704577df96d1484
KMS                              : EKMFWeb
KMS key label                    : ZKEY.XTS1.00002
                                : ZKEY.XTS2.00002
Created                          : 2021-03-17 17:31:14
Changed                          : (never)
Re-enciphered                   : (never)
```

You can change the description, the volume, the volume type, and the sector size. You cannot change the name with the **change** command. For how to rename a key, see [“Renaming a key”](#) on page 20.

2. Specify the **zkey change** and the name of the key, followed by the property you want to change. For example, to change the key description:

```
# zkey change -N emkf-dasdb1 -d "XTS key for some other DASD"
```

Results

The key now has the new description in the zkey repository as well as on EKMF Web:

```
# zkey list
Key                               : emkf-dasdb1
-----
Description                       : XTS key for some other DASD
...
Created                          : 2021-03-17 17:31:14
Changed                          : 2021-03-18 12:08:10
Re-enciphered                   : (never)
```

Listing keys

Use the **zkey kms list** command to display eligible secure keys that are managed by EKMF Web. These keys can, but must not be in the zkey repository.

About this task

You can filter the displayed list by:

- Key label, option **-B** or **--label**
- Key name, option **-N** or **--name**
- Associated volumes, option **-l** or **--volumes**
- Volume type, option **-t** or **--volume-type**
- State, option **-s** or **--states**

Most of these options are the same as for the **zkey list** command. For details about the filter options, see [Appendix A, “zkey kms - Managing secure keys with a KMS plug-in,”](#) on page 31, *Pervasive Encryption for Data Volumes*, SC34-2782, or the zkey man page.

Use the **--states** option to filter the list by the key state in EKMF Web. You can specify multiple states, separated by comma. By default, keys in ACTIVE state are displayed.

By default, only keys are displayed that this zkey client is allowed to use. This is controlled by export control options. When the export control options include the identity key of this zkey client as allowed exporting key, can the key be used by this zkey client. Specify the `--all` option to include keys that this zkey client is not allowed to use. The EKMF Web operator can change the export control options of a key to allow a certain zkey identity key to export the key.

Procedure

- To list all active keys the zkey instance can use, issue **zkey kms list**, for example:

```
# zkey kms list
Name                               : emkf-test
-----
Key label                          : ZKEY.XTS1.00002
                                   ZKEY.XTS2.00002
Description                        : A key generated in EKMF Web
Key size                          : 512 bits
XTS type key                       : Yes
Key type                          : CCA-AES-CIPHER
Volumes                           : /dev/dasdb1:enc_disk
Volume type                       : LUKS2
Sector size                       : (system default)
Addl. infos                       : State: ACTIVE
                                   Exporting keys: ZKEY.ID.EC.00001

Name                               : emkf-test2
-----
Key label                          : ZKEY.XTS1.00003
                                   ZKEY.XTS2.00003
Description                        : 2nd key generated in EKMF Web
Key size                          : 512 bits
XTS type key                       : Yes
Key type                          : CCA-AES-CIPHER
Volumes                           : /dev/dasda1:enc_disk
Volume type                       : LUKS2
Sector size                       : (system default)
Addl. infos                       : State: ACTIVE
                                   Exporting keys: ZKEY.ID.EC.00001
```

For more information on how to make keys available to a zkey instance, see [Chapter 5, “Sharing an EKMF Web key with another system,”](#) on page 23.

- To filter the list by name, specify a name, or part of a name, for example:

```
# zkey kms list -N "emkf*"
```

This command would result in the same list as in the example before.

Renaming a key

To change the name of a key, use the **zkey rename** command.

About this task

The zkey name is specific to the Linux instance with the zkey repository you are working with. The name is stored in EKMF Web for this instance.

For more information about the **zkey rename** command, see the **zkey** command reference in *Pervasive Encryption for Data Volumes*, SC34-2782, or the man page.

Procedure

Specify the name of the key and the new name. Issue a command of the following form:

```
# zkey rename -N <name> -w <new_name>
```

For example, to rename the key with the name `emkf-dasdb1` into `emkf-dasdb2`:

```
# zkey rename -N emkf-dasdb1 -w emkf-dasdb2
```

Results

The key name is changed in the zkey repository and in EKMF Web. Use the **zkey list** command to check the name:

```
# zkey list
....
Name                : emkf-dasdb2
....
```

Renaming a key changes only the name. EKMF Web still knows which key it is and any other changes feed through.

Refreshing keys

Use the **zkey kms refresh** command to refresh secure keys that are bound to EKMF Web.

About this task

Refreshing a key updates the secure key by reimporting it from EKMF Web.

You can filter the list of keys to be refreshed by:

- Key name, option -N or --name
- Key type, option -K or --key-type
- Associated volumes, option -l or --volumes
- Volume type, option -l or --volume-type

These options are the same as for other **zkey kms** commands. For details about the filter options, see [Appendix A, “zkey kms - Managing secure keys with a KMS plug-in,” on page 31, *Pervasive Encryption for Data Volumes*, SC34-2782](#), or the zkey man page.

Procedure

- To refresh a key, issue a command of the form:

```
# zkey kms refresh -N <name>
```

You can use wildcards to refresh several keys.

For example, to refresh all keys whose names start with "sec", issue:

```
zkey kms refresh -N "sec*"
```

- To refresh key properties, use the -P option

Refreshing updates the information on the zkey repository with the information from EKMF Web including the description, associated volumes, volume type, and sector size.

Refreshing the properties is useful when key properties have changed through the EKMF Web UI, or by zkey on another system (for shared keys).

Chapter 5. Sharing an EKM Web key with another system

A key that is generated in EKM Web can be reused on other Linux instances. This sharing is useful for backup, recovery, and cloning setups.

Before you begin

The Linux instance that you want to share the key with, or reuse the key on, must be connected to the same EKM Web instance as the original Linux instance from which the key was generated, see [“Connecting zkey with EKM Web” on page 11](#).

You need to know the label of the identity key of the second Linux instance. To find out what the label is, on that Linux instance issue the command **zkey kms info**, for example:

```
# zkey kms info
KMS-Plugin:          EKMWeb
...
Identity key:        ECC (secp521r1)
Registered key label: ZKEY.ID.EC.00025
```

The identity key label is the last entry in the list of information.

About this task

A key can be imported to other Linux instances. Use this capability when you set up backup or recovery systems. System-specific properties, such as the volume, might need to be changed on the imported key.

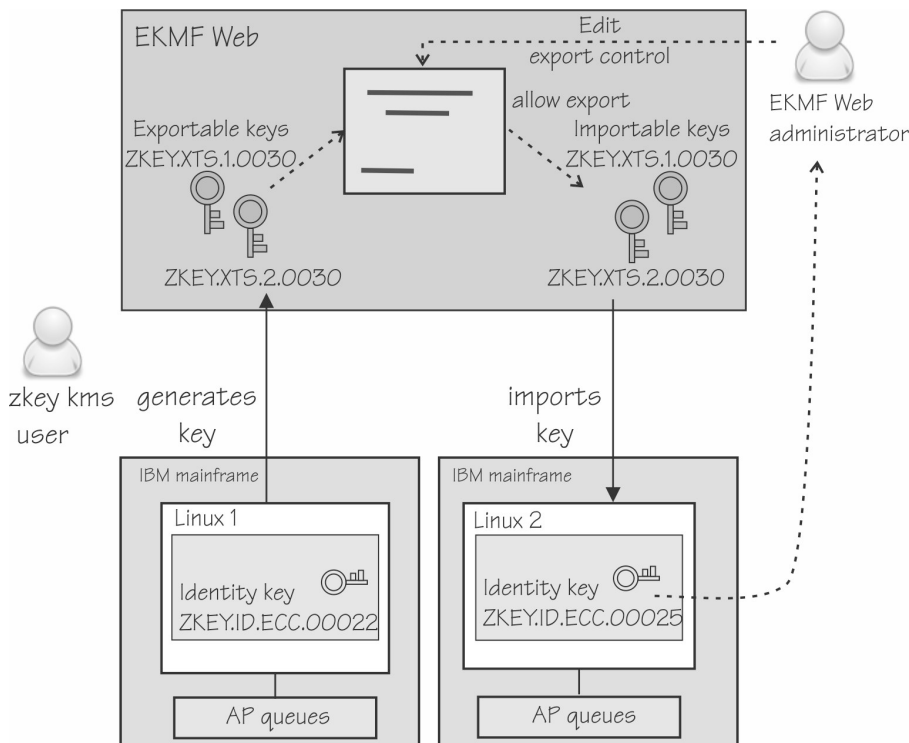
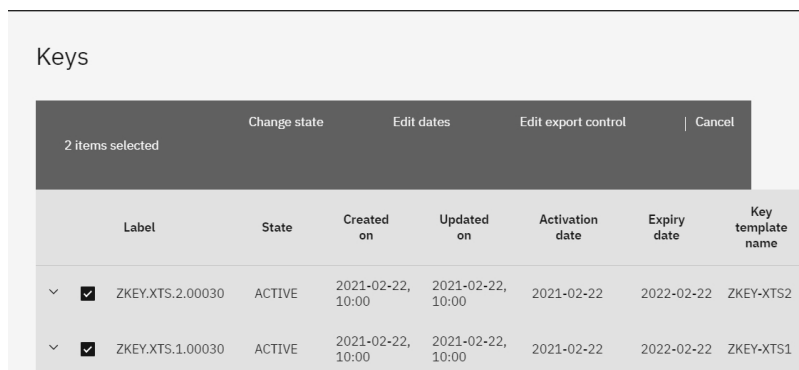


Figure 9. A key can be reused on another Linux instance

The imported key is protected by a transport key when it is sent to the second Linux instance, and then reencrypted with the master key of the AP queues the second instance uses.

Procedure

1. The EKMF Web administrator must allow the keys to be exported from EKMF Web.
 - a) On EKMF Web, go to **Key management** on the left navigation bar.
 - b) Go to **Keys**
 - c) Select the keys for which to edit export control.
For example, assume you want to export XTS keys ZKEY.XTS.1.0030 and ZKEY.XTS.2.0030 that were generated by the Linux instance with the identity key ZKEY.ID.ECC.0022 to the Linux instance with the identity key ZKEY.ID.ECC.0025. Select the keys as shown in [Figure 10 on page 24](#).



Keys

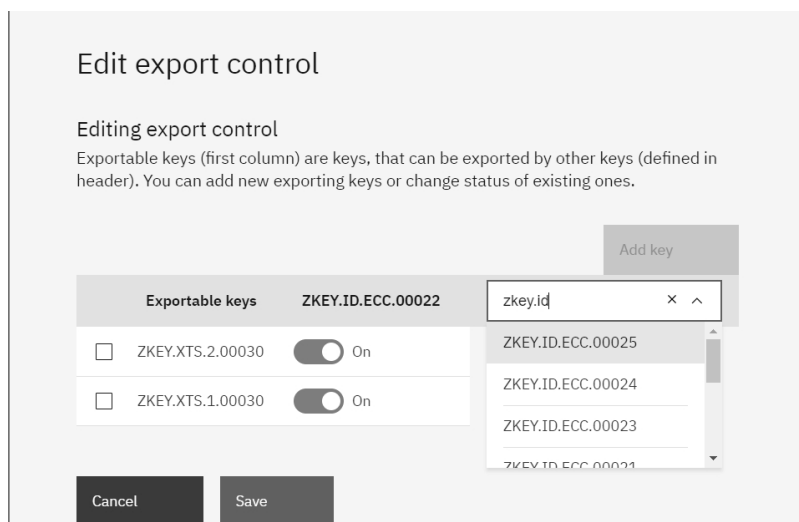
2 items selected

Change state Edit dates Edit export control Cancel

	Label	State	Created on	Updated on	Activation date	Expiry date	Key template name
<input checked="" type="checkbox"/>	ZKEY.XTS.2.00030	ACTIVE	2021-02-22, 10:00	2021-02-22, 10:00	2021-02-22	2022-02-22	ZKEY-XTS2
<input checked="" type="checkbox"/>	ZKEY.XTS.1.00030	ACTIVE	2021-02-22, 10:00	2021-02-22, 10:00	2021-02-22	2022-02-22	ZKEY-XTS1

Figure 10. Select the keys for which you want to change export control

- d) Select **Edit export control**.
- e) On the Export control page, click **Add key**.
- f) In the field that opens, type the name of identity key you want to work with
For example, assuming you want to export the keys to the Linux instance with the identity key ZKEY.ID.ECC.0025, as shown in [Figure 11 on page 24](#).



Edit export control

Editing export control

Exportable keys (first column) are keys, that can be exported by other keys (defined in header). You can add new exporting keys or change status of existing ones.

Add key

Exportable keys	ZKEY.ID.ECC.00022
<input type="checkbox"/> ZKEY.XTS.2.00030	<input type="checkbox"/> On
<input type="checkbox"/> ZKEY.XTS.1.00030	<input type="checkbox"/> On

Cancel Save

zkey.id

- ZKEY.ID.ECC.00025
- ZKEY.ID.ECC.00024
- ZKEY.ID.ECC.00023
- ZKEY.ID.ECC.00021

Figure 11. Add the identity key of the Linux instance on which you want to reuse the keys

- g) Click **Save**
- In the example shown in [Figure 12 on page 25](#), the Linux instance with the identity key ZKEY.ID.ECC.0025 can now import the XTS keys.

Edit export control

Editing export control

Exportable keys (first column) are keys, that can be exported by other keys (defined in header). You can add new exporting keys or change status of existing ones.

Add key

Exportable keys	ZKEY.ID.ECC.00022	ZKEY.ID.ECC.00025
<input type="checkbox"/> ZKEY.XTS.2.00030	<input checked="" type="checkbox"/> On	<input checked="" type="checkbox"/> On
<input type="checkbox"/> ZKEY.XTS.1.00030	<input checked="" type="checkbox"/> On	<input checked="" type="checkbox"/> On

Cancel
Save

Figure 12. The identity key of the Linux instance on which you want to reuse the keys is added

- Optional: On the second Linux instance, check that the keys you want to use are available for importing.

To check that the key you wanted to import is available, use the **zkey kms list** command.

For example, the XTS keys the EKMF Web administrator allowed export for now show up in the list:

```
# zkey kms list
Name                               : emkf-dasdb1
-----
Key label                          : ZKEY.XTS1.0030
                                   ZKEY.XTS2.0030
Description                        : XTS key for DASD B1
Key size                          : 512 bits
XTS type key                       : Yes
Key type                          : CCA-AESICIPHER
Volumes                           : /dev/dasdb1:enc_disk
Volume type                       : LUKS2
Sector size                       : (system default)
Addl. infos                       : State: ACTIVE
                                   Exporting keys: ZKEY.ID.EC.00022
                                                ZKEY.ID.EC.00025
```

The keys you want to import must be in the ACTIVE state.

- On the second Linux instance, use **zkey** to import the keys that you want to use. Issue the **zkey kms import** command and specify the key label.

For example, to import the keys with labels starting with ZKEY.XTS, issue a command as follows:

```
# zkey kms import -B "ZKEY.XTS*"
```

You can filter the list of keys to import, by specifying the **-N** (name), **-B** (label), **-l** (volume), and **-t** (type) options with the import command.

For example, to import the key named emkf-dasdb1 of type LUKS2:

```
# zkey kms import --name emkf-dasdb1 -t LUKS2
```

- Change system-specific properties.

The volume and the name are system-specific and might have to be changed. AP queues are bound to the key management system, and are set automatically.

If needed, change the volume with the **zkey change** command, for example:

```
# zkey change --name emkf-dasdb1 \  
-l /dev/mapper/disk2:enc-disk2
```

To change the key name, use the **zkey kms rename** command, see [“Renaming a key” on page 20](#).

Results

You can now use the XTS keys on the second Linux instance.

Chapter 6. Changing the CCA master key

Changing a master key requires reenciphering all secure keys that are enciphered with it. This includes all secure keys in the secure key repository, but also secure keys used by the EKMF Web plug-in, such as the identity key.

About this task

A new CCA master key is set on the cryptographic coprocessor that the AP queues of your Linux instance uses. How to change master keys is described in [Setting a master key on the cryptographic coprocessor](#). See also the `zkey` man page.

When a master key changed, you must reencipher all secure keys that are contained in the secure key repository that are associated with the AP queues for which you change the master key. For identity keys, you must use the **`zkey kms reencipher`** command. For keys used to encrypt volumes, you can also use the **`zkey kms refresh`** command.

Procedure

1. Load the new master key into the NEW register using the TKE.
2. Reencipher the identity key of the EKMF Web plug-in.
Reencipher with the `--staged` option. For example, to reencipher an identity key with the master key in the NEW register, issue:

```
# zkey kms reencipher --to-new --staged
```

3. Reencipher the keys in the secure key repository.
For example, to reencipher keys that use the AP queues 08.002f, and 09.002f, issue:

```
# zkey reencipher --apqns 08.002f,09.002f --to-new --staged
```

4. On the TKE, make the new master key the active key by moving it into the CURRENT register.
5. Complete the reenciphering of the identity key.

