IBM Enterprise Key Management Foundation - Web Edition - Installation and Configuration Guide

Tables of Contents

Notices

Notices References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

March 2021 Edition

This edition applies to EKMF Web Edition FMID HKMG200 and FMID HKMG201 Version 2.0.0, and to all subsequent releases and modifications until otherwise indicated in new editions.

About This Publication

IBM® EKMF Web for Pervasive Encryption (EKMF Web) is a web application that you use to manage keys. It uses the following components:

- A WebSphere Liberty server to provide these services:
    - REST services, and
    - A user interface for web browsers
- Agents that run on the z/OS LPARs to install keys into relevant keystores when using EKMF Web for Pervasive Encryption (PE)
- Db2 tables and views to store the keys in a key repository for management and potential recovery

Planning Considerations

The installation of EKMF Web includes the following installation steps:

Area	Details	Installation skills required

Area	Details	Installation skills required
z/OS network definitions	The connection between EKMF Web and the EKMF Agent program is based on the TCP/IP protocol. A raw IP socket interface is used. For installations already running the Trusted Key Entry (TKE), the EKMF host customization of the TCP/IP setup is similar to customization for the TKE host software.	z/OS TCP/IP
Db2 definitions and customization	The EKMF Web installation provides sample Db2 definitions. The sample must be customized for the installation.	z/OS Db2 administrator
ICSF software	Installation and configuration of ICSF.	z/OS system programmer
RACF definitions and customization	EKMF Web requires new RACF FACILITY/XFACILIT profiles to be defined. Also, access to the ICSF callable services, keys, and Db2 must be permitted. **Note:** This manual and other EKMF Web documents refer to the RACF Security Server, but other authorization systems can be used as well.	RACF administrator
Agent installation	The EKMF Web installation delivers load modules for the execution of the EKMF Agent. The load modules must be distributed and installed on each of the LPARs where an EKMF Agent must execute. On the LPARs serving the EKMF database, a Db2 package/bind and plan must be available for the EKMF Agent. Special attention should be given to the EKMF APF authorized modules. These modules should be reviewed and installed according to the procedures normally used for APF authorized programs.	z/OS application developer and z/OS system programmer

Program Requirements

The EKMF Web installation requires that other IBM program products are installed on the system, depending on the actual use and configuration of EKMF. The installation and customization of the other products are not within the scope of this document.

For all configurations the installation will require the following:

- z/OS V2.3 or higher
- Db2 V12 or higher
- Runtime libraries for Language Environment (LE)
- WebSphere Liberty 19.0.0.3 or later is installed
- Integrated Cryptographic Service Facility (ICSF) and IBM Crypto Express Card (CEX) assigned to LPAR where WebSphere Liberty is installed
    - Master keys configured for DES, AES, ECC (for Diffie-Hellman based link encryption) and RSA (for AES data key support)
    - Dynamic CKDS and PKDS Access must be enabled
    - CKDS configured to use variable-length key tokens
    - CEX6C and ICSF with FMID HCR77C1 or above for AES CIPHER key support
    - Mixed environments with both CEX5C and CEX6C require APAR OA58359.

SMP/E Installation

EKMF Web needs the following components to be installed via System Modification Program/Extended (SMP/E):

- The EKMF Web agent
- The EKMF Web web application, running in a WebSphere Liberty server

Agent SMP/E installation

You need to install the EKMF Agent via SMP/E using the FMID HKMG201.

Note: In case your FMID is different, for example HKMGAL1 or HKMGAL0, you will not be able to use this documentation. Contact the IBM Crypto Competency Center for the correct manuals to install the Agent for usage with EKMF Workstation via ccc@dk.ibm.com.

The following default target libraries are available after apply of the FMID:

Target data set	Description
KMG.SKMGCOPY	COBOL copy books of the EKMF Db2 tables
KMG.SKMGDBRM	Db2 bind members
KMG.SKMGMOD0	Load library for the EKMF server. It includes modules linked AC(1)
KMG.SKMGSAMP	Sample JCL, parameters
KMG.SKMGEXEC	Rexx utilities ASIS
KMG.SKMGSQL	Sample JCL to execute DDL and DDL/SQL statements for creating the EKMF database

Web application SMP/E installation

The EKMF Web package is installed via SMP/E and delivers files in the UNIX System Services (uss) as follows in the **`/usr/lpp/IBM/ekmfweb/V2R0`** folder:

- Applications in the **`apps`** directory:
  - **`ekmf-rest-api.war`**
  - **`ekmf-web-front-end.war`**
  - **`openid-login-form.war`**
- Db2 ddl files in **`resources/db_definitions[CCSID]`**
  - Files are in lexicographic order; they should be applied in the order they are listed
  - Files with numbers in the 900 range cover views; they drop existing views and create new versions
- JCCA distribution package for z/OS, extracted to **`liberty/resources/lib/jcca`**
- Liberty configuration files:
  - **`server.env`**, which contains environment variables that are used by the server
  - **`server.xml`**, which contains all required include files
- More Liberty configuration files in the **`includes`** directory:
  - **`server-db2-zos.xml`**, which contains database configuration
  - **`server-ekmf-rest-api.xml`**, which contains the EKMF Web back-end application definition
  - **`server-front-end.xml`**, which contains the EKMF Web front-end application definition
  - **`server-open-id-provider.xml`**, which contains the OpenID Connect provider and client configurations
  - **`server-user-registry-tso.xml`**, which is used to enable login with RACF (SAF) credentials

- Java Options: **jvm.options**, which is used to define various Java options
- EKMF Web key hierarchy template definitions: **EKMF_WEB_HIERARCHY.xml** file in **resources/key_hierarchy** contains template definitions for the EKMF Web Recovery Key and the two KEKs. This is only relevant if you have an EKMF Workstation

TCP/IP Port Reservation

Update the TCP/IP PROFILE information with the port selected for the EKMF Agent and Liberty server.

For example, add the following statement:

```
PORT
50050 TCP  EKMF    ; EKMF Agent
9080  TCP  EKMFSVR ; EKMF Web Liberty server http port
9443  TCP  EKMFSVR ; EKMF Web Liberty server https port
```

TCP indicates that the TCP transport layer will be used. The ports 50050, 9080 and 9443 must be replaced with the port number selected for the EKMF Agent and the Liberty server.
EKMF and **EKMFSVR** must be replaced by the names of the started tasks for the EKMF Agent and the Liberty server respectively.

Database setup for EKMF Web

The database for EKMF Web is done by executing the Db2 ddl files in **resources/db_definitions** in the following order:

1. **EKMF_Common.ddl**
2. **EKMF_Web_PE.ddl**
3. **EKMF_Pervasive_Encryption.ddl**

All Db2 objects referenced in the DDLs need to be in the same schema in Db2.

Note: In case you alread have an existing EKMF Agent installed, for example as part of the EKMF Workstation solution, you can skip **EKMF_Common.ddl**. Those Db2 objects will already have been created.

Next step: Give EKMF Web access to the database

Database access for EKMF Web

For specific Db2 authorization to the Db2 views used by EKMF Web application, you should run the following **GRANT** statements where **<id>** is the Liberty server user.

For EKMF Web base

```
GRANT SELECT ON TABLE EKMF_AUDIT_LOG_VIEW TO '<id>'
GRANT INSERT ON TABLE EKMF_TEMPLATE_KEYSTORE_REL TO '<id>'
GRANT SELECT ON TABLE EKMF_TEMPLATE_KEYSTORE_REL TO '<id>'
```

```
GRANT UPDATE ON TABLE EKMF_TEMPLATE_KEYSTORE_REL TO '<id>'
GRANT DELETE ON TABLE EKMF_TEMPLATE_KEYSTORE_REL TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_TEMPLATES_MANAGING_SYSTEM TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_INSTANCES TO '<id>'
GRANT SELECT ON TABLE EKMF_KEY_DISTRIBUTIONS TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_MATERIALS TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_TAGS TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEYS TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_TEMPLATES  TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_TEMPLATES_ORIGINS  TO '<id>'
GRANT INSERT ON TABLE KEY_STORES TO '<id>'
GRANT SELECT ON TABLE KEY_STORES TO '<id>'
GRANT UPDATE ON TABLE KEY_STORES TO '<id>'
GRANT DELETE ON TABLE KEY_STORES TO '<id>'
GRANT INSERT ON TABLE VTSAUDIT TO '<id>'
GRANT SELECT ON TABLE VTSAUDIT TO '<id>'
GRANT INSERT ON TABLE XTEMPLATESPURE TO '<id>'
GRANT SELECT ON TABLE XTEMPLATESPURE TO '<id>'
GRANT UPDATE ON TABLE XTEMPLATESPURE TO '<id>'
GRANT DELETE ON TABLE XTEMPLATESPURE TO '<id>'
GRANT INSERT ON TABLE XUKDS7 TO '<id>'
GRANT SELECT ON TABLE XUKDS7 TO '<id>'
GRANT UPDATE ON TABLE XUKDS7 TO '<id>'
GRANT DELETE ON TABLE XUKDS7 TO '<id>'
```

For the Pervasive Encryption feature

```
GRANT SELECT ON TABLE KMGPE_DATASET TO '<id>'
GRANT SELECT ON TABLE KMGPE_SCAN_FILTER TO '<id>'
GRANT SELECT ON TABLE KMGPE_SEC_DATASET TO '<id>'
GRANT SELECT ON TABLE KMGPE_SYSTEM_INFO TO '<id>'
GRANT INSERT ON TABLE KMGAGENT_KEYSTORES TO '<id>'
GRANT SELECT ON TABLE KMGAGENT_KEYSTORES TO '<id>'
GRANT UPDATE ON TABLE KMGAGENT_KEYSTORES TO '<id>'
GRANT DELETE ON TABLE KMGAGENT_KEYSTORES TO '<id>'
```

For the Cloud Keystores feature

```
GRANT INSERT ON TABLE AWSKMS_KEYSTORES TO '<id>'
GRANT SELECT ON TABLE AWSKMS_KEYSTORES TO '<id>'
GRANT UPDATE ON TABLE AWSKMS_KEYSTORES TO '<id>'
GRANT DELETE ON TABLE AWSKMS_KEYSTORES TO '<id>'
GRANT INSERT ON TABLE AZURE_KEYSTORES TO '<id>'
GRANT SELECT ON TABLE AZURE_KEYSTORES TO '<id>'
GRANT UPDATE ON TABLE AZURE_KEYSTORES TO '<id>'
GRANT DELETE ON TABLE AZURE_KEYSTORES TO '<id>'
GRANT INSERT ON TABLE IBMCLOUD_KEYSTORES TO '<id>'
GRANT SELECT ON TABLE IBMCLOUD_KEYSTORES TO '<id>'
GRANT UPDATE ON TABLE IBMCLOUD_KEYSTORES TO '<id>'
GRANT DELETE ON TABLE IBMCLOUD_KEYSTORES TO '<id>'
```

For the API feature

```
GRANT INSERT ON TABLE XCERTS TO '<id>'
GRANT SELECT ON TABLE XCERTS TO '<id>'
GRANT UPDATE ON TABLE XCERTS TO '<id>'
GRANT DELETE ON TABLE XCERTS TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_CERTIFICATES TO '<id>'
GRANT SELECT ON TABLE EKMF_WEB_KEY_MATERIAL_EXPORT_CONTROL_KEYS_ALLOWED TO '<id>'
```

Database Backup

It is recommended that the EKMF Db2 database is backed up at regular intervals. Because the tables in some way are only used as backup of the production keys and certificates, many years can pass before a data loss is observed. Consider the lifetime of your keys and certificates to determine the lifetime of the EKMF Db2 backups. To recover from a data centre disaster where all key stores and backups are destroyed, it is advised to have procedures in place to keep some backups at a remote site.

Consult your database administrator about the frequency and lifetime of the backups. Your database administrator should be able to define proper procedures for backup, restore and disaster recovery in line with your installation's requirements.

Security Setup

This chapter will walk you through the security definitions required for EKMF Web.

Agent Security Setup

EKMF APF Authorization

KMG.SKMGMOD0 contains modules linked with AC(1). Ensure that EKMF fetches these modules from an APF authorized library. If a STEPLIB concatenation is used, then ensure that all STEPLIB data sets are on the APF list.

Add the module KMGPRACF to the AUTHTSF table of **SYS1.PARMLIB(IKJTSOxx)**. After the IKJTSOxx member has been updated, changes can be applied on the fly with the 'TSO PARMLIB' command:

```
TSO PARMLIB UPDATE(xx)
```

System Resources

Check the STEPLIB and KMGPARM DD-names in the EKMF procedure. The EKMF Agent user ID needs READ access to these data sets.

User ID and started task setup

An EKMF system operates with different kinds of users seen from a RACF (SAF) perspective.

- The EKMF Agent started **<task-user>** ID, **'EKMF'** in the examples, part of the group **'EKMFSGRP'**
- For EKMF Web, the EKMF Web **<client-user>** ID - which is the user ID defined in the **&WEBCLIENT** KPMGPARM - is assumed to be **'EKMFCLT'** in the examples and part of the group **'EKMFUGRP'**

The RACF resources accessed through the EKMF Agent are:

- Existing RACF system resources for RACF, ICSF, etc

- Specific EKMF resources checked by the EKMF authorized KMGPRACF module

The user ID assigned to the EKMF Agent, must have an OMVS segment to use the TCP/IP services. The UID does not have to be 0 (superuser), but can be any valid number. As the user ID associated with the EKMF Agent does not need a password to logon, a protected user ID can be defined, which is a user ID with no password.