```
# zkey kms reencipher --complete
```

6. Complete the reenciphering of the keys in the secure key repository and the KMS keys.

```
# zkey reencipher --apqns 08.002f,09.002f --complete
```

Alternatively, use **`zkey kms refresh`**. The refresh command reimports the key from EKMF Web using the current master key. You can use the refresh command to reimport keys even if the master keys were already changed.

Note: The refresh command does not reencipher local keys.

7. You must also re-encipher any secure AES volume keys when the AES master key changes. Use the **`zkey-cryptsetup`** command to do this.
For details about the **`zkey-cryptsetup`** command, see the command reference in *Pervasive Encryption for Data Volumes*, SC34-2782.

To re-encipher the secure key of the encrypted volume `/dev/mapper/disk1` in staged mode and complete it later:

```
# zkey-cryptsetup reencipher /dev/mapper/disk1 --staged

Enter passphrase for '/dev/mapper/disk1': disk1pw
The secure volume key of device '/dev/mapper/disk1' is enciphered with the
CURRENT master key and is being re-enciphered with the NEW master key.
Staged re-enciphering is initiated for device '/dev/mapper/disk1'. After the NEW
master key has been set to become the CURRENT master key, run 'zkey-cryptsetup
reencipher' with option '--complete' to complete the re-enciphering process.

# zkey-cryptsetup reencipher /dev/mapper/disk1 --complete
```

Chapter 7. Recovering a secure key repository

Restore a corrupted repository or create a backup repository.

Before you begin

This scenario assumes:

- Your organization uses EKMF Web
- You have a user ID and password for EKMF Web

About this task

Assume you have an EKMF Web and pervasive encryption solution on Linux on Z, but your data center is flooded. Now your secure key repository is gone. But luckily, your encrypted volumes are safe, and you use EKMF Web, so your keys are safe.

You can recover a secure key repository by setting up a new repository on a new Linux instance, and reimporting the keys from EKMF Web.

Procedure

1. Install a new Linux instance with **zkey**.
2. On the new Linux instance, bind **zkey** to EKMF Web.
For details, see [“Connecting zkey with EKMF Web” on page 11](#).
3. On the new Linux instance, configure **zkey**.
For details, see [“Configuring a key management system plug-in” on page 11](#).
4. On EKMF Web, the administrator must allow the keys from your old secure key repository to be exported from EKMF Web into the new secure key repository.
5. On the new Linux instance, import the keys by using the **zkey kms import** command.
To import all eligible keys, issue the following command:

```
# zkey kms import
```

For details on how to refresh only certain keys, see [“zkey kms import” on page 35](#).

Results

Your secure key repository is populated with the keys from EKMF Web, and you can resume working.

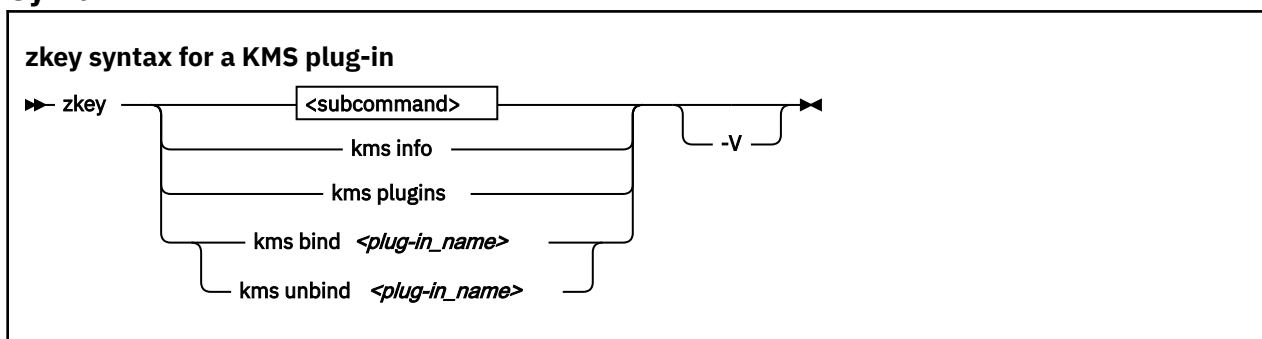
If the host name of the newly installed system remains the same, the key names and key properties of the imported keys also remain the same as on the original system.

If the host name is now different, some of the key properties might need adapting to the current system. Use the **zkey change** and **zkey rename** commands to adapt the key name and properties as needed.

Appendix A. zkey kms - Managing secure keys with a KMS plug-in

The **zkey** command has specific options for the EKMF Web plug-in. See the **zkey** command described in *Pervasive Encryption for Data Volumes*, SC34-2782 for the non-EKMF Web specific options. Also, see the `zkey man` page and the `zkey-ekmfweb man` page.

Syntax



<subcommand>

is described in the following sections:

- [“zkey kms configure” on page 31](#)
- [“zkey generate” on page 34](#)
- [“zkey kms import” on page 35](#)
- [“zkey kms list” on page 36](#)
- [“zkey kms reencipher” on page 37](#)
- [“zkey kms refresh” on page 37](#)
- [“zkey remove” on page 38](#)

kms info

displays information about the EKMF Web plug-in and its configuration.

kms plugins

displays the available KMS plug-ins.

kms bind <plug-in_name>

binds the zkey instance to the specified KMS plug-in.

kms unbind <plug-in_name>

releases the zkey instance from the specified KMS plug-in.

-h or --help

displays help information for the command. Specify `zkey <subcommand> -h` to get help for a subcommand. This option also displays KMS-specific option, if any.

-V or --verbose

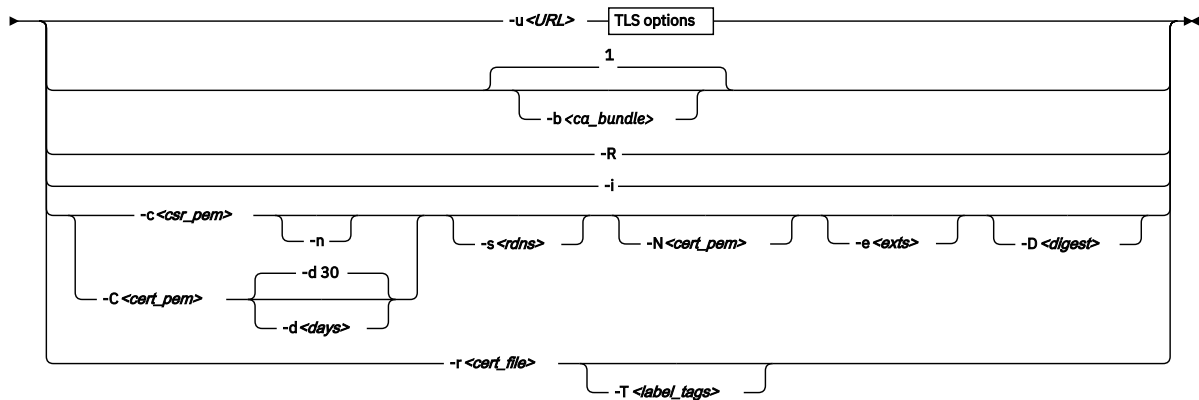
displays more details.

zkey kms configure

Use the **zkey kms configure** command to configure the EKMF Web plug-in.

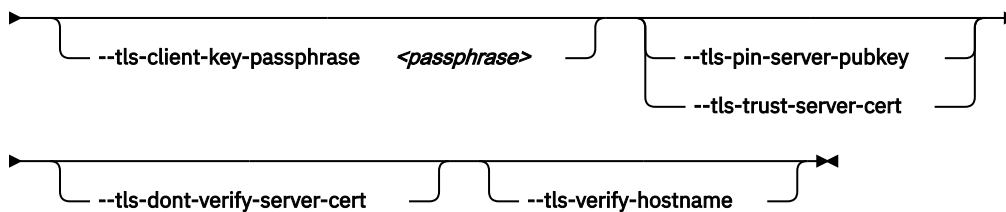
zkey kms configure syntax

►► zkey kms con →



TLS options

►► --tls-client-cert <cert_file> — --tls-client-key <priv_key> →



Notes:

¹ CA certificates

where:

-u or --ekmfweb-url <URL>

Specifies the URL of the EKM Web server. The URL starts with `https://`, and can contain a port number, which is separated by a colon. The default port number is 443 for HTTPS.

TLS options

--tls-client-cert <cert_file>

Specifies the X.509 certificate file that contains the client's transport layer security (TLS) certificate for use with TLS client authentication.

--tls-client-key <priv_key>

Specifies the X.509 certificate file that contains the client's private key for use with TLS client authentication.

--tls-client-key-passphrase <passphrase>

For passphrase-protected PEM files: Specifies the passphrase to unlock the PEM file that is specified with the `--tls-client-key` option.

--tls-pin-server-pubkey

For CA-signed EKM Web server certificates: Pins the public key of the EKM Web server. With a pinned key, it is verified that every connection uses the same EKM Web server certificate as the one used when the connection to the EKM Web server was configured.

--tls-trust-server-cert

Trusts the EKM Web server's certificate even if it is a self-signed certificate, or it was not verified due to other reasons. Use this option instead of the `--tls-pin-server-pubkey` option when you are using self-signed EKM Web server certificates.

--tls-dont-verify-server-cert

For self-signed EKMF Web server certificates used in test environments: Bypasses the EKMF Web server certificate verification by default. For CA-signed EKMF Web server certificates, the default is to verify them.

This option overrides `--tls-trust-server-cert`.

--tls-verify-hostname

Verifies that the EKMF Web server certificate's **Common Name** field or a **Subject Alternate Name** field matches the hostname that is used to connect to the EKMF Web server.

-b or --tls-ca-bundle <ca_bundle>

Specifies the CA bundle PEM file, or the directory that contains the CA certificates that are used to verify the EKMF Web server certificate during the TLS handshake. If the option specifies a directory path, then this directory must be prepared with OpenSSL's **c_rehash** utility.

The default is to use the system CA certificates.

-R or --refresh-settings

Refreshes the EKMF Web server settings. The settings are automatically refreshed when the connection to the EKMF Web server is configured or reconfigured. Use this option when the settings of the configured EKMF Web server changed.

-i or --gen-identity-key

Generates an identity key for the EKMF Web plug-in. An identity key is automatically generated for you when you configure the EKMF Web server connection. Use this option to generate a new identity key. If you regenerate the identity key you must also regenerate a registration certificate with the newly generated identity key, and reregister this zkey client with the EKMF Web server.

-c or --gen-csr <csr_pem>

Generates a certificate signing request (CSR) with the identity key and store it into the specified PEM file. You pass this CSR to a CA to have it issue a CA signed certificate for the EKMF Web plug-in. You need to register the certificate with EKMF Web before you can access EKMF Web.

-C or --gen-self-signed-cert <csr_pem>

Generates a self-signed certificate with the identity key and store it into the specified PEM file. You need to register the certificate with EKMF Web before you can access EKMF Web.

-s or --cert-subject <rdns>

Specifies the subject name for generating a CSR or self-signed certificate, in the form `<type>=<value>(;<type>=<value>)*[;]` with types recognized by OpenSSL.

-e or --cert-extensions <exts>

Specifies the certificate extensions for generating a CSR or self-signed certificate, in the form `<name>=[critical,]<value>(;<name>=[critical,]<value>)*[;]` with extension names and values recognized by OpenSSL.

-N or --renew-cert <cert_pem>

Specifies an existing PEM file that contains the certificate to be renewed. The certificate's subject name and extensions are used to generate the CSR or renewed self-signed certificate.

-n or --csr-new-header

Adds the word NEW to the PEM file header and footer lines on the CSR. Some software and some CAs need this marking.

-d or --cert-validity-days <days>

Specifies the number of days the self-signed certificate is valid. The default is 30 days.

-D or --cert-digest <digest>

Specifies the digest algorithm to use when you generate a certificate-signing request or self-signed certificate. If this specification is omitted, the OpenSSL default is used.

-r or --register <cert_file>

Registers the zkey client with EKMF Web by generating an identity key in EKMF Web by using a certificate from the specified file. Supported certificate files formats are `.pem`, `.crt`, `.cert`, `.cer`, and `.der` (that is, either base64 or DER encoded).

To register a self-signed certificate that you are about to generate by using the `--gen-self-signed-cert` option, specify the same certificate file name here, and the generated certificate is registered immediately.

-T or --label-tags <label-tags>

Specifies the label tags for generating the identity key in EKMF Web when registering the zkey client, in the form `<tag>=<value>(,<tag>=<value>)*[,]` with tags as defined by the key template. Use the **zkey kms info** command to display the key templates. For registration, the template for identity keys is used.

Examples

- To connect to the EKMF Web server on `my.ekmfweb.server`, issue:

```
# zkey kms configure -u https://my.ekmfweb.server
```

- To configure the connection to the EKMF Web server on `my.ekmfweb.server`, pin the server's public key from the server's TLS certificate, and verify that the hostname matches the server's **Common Name** in the certificate, issue:

```
zkey kms configure -u https://my.ekmfweb.server --tls-pin-server-pubkey --tls-verify-hostname
```

- To generate a certificate-signing request with the identity key and the specified subject name, and store it in a file named `csr.pem`, issue:

```
zkey kms configure -c csr.pem -s "CN=my.zkey.client;OU=Example;C=US"
```

- To generate a certificate-signing request with the identity key to renew the existing certificate in the file named `cert.pem` and store it in a file named `csr.pem`, issue:

```
zkey kms configure -c csr.pem -N cert.pem
```

- To generate a self-signed certificate with the identity key and the specified subject name and a validity of 50 days, and store it in a file named `cert.pem`, issue:

```
zkey kms configure -C cert.pem -s "CN=my.zkey.client;OU=Example;C=US" -d 50
```

- To generate a self-signed certificate with the identity key and the specified subject name and a certificate extension to limit the key usage, and store it in a file named `cert.pem`, issue:

```
zkey kms configure -C cert.pem -s "CN=my.zkey.client;OU=Example;C=US" -e "keyUsage=critical,digitalSignature,keyAgreement"
```

- To register the zkey client with EKMF Web by using the certificate in the file named `cert.pem`, issue:

```
zkey kms configure -r cert.pem
```

- To register the zkey client with EKMF Web by using the certificate in the `cert.pem` file, and the label tags `ENV=TEST` and `APP=LINUX` for the identity key, issue:

```
zkey kms configure -r cert.pem -T "ENV=TEST,APP=LINUX"
```

zkey generate

Use the `generate` command to generate a secure AES key in EKMF Web.

zkey generate syntax - options relevant to EKMF Web

► zkey generate — **-T <label-tags>** — **--local** ►

where:

-T, --label-tags <label-tags>

Specifies the label tags for generating a secure key in EKMF Web, in the form **<tag>=<value>(,<tag>=<value>)*[,]** with tags as defined by the key template. Use the **zkey kms info** command to find out which label tags the configured key template uses.

--local

Generates keys in the local secure key repository. These keys cannot be used in EKMF Web.

For a complete list of options for the **zkey generate** command, see the zkey man page.

zkey kms import

Use the **zkey kms import** command to import secure keys from EKMF Web into the secure key repository on your Linux instance. The default is to import all eligible keys.

zkey kms import syntax

► zkey kms im — **-B <key_label>** — **-N <key_name>** —

— **-l <vol_name>** — **:dm_name** — **-t <vol_type>** —

— **--no-volume-check** — **-q** ►

Where:

-B or --label <key_label>

Specifies the label of the secure key in the KMS. Use wildcards to select multiple secure keys. If you use wildcards, enclose the value in quotation marks.

-N or --name <key_name>

Specifies the key name of the secure key.

-l or --volumes <vol_name>

You can associate volumes with a key. Each volume association specifies the name of the block device, for example `/dev/mapper/disk1`, and the device mapper name separated by a colon.

Separate multiple volume associations with a comma, for example:

```
# zkey kms import -l /dev/mapper/disk1:enc-disk1,/dev/mapper/disk2:enc-disk2
```

-t or --volume-type <vol_type>

Specifies the volume type of the associated volumes used with dm-crypt. Possible values are PLAIN or LUKS2

--no-volume-check

Omits the volume check, and imports the keys even if the associated volumes do not exist.

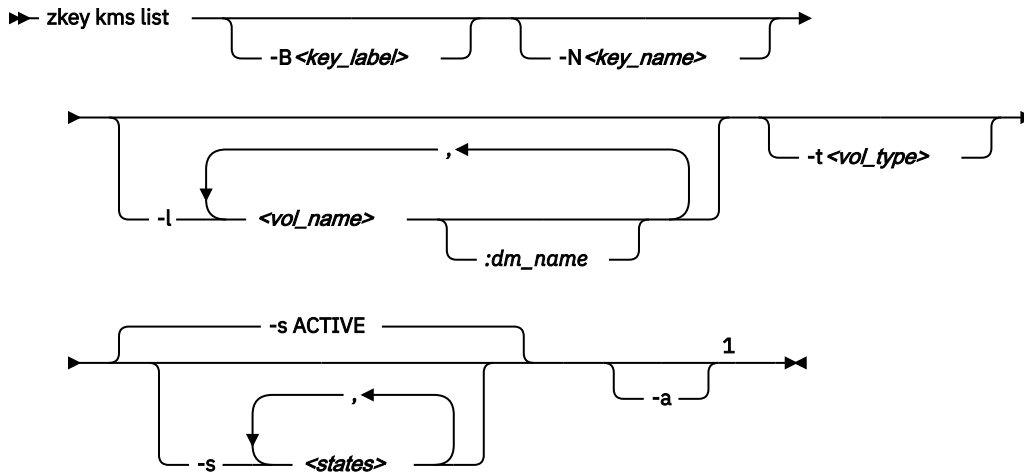
-q or --batch-mode

Suppresses prompts for names of existing keys. Keys with an existing name are skipped.

zkey kms list

Use the **zkey kms list** command to list EKMF Web encryption keys.

zkey kms list syntax



Notes:

¹ The default lists keys that the current secure key repository can use.

where:

-B or --label <key_label>

Specifies the label of the secure key in the KMS. Use wildcards to list multiple secure keys. If you use wildcards, enclose the value in quotation marks.

-N or --name <key_name>

Specifies the key name of the secure key.

-l or --volumes <vol_name>

You can filter the list by the volumes that are associated with a key. Each volume association specifies the name of the block device, for example `/dev/mapper/disk1`, and the device mapper name separated by a colon.

Separate multiple volume associations with a comma, for example:

```
# zkey kms list -l /dev/mapper/disk1:enc-disk1,/dev/mapper/disk2:enc-disk2
```

-t or --volume-type <vol_type>

Filters the list by volume type of the associated volumes used with dm-crypt. Possible values are PLAIN or LUKS2

-s or --states <states>

Filters the list by key states. Separate multiple states with a comma. Possible states are PREACTIVATION, ACTIVE, DEACTIVATED, COMPROMISED, DESTROYED, and DESTROYED-COMPROMISED.

The default is to list the ACTIVE keys.

-a or --all

Lists all keys that can be used for volume encryption.

By default, keys that can be exported to this secure key repository are listed.

Examples

- To list secure keys managed by EKMF Web, which this zkey client is allowed to use and are in state ACTIVE:

```
# zkey kms list
```

- To list secure keys managed by EKMF Web, which this zkey client is allowed to use, and are in ACTIVE or DEACTIVATED state:

```
# zkey kms list --states ACTIVE,DEACTIVATED
```

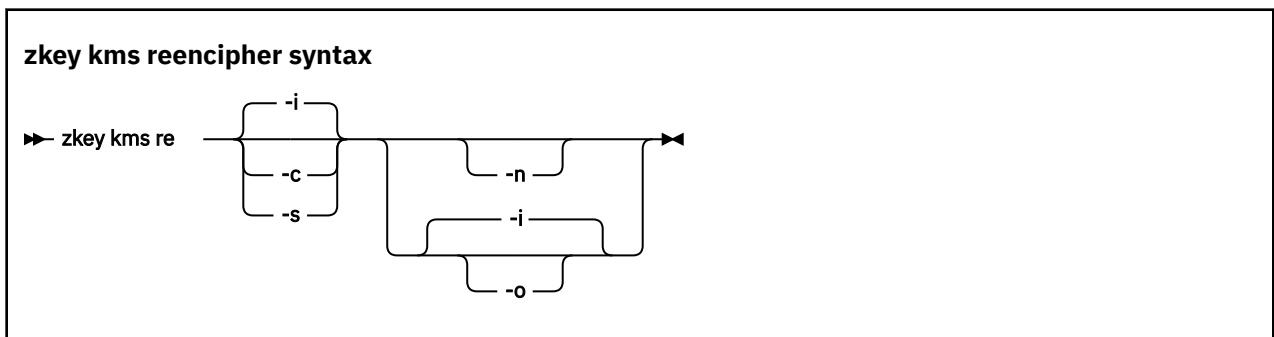
- To list secure keys managed by EKMF Web, regardless of whether zkey client is allowed to use it:

```
# zkey kms list --all
```

zkey kms reencrypt

Use the **zkey kms reencrypt** command to reencrypt a secure AES key in EKMF Web.

There can be plug-in specific options. Use **zkey kms reencrypt --help** to see which plug-in specific options a plug-in provide or requires.



where:

-n or --to-new

Reencrypts a secure key with the master key in the NEW register.

-o or --from-old

Reencrypts a secure key that is currently encrypted with the master key in the OLD register with the master key in the CURRENT register.

If both -n and -o are specified, a secure key that is currently encrypted with the master key in the OLD register is reencrypted with the master key in the NEW register.

-i or --in-place

Forces an in-place re-encrypting. This is the default for --from-old.

-s or --staged

Stores the key in a file, *<key-name>.renc*, in the secure key repository. The key in *<keyname>.skey* is still valid. Once a new master key has been set, you must rerun the reencrypt command with option --complete. This copies the file *<key-name>.renc* to *<key-name>.skey* and thus completes the staged re-encrypting. Re-encrypting from CURRENT to NEW is by default done in staged mode.

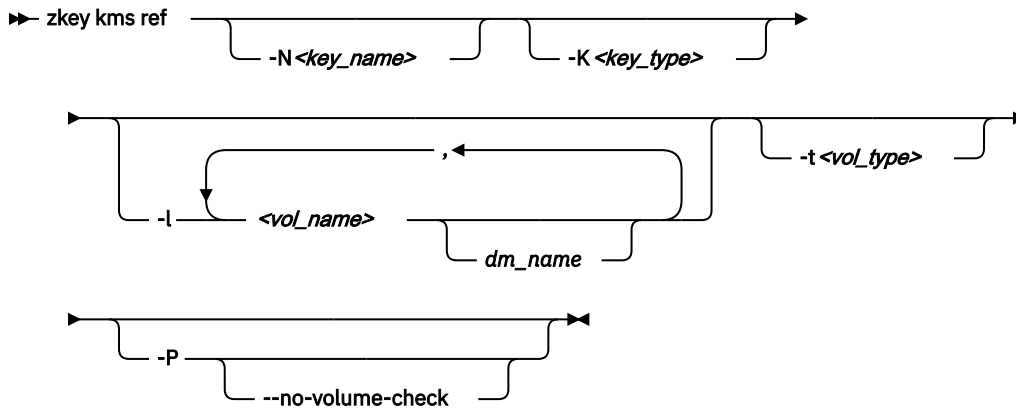
-c or --complete

Completes a staged re-encrypting.

zkey kms refresh

Use the **zkey kms refresh** command to reimport all, or a selection of EKMF Web encryption keys, or refresh key properties.

zkey kms refresh syntax



where:

-N or --name <key_name>

Specifies the key name of the secure key. Use wildcards to refresh multiple secure keys. If you use wildcards, enclose the value in quotation marks.

-K or --key-type <key_type>

Refreshes keys with the specified key type. Possible values are CCAESDATA, CCA-AESCIPHER, or EP11-AES.

-l or --volumes <vol_name>

You can filter the list of keys to refresh by the volumes that are associated with a key. Use wildcards to refresh keys for multiple volumes. If you use wildcards, enclose the value in quotation marks.

-t or --volume-type <vol_type>

Refreshes keys with the specified volume type. Possible values are PLAIN or LUKS2.

-P or --refresh-properties

Updates the associated information, such as the textual description, associated volumes, volume type, and sector size, with the information stored in the key management system.

--no-volume-check

Omits checking if the volumes that are associated with the secure keys to be refreshed are available, or are already associated with other secure keys in the repository. This option has an effect only if specified together with the `--refresh-properties` option.

Examples

- To refresh secure keys from EKMF Web whose name starts with "sec".

```
# zkey kms ref -N "sec*"
```

- To refresh the secure key with the name "seckey" from EKMF Web including its properties:

```
# zkey kms ref -N seckey -P
```

zkey remove

Use the remove command to remove EKMF Web encryption keys from the local repository, and optionally set a new key state for the key in EKMF Web.

zkey remove syntax - options relevant to EKMF Web

►► zkey rem — -N<key_name> — -s<state> — -F

where:

-N or --name <key_name>

Specifies the key name of the secure key.

-s, --state <state>

Specifies the new state for the key in EKMF Web after removing the secure key from the local secure key repository. Possible states are DEACTIVATED, COMPROMISED, DESTROYED, and DESTROYED-COMPROMISED.

The default is to remove the key from the local secure key repository, but leave the state in EKMF Web unchanged.

-F or --force

Forces the removal of the key

Example

- To remove the secure key named seckey from the repository and set the state of the key to DEACTIVATED in EKMF Web:

```
# zkey remove --name seckey --state DEACTIVATED
```


Accessibility

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Documentation accessibility

The Linux on Z and LinuxONE publications are in Adobe Portable Document Format (PDF) and should be compliant with accessibility standards. If you experience difficulties when you use the PDF file and want to request a Web-based format for this publication send an email to eservdoc@de.ibm.com or write to:

IBM Deutschland Research & Development GmbH
Information Development
Department 3282
Schoenaicher Strasse 220
71032 Boeblingen
Germany

In the request, be sure to include the publication number and title.

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.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility at

www.ibm.com/able

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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.

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

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Adobe is either a registered trademark or trademark of Adobe Systems Incorporated in the United States, and/or other countries.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Index

A

accessibility [41](#)
administration
 setup [5](#)

B

backup [23](#)
bibliography [v](#)
bind zkey to EKMF Web [11](#)

C

CCA master key change [27](#)
change CCA master key [27](#)
change key properties [18](#)
configuring
 plugin [11](#)
connect zkey with EKMF Web [11](#)

D

distribution [v](#)

E

EKMF Web
 change CCA master key [27](#)
 connect zkey to [11](#)
 generate key [17](#)
 reuse key [23](#)
EKMF Web setup [5](#)

F

filter listed keys [19](#)

G

generate key using EKMF Web [17](#)

H

hardware prerequisites [3](#)

K

key management system [2](#)
key properties
 refresh [21](#)
key repository
 backup [29](#)
 recovery [29](#)
 secure communication channel [2](#)

key templates [2](#)
key, change CCA master [27](#)
key, change properties [18](#)
key, display list [19](#)
key, refresh [21](#)
key, rename [20](#)
keys
 recover from EKMF Web [29](#)

L

label
 for keys, when generating [17](#)
 option for specifying [17](#)
list keys [19](#)

M

management system
 key [2](#)
master key, changing [27](#)

N

name of key, change [20](#)

O

options
 for refreshing keys [21](#)
 listing keys [19](#)

P

plugin
 configuring [11](#)
prerequisites
 hardware prerequisites [3](#)
 software prerequisites [3](#)
properties, change [18](#)

R

recovery
 zkey repository [29](#)
refresh
 key [21](#)
 key properties [21](#)
rename key [20](#)
reuse key [23](#)

S

secure key repository
 backup [29](#)

- secure key repository (*continued*)
 - recovery [29](#)
- setup
 - administrator task [5](#)
 - EKMF Web [5](#)
 - zkey [11](#)
 - zkey user task [11](#)
- sharing keys across Linux instances [23](#)
- software prerequisites [3](#)

T

- templates
 - different key types [2](#)
- terminology [v](#)

Z

- zkey
 - connect with EKMF Web [11](#)
 - options for EKMF Web [31](#)
 - recover repository [29](#)
 - use with EKMF Web [17](#)
- zkey setup [11](#)
- zkey user
 - setup [11](#)



SC34-7740-00