Below, see an example of a RACF command for adding a user ID for EKMF:

```
/* Define RACF access group for EKMF Agent(s) to use */
ADDGROUP EKMFSGRP SUPGROUP(SYS1) OWNER(SYS1) OMVS(GID(123456))
/* Define EKMF Agent user ID                          */
ADDUSER EKMF NOPASSWORD DFLTGRP(EKMFSGRP) NAME('EKMF AGENT')
```

The following example adds client user IDs to the group of clients. It is assumed that those IDs have already been created:

```
/* Define RACF access group for EKMF client(s) to use*/
ADDGROUP EKMFUGRP OWNER(SYS1) SUP(SYS1)
/* Connect EKMF client to EKMF user access group     */
CONNECT EKMFCLT GROUP(EKMFUGRP)
```

Assuming that you create a PROCLIB entry called EKMF, then you can assign the EKMF user ID when the Agent is started, using the RACF STARTED class entry below:

```
/* Create profile to assign user ID to the EKMF PROC */
RDEFINE STARTED EKMF.* STDATA(USER(EKMF))
SETROPTS RACLIST(STARTED) REFRESH
```

Required RACF definitions

The following section will walk you through the different RACF resources which need to be accessed. For a full list with a detailed description of each class, refer to the RACF Reference table.

The EKMF Agent **<task-user>** ID must have READ access to **KMG.EKMF.KMGPRACF** to run an agent. An EKMF **<client-user>** needs access to **KMG.EKMF.KMGPRACF.<task-user>** to connect to a specific EKMF Agent which executes using the **<task-user>** as user ID.

```
/* Allow access to EKMF Agent runtime and client access to agent */
RDEFINE FACILITY KMG.EKMF.KMGPRACF  UACC(NONE) OWNER(EKMFSGRP)
PERMIT KMG.EKMF.KMGPRACF CLASS(FACILITY) ID(EKMFSGRP) ACCESS(READ)

RDEFINE FACILITY KMG.EKMF.KMGPRACF.* UACC(NONE) OWNER(EKMFSGRP)
RDEFINE FACILITY KMG.EKMF.KMGPRACF.EKMF UACC(NONE) OWNER(EKMFSGRP)
PERMIT KMG.EKMF.KMGPRACF.EKMF CLASS(FACILITY) ID(EKMFUGRP) ACCESS(READ)

SETROPTS RACLIST(FACILITY) REFRESH
```

When an EKMF client user logs on to the EKMF Agent, a check for the KMGPRACF resource is done in the **APPL** class. If KMGPRACF is defined in the APPL class, then the EKMF client user ID needs READ access.

```
/* Create and permit APPL class profile               */
RDEFINE APPL KMGPRACF UACC(NONE) OWNER(EKMFSGRP)
PERMIT KMGPRACF CLASS(APPL) ID(EKMFUGRP) ACC(READ)
SETROPTS RACLIST(APPL) REFRESH
```

ICSF and ACPs

For a full description of the ICSF functions and ACPS, refer to the ICSF and ACP Reference.

Access to ICSF functions in CSFSERV class

To be able to call ICSF, the EKMF Agent **<task-user>** ID needs READ access to the following profiles in the CSFSERV class:

- CSFCRC
- CSFDEC
- CSFENC
- CSFIQF
- CSFKDMR
- CSFKDSL
- CSFKIM
- CSFKRD
- CSFMGN
- CSFMVR
- CSFPKRD
- CSFPRR2
- CSFDSG
- CSFDSV
- CSFEDH
- CSFKRC2
- CSFKRR2
- CSFKRW
- CSFKRW2
- CSFKYT
- CSFKYT2
- CSFOWH
- CSFPKG
- CSFPKI
- CSFPKRC
- CSFPKRR
- CSFPKX

It is recommended that you establish a generic profile to protect all the **CSFSERV** resources that are not in use. If this has not been done yet, you could use for example the following command

```
RDEFINE CSFSERV ** OWNER(EKMFSGRP) UACC(NONE)
```

After that, you need to grant access to each required ICSF resource. For example, to grant access to the **CSFDSG** service:

```
RDEFINE CSFSERV CSFDSG OWNER(EKMFSGRP) UACC(NONE)
PERMIT CSFDSG CLASS(CSFSERV) ACCESS(READ) ID(EKMFSGRP)
```

This has to be repeated for all required ICSF resources. At the end, issue a **REFRESH** command:

```
SETROPTS RACLIST(CSFSERV) REFRESH
```

ACPs that need to be enabled for the Agent

In addition, the following Access Control Points (ACPs) must be enabled in the crypto adapter:

- 0010
- 0011
- 0012

- 0021
- 0022
- 0100
- 0101
- 0103
- 0104
- 0106
- 0130
- 0131
- 0144
- 0235
- 0275
- 0327
- 0329
- 0360
- 0362
- 0367
- 000E
- 000F
- 001D
- 003A *1)
- 00F4
- 00FC
- 00FD
- 012A
- 012B
- 012E
- 012F
- 01FF
- 023D
- 02B4
- 02B9

Note 1*) - Only needed if your installation needs to install trusted blocks from the EKMF Workstation.

Access to CSFKEYS class profiles

In the CSFKEYS class the EKMF **<task-user>** needs access to the key labels it administers, like specified in the following example, which allows access to all Pervasive Encryption keys with the matching **<key-label>**:

```
RDEFINE CSFKEYS <key-label> UACC(NONE) ICSF(SYMCPACFWRAP(YES),SYMCPACFRET(YES))
PERMIT  <key-label> CLASS(CSFKEYS) ID(EKMFSGRP) ACCESS(CONTROL)
SETROPTS RACLIST(CSFKEYS) REFRESH
```

Depending on the configuration of the Agent, the Agent's **<task-user>** will need **READ** access to additional key labels in the **CSFKEYS** class. If ICSF granular key label access control is enabled, the access level required is **CONTROL** instead of **READ**.

If you are using Diffie-Hellman link encryption, then the Agent's **<task-user>** needs access to the corresponding key label named **&SYS-ECCSIGN-PREFIX.<task-user>**. **&SYS-ECCSIGN-PREFIX** is defined as KMGPARM.
For example: If **&SYS-ECCSIGN-PREFIX(EKMF.SYSTEM.ECCSIGN)** is specified and the EKMF Agent **<task-user>** is **'EKMF'**, then:
the **<key-label>** is **EKMF.SYSTEM.ECCSIGN.EKMF**.

If **&SYS-RSAKEK-PREFIX** is specified in the KMGPARM options of the Agent, then the Agent's **<task-user>** needs access to the corresponding **&SYS-RSAKEK-PREFIX.<key-label>** key label.
For example: If **&SYS-RSAKEK-PREFIX(EKMF.SYSTEM.RSAKEK)** is defined and the EKMF Agent**<task-user>** is **'EKMF'**, then:
the **<key-label>** is **&SYS-RSAKEK-PREFIX.<task-user>**.

If ICSF keystore policy checking is active and the **CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL** resource in **XFACILIT** class is defined, the **CSF-PKDS-DEFAULT** resource in **CSFKEYS** class must also be defined and the the Agent's **<task-user>** needs access.

Security setup for WebSphere Liberty

There are a couple of security definitions which are required for the EKMF Web Liberty server. You need to define started tasks for the Liberty server itself and the Angel server. In addition, you must authorize the WebShere Liberty server to access these services:

- The Liberty angel process services, which are required for z/OS authorized services
- The z/OS authorized services
- The SAF authentication services
- The Resource Recovery Services (RRS)

See Configuring security for z/OS Connect and Enabling z/OS authorized services on Liberty for z/OS for details.

Server STARTED Profiles

You must define a **STARTED** class profile for the Liberty server so that the server is started as a z/OS started task.

The example has the following assumptions:

- The **EKMFSVR** procedure starts the server
- **LIBSRV** is the user ID for the Liberty server

```
RDEFINE STARTED EKMFSVR.* OWNER('<owner>') UACC(NONE) STDATA(USER(LIBSRV) +
    GROUP(LIBSRVGP) PRIVILEGED(NO) TRUSTED(NO) TRACE(YES))

SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

The server user ID must also have **write** access to files in the **$WLP_USER_DIR** directory.

You must define a **STARTED** class profile for the Liberty Angel process so that the process is started as a z/OS started task.

The example has the following assumptions:

- The **BBGZANGL** procedure starts the server
- **LIBSRV** is the user ID for the Liberty server

```
   RDEFINE STARTED BBGZANGL.* OWNER('<owner>') UACC(NONE) STDATA(USER(LIBSRV) +
     GROUP(LIBSRVGP) PRIVILEGED(NO) TRUSTED(NO) TRACE(NO))

   SETROPTS RACLIST(STARTED) GENERIC(STARTED) REFRESH
```

Liberty Authorization

```
RDEFINE SERVER BBG.ANGEL OWNER('<owner>') UACC(NONE)
PERMIT BBG.ANGEL CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM OWNER('<owner>') UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.SAFCRED OWNER('<owner>') UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.SAFCRED CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSWLM UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.ZOSWLM CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.TXRRS UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.TXRRS CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSDUMP UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.ZOSDUMP CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.LOCALCOM UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.LOCALCOM CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.WOLA UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.WOLA CLASS(SERVER)ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.PRODMGR UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.PRODMGR CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSAFM.ZOSAIO UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSAFM.ZOSAIO CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

RDEFINE SERVER BBG.AUTHMOD.BBGZSCFM.WOLA UACC(NONE)
PERMIT BBG.AUTHMOD.BBGZSCFM.WOLA CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

SETROPTS RACLIST(SERVER) REFRESH
```

The user ID for the Liberty server must have **READ** access to the EKMF Web prefix.

```
RDEFINE SERVER BBG.SECPFX.EKMFWEB OWNER('<owner>') UACC(NONE)

PERMIT BBG.SECPFX.EKMFWEB CLASS(SERVER) ACCESS(READ) ID(LIBSRV)

SETROPTS RACLIST(SERVER) REFRESH
```

The unauthenticated user ID for the Liberty server (**WSGUEST** by default) requires **READ** access to the **EKMFWEB** application id in the **APPL** class:

```
ADDUSER WSGUEST RESTRICTED NOPASSWORD NOOIDCARD +
    NAME('WAS DEFAULT USER    ') +
    OWNER(WSCLGP) +
    OMVS( autouid +
    HOME('/local/WebSphere/home/WSCLGP') +
    PROGRAM('/bin/sh') +
    ) +
    DFLTGRP(WSCLGP)

CONNECT WSGUEST GROUP(WSCLGP) OWNER(WSCLGP) AUTH(USE) UACC(NONE)

RDEFINE APPL EKMFWEB UACC(NONE)

PERMIT EKMFWEB CLASS(APPL) ACCESS(READ) ID(WSGUEST)
```

```
SETROPTS REFRESH RACLIST(APPL)
```

In a similar way, you must grant **READ** access to the **EKMFWEB** application id in the **APPL** class for all users who need to use EKMF Web.

You should define a generic application profile in the **EJBROLE** class with **UACC(NONE)** and no users in the access list. This is to ensure that future upgrades to the application will not accidentally grant users access to new roles until the resources have specifically been defined in RACF. For authentication purposes, two additional profiles in the **EJBROLE** class need to be defined and all users should have **READ** access to these last two resources.

```
RDEFINE EJBROLE EKMFWEB.*.* OWNER('<owner>') UACC(NONE)
RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.authenticated OWNER('<owner>') UACC(NONE)
RDEFINE EJBROLE EKMFWEB.com.ibm.ws.security.oauth20.* OWNER('<owner>') UACC(NONE)

PERMIT EKMFWEB.ekmf-rest-api.authenticated CLASS(EJBROLE) ACCESS(READ) ID(*)
PERMIT EKMFWEB.com.ibm.ws.security.oauth20.* CLASS(EJBROLE) ACCESS(READ) ID(*)

SETROPTS REFRESH RACLIST(EJBROLE)
```

The Liberty server user ID must have **READ** access to list SAF key rings.

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING OWNER('<owner>') UACC(NONE)
    PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ACCESS(READ) +
    ID(LIBSRV)

SETROPTS RACLIST(FACILITY) REFRESH
```

Server Certificate

EKMF Web uses SSL/TLS for all communication.

Use the **RACDCERT GENCERT** command to create server certificates. The following example assumes that the WebSphere Liberty server user ID is **LIBSRV**. Change **www.example.com** to match the server to which you want to deploy EKMF Web.

```
SETROPTS CLASSACT(DIGTCERT)

RACDCERT CERTAUTH GENCERT +
    SUBJECTSDN(CN('EKMF Web CA') OU('Liberty') O('IBM')) +
    WITHLABEL('EkmfWebLibertyCA') +
    NOTAFTER(DATE(2028-12-31) TIME(23:59:59)) +
    RSA SIZE(2048)

RACDCERT ID(LIBSRV) GENCERT +
    SUBJECTSDN(CN('www.example.com') OU('Liberty') O('IBM')) +
    WITHLABEL('EkmfWebLibertyServer') +
    SIGNWITH(CERTAUTH LABEL('EkmfWebLibertyCA')) +
    NOTAFTER(DATE(2023-12-31) TIME(23:59:59)) +
    RSA SIZE(2048)

RACDCERT ID(LIBSRV) GENCERT +
    SUBJECTSDN(CN('EkmfWebLibertyOpenID') OU('Liberty') O('IBM')) +
    WITHLABEL('EkmfWebLibertyOpenID') +
    SIGNWITH(CERTAUTH LABEL('EkmfWebLibertyCA')) +
    NOTAFTER(DATE(2023-12-31) TIME(23:59:59)) +
    RSA SIZE(2048)

SETROPTS RACLIST(DIGTCERT) REFRESH
```

Add the server certificates to a SAF key ring.

```
SETROPTS CLASSACT(DIGTRING)

RACDCERT ID(LIBSRV) ADDRING(EkmfKeyRing)

RACDCERT ID(LIBSRV) CONNECT(LABEL('EkmfWebLibertyServer') +
    RING (EkmfKeyRing) DEFAULT USAGE(PERSONAL))

RACDCERT ID(LIBSRV) CONNECT(LABEL('EkmfWebLibertyOpenID') +
    RING (EkmfKeyRing) USAGE(PERSONAL))

RACDCERT ID(LIBSRV) CONNECT(CERTAUTH LABEL('EkmfWebLibertyCA') +
    RING(EkmfKeyRing) USAGE(CERTAUTH))

SETROPTS RACLIST(DIGTRING) REFRESH
```

Create a data set for the CA certificate and export it for installation into a client browser or application truststore. Alternatively use your browser on logon to EKMF Web to download the certificate via the security icon. Change **HLQ** into your high-level qualifier.

```
RACDCERT CERTAUTH EXPORT (LABEL('EkmfWebLibertyCA')) +
    DSN ('HLQ.LIBSRV.CERT') FORMAT(CERTDER)
```

Enable the Liberty user to use the key ring:

```
RDEFINE RDATALIB LIBSRV.EKMFKEYRING.LST OWNER('<owner>') UACC(NONE)

PERMIT LIBSRV.EKMFKEYRING.LST CLASS(RDATALIB) +
    ACCESS(READ) ID(LIBSRV)

SETROPTS RACLIST(RDATALIB) REFRESH
```

Additional certificates

For cloud keystore connections we need to add the CA certificates to RACF. Upload the certificate to a VB LRECL=84 dataset and add to RACF. In the command below we assume the dataset name is **'<HLQ>.CA.CERT'**.

To obtain the root certificates for cloud vendors:

- IBM Cloud
- Amazon Web Services
- Microsoft Azure

At the time of writing these instructions, the certificates for IBM and Microsoft were provided by DigiCert.

```
SETROPTS CLASSACT(DIGTCERT)

RACDCERT CERTAUTH ADD('<HLQ>.CA.CERT') +
WITHLABEL('DIGICERT-ROOT-CA') TRUST

SETROPTS RACLIST(DIGTCERT) REFRESH
```

ICSF Authorization

The z/OS user ID (**LIBSRV**) that is associated with the EKMF Web server must be authorized to ICSF services. For detailed information about ICSF authorization, see the following page: Cryptographic Support Downloads for IBM z/OS.

For a full description of the ICSF functions and ACPS, refer to the ICSF and ACP Reference.

Access to ICSF functions in CSFSERV class

The z/OS user ID (**LIBSRV**) needs access to the following profiles in the CSFSERV class:

- CSFKGN
- CSFKGN2
- CSFKRC
- CSFKRR
- CSFKTB2
- CSFKYTX
- CSFPKB
- CSFRNG
- CSFDSG
- CSFDSV
- CSFEDH
- CSFKRC2
- CSFKRR2
- CSFKRW
- CSFKRW2
- CSFKYT
- CSFKYT2
- CSFOWH
- CSFPKG
- CSFPKI
- CSFPKRC
- CSFPKRR
- CSFPKRW
- CSFPKX
- CSFRNGL
- CSFSAD
- CSFSAE
- CSFSYI
- CSFSYI2
- CSFSYX

It is recommended that you establish a generic profile to protect all the **CSFSERV** resources that are not in use. If this has not been done yet, you could use for example the following command

```
RDEFINE CSFSERV ** OWNER(<owner>) UACC(NONE)
```

After that, you need to grant access to each required ICSF resource. For example, to grant access to the **CSFDSG** service:

```
RDEFINE CSFSERV CSFDSG OWNER(<owner>) UACC(NONE)
PERMIT CSFDSG CLASS(CSFSERV) ACCESS(READ) ID(LIBSERV)
```

This has to be repeated for all required ICSF resources. At the end, issue a **REFRESH** command:

```
SETROPTS RACLIST(CSFSERV) REFRESH
```

ACPs that need to be enabled for the Liberty server

In addition, the following Access Control Points must be enabled in the crypto card:

- 0021

- 0022
- 0100
- 0101
- 0103
- 0104
- 0106
- 0130
- 0131
- 0144
- 0235
- 0327
- 0329
- 0360
- 0362
- 0367
- 001D
- 008C
- 008E
- 00D7
- 00EA
- 00EB
- 00EC
- 00FC
- 00FD
- 012A
- 012B
- 012E
- 012F
- 01FF
- 023D
- 02B9

CSFKEYS access

Depending on the naming standards of the key labels in use, the user ID for the Liberty server must have permission for **CONTROL** access to **CSFKEYS** class profiles. This level of access enables the Liberty Server user ID to administer the keys. The actual commands depend on existing profiles in the **CSFKEYS** class. The following example shows how to define a **CSFKEYS** profile. This example profile limits this user ID to administering key labels that begin with the letter 'T' only. (The actual users of the 'T' key labels most likely need only **READ** access.)

```
RDEFINE CSFKEYS T* OWNER('<owner>') UACC(NONE)
ICSF(SYMCPACFWRAP(YES),SYMCPACFRET(YES))
PERMIT T* CLASS(CSFKEYS) ACCESS(CONTROL) ID(LIBSRV)

SETROPTS RACLIST(CSFKEYS) REFRESH
```

If ICSF keystore policy checking is active and the **CSF.PKDS.TOKEN.CHECK.DEFAULT.LABEL** resource in **XFACILIT** class is defined, the **CSF-PKDS-DEFAULT** resource in **CSFKEYS** class must also be defined and the user ID for the Liberty server must have READ access.

Session Link Encryption

Link encryption is the mechanism that creates an encrypted communication session between the Agent and a client. The critical data to protect are user IDs and potentially logon passwords. Key material itself is always encrypted.

Diffie-Hellman Link Encryption

Diffie-Hellman (DH) link encryption is the preferred link encryption scheme. It provides the strongest link encryption algorithm and key size, and can be achieved from the very first connection between the Agent and the client. The keys used in the Diffie-Hellman link encryption are 521 bit size ECC keys, while the generated session encryption keys are AES 256 CIPHER keys. Trust is created between the client and the EKMF Agent by the SHA-256 value of the ECC signature public key tokens.

The DH link encryption requires the following to create public key tokens:

- ICSF must be configured with ECC and AES master keys
- For the EKMF Agent, the following KMGPARM parameters must be set:
  - **`&WS-AUTH`** must be set to **`'ON'`**
  - **`&SYS-ECCSIGN-PREFIX`** must be defined, creating an ECC signature key for the EKMF Agent with the SHA-256 value found in the EKMF Agent job log
- EKMF Web must be started and a EKMF Web Identity Key Label be defined

The EKMF Agent will accept the client's public key token by having access to the XFACILIT class resource **`KMG.WS.<64-character-hex-fingerprint>`**.

For example, if your EKMG Agent's **`<task-user>`** is **`'EKMF'`** and the SHA-256 value of the ECC signature public key token for the client is **`8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102`**, then define and permit this profile for the EKMF Agent:

```
RDEFINE XFACILIT -
KMG.WS.8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102
PERMIT -
KMG.WS.8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102 -
CLASS(XFACILIT) ACC(READ) ID(EKMF)

SETROPTS RACL(XFACILIT) REFRESH
```

An EKMF client user ID must be given permission to be used with an EKMF agent running under the specific task user ID. Therefore, every EKMF Workstation **`<client-user>`** must have access to the **`KMG.EKMF.KMGPRACF.<task-user>`** profile in the FACILITY class. While EKMF Web only has one client user ID, there may be multiple client user IDs for the EKMF Workstation.

For example, the following commands are used to permit client user **`'EKMFCLT'`** to be used with an EKMF agent running as user **`'EKMF'`**:

```
RDEFINE FACILITY KMG.EKMF.KMGPRACF.EKMF
PERMIT KMG.EKMF.KMGPRACF.EKMF CLASS(FACILITY) ACC(READ) ID(EKMFCLT)
```

In addition, **for EKMF Web only**, the Agent requires access to the **`KMG.WEBCLIENT.<client-user>`** and **`KMG.LG.<64-character-hex-fingerprint>`** profiles in the XFACILIT class. The **`<client-user>`** must match the value specified for the **`&WEBCLIENT`** parameter in KMGPARM.

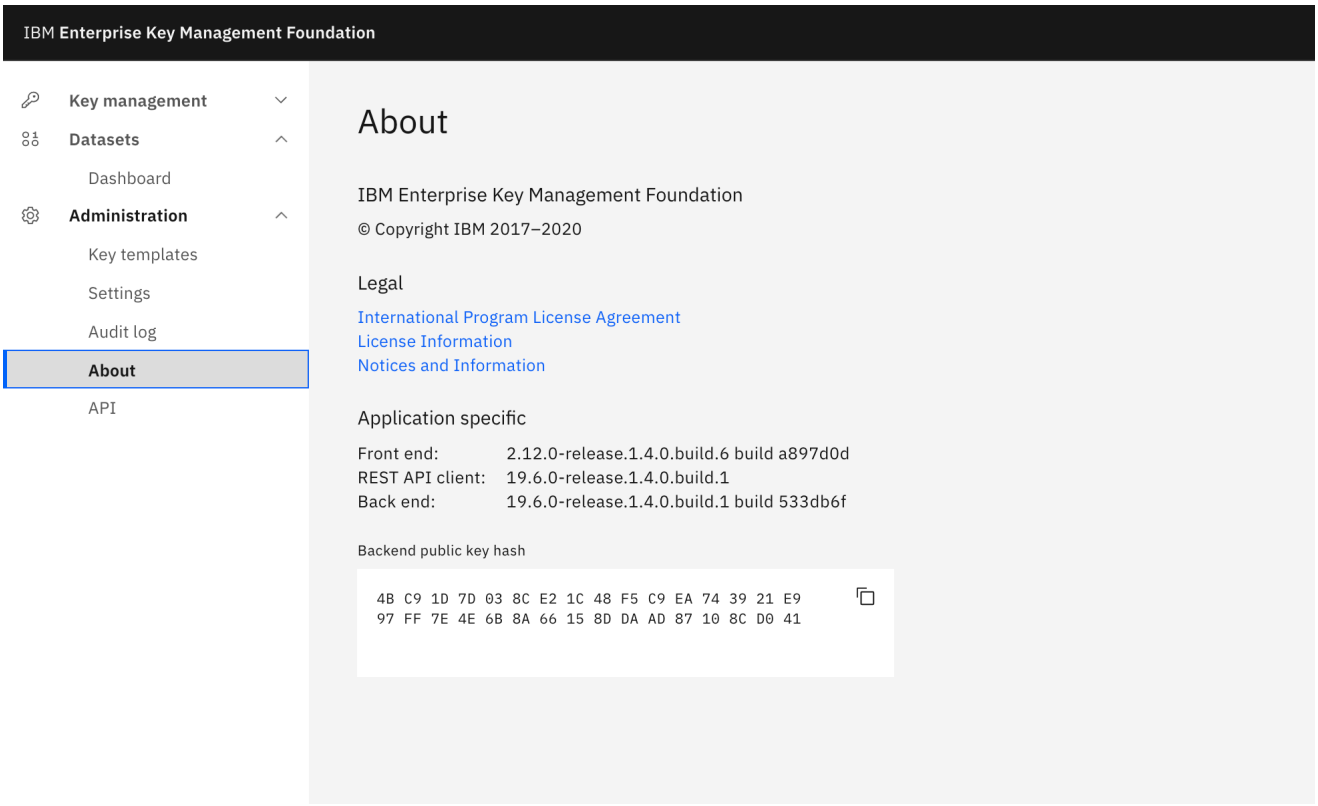
For example, using **`'EKMF'`** as the EKMF agent task user ID, **`&WEBCLIENT(EKMFCLT)`** in KMGPARM and **`8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102`** as fingerprint, specify:

```
RDEFINE XFACILIT KMG.WEBCLIENT.EKMFCLT
PERMIT KMG.WEBCLIENT.EKMFCLT CLASS(XFACILIT) ACC(READ) ID(EKMF)
```

```
RDEFINE XFACILIT -
KMG.LG.8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102
PERMIT -
KMG.LG.8A7A87509C40A5ED228E05DED51DD690EEFAC4C49B83E0A9A288B515D6745102 -
CLASS(XFACILIT) ACC(READ) ID(EKMF)
```

Where to find EKMF Web's identity key

EKMF Web calls this value the 'Backend public key hash' and displays this value on its *About* screen. The Agent uses this key to authenticate EKMF Web.



Preventing clients from disabling link encryption (optional)

There is a possibility to establish an unencrypted connection to an Agent. For the Agent to accept those connection, its **<task-user>** must have READ access to **KMG.EKMF.LNKCRYOFF**. It is recommended to define the profile with **UACC(NONE)** to avoid unecrypted connections.

```
RDEFINE FACILITY KMG.EKMF.LNKCRYOFF UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

The Agent can be configured to turn off the checking of the client's signature (when the KMGPARM **&WS-AUTH** is set to **'OFF'**). To allow this, the Agent **<task-user>** must have READ access to **KMG.WS.AUTHOFF**. It is recommended to define the profile with **UACC(NONE)** to ensure client signatures are always checked.

```
RDEFINE XFACILIT KMG.WS.AUTHOFF UACC(NONE)
SETROPTS RACLIST(XFACILIT) REFRESH
```

Roles

EKMF Web defines the following roles. If a subrole is required, you also need access to the related main role:

Role	Subrole	Description
`auditlog:read`		Allows the user to read the audit log.
`datasets:read`		List and view individual data sets.
`keys:generate`	`keys:non_existing:generate`	Allows the user to generate keys that are in the `PRE-ACTIVATION` state. To generate keys in `ACTIVE` state, the user must also have the `keys:pre_activation: activate` role.
`keys:read`		List and view individual keys from the database.
`keys:write`	`keys:pre_activation:activate`	Allows the user to activate keys in `PRE-ACTIVATION` state.
`keys:write`	`keys:pre_activation:destroy`	Allows the user to destroy keys in `PRE-ACTIVATION` state.
`keys:write`	`keys:pre_activation: mark_compromised`	Allows the user to mark keys in `PRE-ACTIVATION` state compromised.
`keys:write`	`keys:active:deactivate`	Allows the user to deactivate keys that are in the `ACTIVE` state.
`keys:write`	`keys:active:mark_compromised`	Allows the user to mark keys in `ACTIVE` state compromised.
`keys:write`	`keys:deactivated:reactivate`	Allows the user to reactivate keys that are in the `DEACTIVATED` state.
`keys:write`	`keys:deactivated:destroy`	Allows the user to destroy keys in `DEACTIVATED` state.
`keys:write`	`keys:deactivated:mark_compromised`	Allows the user to mark keys in `DEACTIVATED` state compromised.
`keys:write`	`keys:destroyed:mark_compromised`	Allows the user to mark keys in `DESTROYED` state compromised.
`keys:distribute`	`keys:active:install`	Install active keys in keystores.
`keys:distribute`	`keys:active:uninstall`	Uninstall active keys from keystores.
`keys:distribute`	`keys:deactivated:install`	Install deactivated keys in keystores.
`keys:distribute`	`keys:deactivated:uninstall`	Uninstall deactivated keys from keystores.
`keys:distribute`	`keys:compromised:install`	Install compromised keys in keystores.
`keys:distribute`	`keys:compromised:uninstall`	Uninstall compromised keys from keystores.
`keystores:read`		List and view individual keystore definitions.
`keystores:write`		Create and update keystore definitions.
`settings:write`		Allows the user to set settings.
`settings:write`	`keys:non_existing:import`	Allows the user to import the recovery key.
`templates:read`		List and view individual key templates.

Role	Subrole	Description
templates:write		Create, update, and delete key templates. This role also allows the setup of hierarchy keys.

Some key state actions automatically imply further actions in EKMF Web and you need to allow access to additional roles to allow the actions. This is described in the table below. The first example shows you need **keys:write** as main role, in accordance with the above table, in order to perform the **keys:pre_activation:activate** action. However since this automatically installs the key in the required keystores, you also need access to the **keys:active:install** role.

key state action	Roles required	Description
Activate	**keys:pre_activation:activate** **keys:write** **keys:active:install**	Used to change key state from **PRE-ACTIVATION** to **ACTIVE**, but the action also automatically distributes the key to the keystores listed in the related template.
Deactivate	**keys:active:deactivate** **keys:write** **keys:deactivated:uninstall**	Used to change key state from **ACTIVE** to **DEACTIVATED**, but the action also automatically uninstalls the key from the keystores it was distributed to.
Reactivate	**keys:deactivated:reactivate** **keys:write** **keys:active:install**	Used to change key state from **DEACTIVATED** to **ACTIVE**, but the action also automatically installs the key to the keystores listed in the related template.

In RACF, role specific profiles must be defined, for each of these, in the **EJBROLE** class and all user IDs must have **READ** access to the profiles corresponding to their required level of access.

```
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.auditlog:read OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.datasets:read OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:active:deactivate OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:active:install OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:active:mark_compromised
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:active:uninstall OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:compromised:destroy
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:compromised:install
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:compromised:uninstall
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:deactivated:destroy
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:deactivated:install
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:deactivated:mark_compromised
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:deactivated:reactivate
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:deactivated:uninstall
OWNER('<owner>') UACC(NONE)
```

```
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:destroyed:mark_compromised
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:distribute OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:generate OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:non_existing:generate
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:non_existing:import
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:pre_activation:activate
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:pre_activation:destroy
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:pre_activation:mark_compromised
OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:read OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keys:write OWNER('<owner>') UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keystores:read OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.keystores:write OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.settings:write OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.templates:read OWNER('<owner>')
UACC(NONE)
    RDEFINE EJBROLE EKMFWEB.ekmf-rest-api.templates:write OWNER('<owner>')
UACC(NONE)

    SETROPTS REFRESH RACLIST(EJBROLE)
```

You can choose to grant access to these roles on a user by user basis or you can grant access to a role specific RACF group and then connect users to these groups. Such groups could be defined like in the following example:

Group	Description	Roles included
EKMFWKA	Key Administrator who sets up the key hierarchy, and controls **keystores** and **templates**, as well as performs special key state actions.	auditlog:read keys:active:install keys:deactivated:reactivate keys:non_existing:import keys:read keys:write keystores:read keystores:write settings:write templates:read templates:write

Group	Description	Roles included
EKMFWKC1	Key Custodian who can generate keys in **PRE-ACTIVATION** state, but cannot **Activate** and **Distribute** them. **Destroy** and **Compromise** actions are also possible.	datasets:read keys:active:deactivate keys:active:mark_compromised keys:active:uninstall keys:compromised:destroy keys:compromised:uninstall keys:deactivated:destroy keys:deactivated:mark_compromised keys:deactivated:uninstall keys:destroyed:mark_compromised keys:distribute keys:generate keys:non_existing:generate keys:pre_activation:destroy keys:pre_activation:mark_compromised keys:read keys:write keystores:read templates:read

Group	Description	Roles included
EKMFWKC2	Key Custodian who cannot **Generate**, but can **Activate** keys bringing them from **PRE-ACTIVATION** to **ACTIVE** state (then distributes them). **Destroy** and **Compromise** actions are also possible.	datasets:read keys:active:deactivate keys:active:install keys:active:mark_compromised keys:active:uninstall keys:compromised:destroy keys:compromised:install keys:compromised:uninstall keys:deactivated:destroy keys:deactivated:install keys:deactivated:mark_compromised keys:deactivated:uninstall keys:destroyed:mark_compromised keys:distribute keys:pre_activation:activate keys:pre_activation:destroy keys:pre_activation:mark_compromised keys:read keys:write keystores:read templates:read
EKMFWAUD	Auditor who can view the audit log as well as keystores, templates, keys, and status of dataset encryption.	auditlog:read datasets:read keys:read keystores:read templates:read

The following RACF commands grant, for instance, the EKMFWKA group access to the listed roles. Similar commands should be performed for the other groups.

```
PERMIT EKMFWEB.ekmf-rest-api.auditlog:read CLASS(EJBROLE) ACCESS(READ) ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:activate:install CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:deactivated:reactivate CLASS(EJBROLE)
ACCESS(READ) ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:non_existing:import CLASS(EJBROLE) ACCESS(READ)
```

```
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:read CLASS(EJBROLE) ACCESS(READ) ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:write CLASS(EJBROLE) ACCESS(READ) ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keystores:read CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keystores:write CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.keys:settings:write CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.templates:read CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)
PERMIT EKMFWEB.ekmf-rest-api.templates:write CLASS(EJBROLE) ACCESS(READ)
ID(EKMFWKA)

SETROPTS REFRESH RACLIST(EJBROLE)
```

For a detailed description of using SAF for authenticating users, please see Configuring authorization for applications in Liberty.

User Registry

EKMF Web uses OpenID Connect as the authentication framework. You can use LDAP or SAF as your user registry. We provide a sample configuration that includes an application that provides the logon page. You may choose to use this, application, or you might choose to use an external identity provider. In this case, you must do these steps:

1. Remove the file **openid-login-form.war** from **$WLP_USER_DIR/servers/ekmfweb/apps**
2. Delete the following snippet from your **server.xml** file:

```
    <application
      id="openid-login-form"
      location="openid-login-form.war"
      name="openid-login-form"
      type="war"
      context-root="/ekmf-openid"/>
```

SAF User Registry

You can configure the built-in OpenID Connect server to work with the SAF user registry. You define the SAF user registry with the following snippet in the **server.xml** file:

```
<include location="includes/server-user-registry-tso.xml"/>
```

LDAP User Registry

You can configure the built-in OpenID Connect server to work with an existing LDAP server. The following example snippet configures an LDAP user registry:

```
<ldapRegistry
    baseDN="o=example.com"
    realm="LDAPRegistry"
    ldapType="LDAP Directory Server"
    port="636"
    host="ldap.example.com"
    sslEnabled="true"
```

```
    sslRef="ldapssl">
    <idsFilters
        groupFilter="(&amp;(cn=%v)(objectclass=groupOfUniqueNames))"
        groupMemberIdMap="groupOfUniqueNames:uniquemember"
        userFilter="(&amp;(emailAddress=%v)(objectclass=ibmPerson))"
        userIdMap="*:emailAddress"/>
</ldapRegistry>
```

External User Registry

You can use an existing OpenID Connect server. The following example snippet configures a client for an external OpenID Connect Provider:

```
<openidConnectClient id="OpenIdConnectClient"
    clientId="XXXXXXXXX" clientSecret="XXXXXXXXXX" (1)
    authorizationEndpointUrl="https://example.com/isam/authorize"
    tokenEndpointUrl="https://example.com/isam/token"
    issuerIdentifier="https://example.com/isam"
    scope="openid"
    signatureAlgorithm="RS256"
    introspectTokens="false"
    trustAliasName="OIDCJWT"
    trustStoreRef="OIDCTRUST"
    mapIdentityToRegistryUser="false"
    grantType="authorization_code" (2)
    userIdentityToCreateSubject="sub"
    createSession="false"
    headerName="Authorization"
    inboundPropagation="required"
    sslRef="EkmfWebSSLConfig" (3)
  <authFilter>
    <requestUrl urlPattern="oidc" matchType="contains"/>
  </authFilter>
</openidConnectClient>
```

1. Identity and secret key of the client. The secret is only used when **grantType** is **authorization_code**. The **clientID** must be the same as the value of EKMF_OAUTH_CLIENT_ID variable in the **$WLP_USER_DIR/servers/ekmfweb/server.env** file.
2. If **grantType** is set to **implicit** there is no need for providing a value to the **clientSecret** parameter.
3. A reference to the ssl configuration id defined in **server.xml** file.

The full extent of the setup of OpenID Connect is outside the scope of this document. See the Liberty website for more details.

Multi-Factor Authentication (MFA)

The provided login application supports Multi-Factor Authentication for z/OS when configured with the SAF user registry.

Depending on the configuration, the user must provide one of the following values on the logon page:

- MVS password and the OTP (or vice versa) separated by a configured separator.
- The OTP, instead of the password.
- A Cache Token Credential obtained from the MFA webservices page.

Agent Setup

The following sections will walk you through setting up the EKMF Agent.

Setting up the agent started task

KMG.SKMGSAMP provides the following sample procedure and and options files:

Member	Description
KMGPRCRY	A procedure for running the EKMF Agent
KMGOPCRY	Sample options for KMGPRCRY.

Use the **KMGPRCRY** procedure as a model to create a member in a system PROCLIB, for example **SYS1.PROCLIB**. For the purposes of this manual it is assumed that your PROCLIB member for EKMF is **'EKMF'**.

Edit the EKMF procedure, so that the **KMGPARM** DD-name allocates a PARMLIB data set member with the desired EKMF options. Use **KMGOPCRY** as a model.

This is a typical sample setup if you configure the Agent for the use with EKMF Web:

```
&CRYPTO-ENGINE(ICSF)     (1)
&IP-PORT(50050)          (2)
&DB2-USE(NO)             (3)
&WEBCLIENT(EKMFCLT)       (4)
&WSAUTH(ON)               (5)
&SYS-ECCSIGN-PREFIX(EKMF.SYSTEM.ECCSIGN)     (6)
&SYS-RSAKEK-PREFIX(EKMF.SYSTEM.RSAKEK)       (7)
&SYS-CCSID(IBM1047)     (8)
```

Those parameters specify

1. The Agent talks to ICSF, which is required for EKMF Web
2. The port that the Agent opens after start
3. The EKMF Web Liberty server talks to Db2 itself, it does not use the Agent for it - therefore no Db2 use for the Agent
4. The userid that will be used for Diffie-Hellman Link encryption, in this example **'EKMFCLT'**
5. Enables Diffie-Hellman Link encryption
6. Key label prefix for the ECC key, required for Diffie-Hellman Link encryption
7. RSA key label prefix if you plan to import AES Data Keys
8. The local codepage

If this is the first time starting up and running the Agent, it is recommended to specify the following parameters as well to enable the self-tests of the Agent:

```
&SELFTEST-CSF(ON)              (1)
&SELFTEST-CLIENT(EKMFCLT)      (2)
&DEBUG(ON)                     (3)
```

1. Checks whether the Agent **<task-user>** id has access to all required ICSF services, protected by the CSFSERV class
2. Checks whether the defined user id has access to logon to the EKMF Agent. It will also run two other tests which are not relevant for EKMF Web.

3. The Agent will write additional debug information

See EKMF KMGPARM options for a detailed description of the options.

Agent startup and setup verification

At this point the definitions should be in places to start the EKMF Agent.

Start the EKMF Agent with the START command:

```
START EKMF
```
or
```
S EKMF
```

The EKMF Agent includes self-tests that can be executed to verify correct setup of the EKMF Agent and EKMF repository database.

The following self-tests are run by the EKMF Agent and will check that the EKMF Agent:

- Is allowed to call the required ICSF services
- Is configured to allow a (specified) client RACF/SAF user ID to log on

This additional self-test JCL is submitted by the user ID for whom the database access test should be done.

Once all tests pass without errors, the setup and configuration of the EKMF Agent and EKMF repository database is complete.

Activating the EKMF Agent self-tests

The EKMF Agent splits its self-tests into different tests. Each test is activated using a parameter in the EKMF Agent's option data set. These parameters are described in detail in the KMGPARM options reference.

To instruct the EKMF Agent to run the self-tests, set the following parameters in the EKMF agent's option data set:

- `&SELFTEST-CSF` must be set to `ON`.
- `&SELFTEST-CLIENT` must be set to the user id that should be tested. Note that this is NOT the user id of the EKMF Agent started task.

Restarting the EKMF agent will run the self-tests and the result will be available in the EKMF Agent job log output. Each self-test can be rerun on demand, using modify commands, as described in EKMF Agent operation.

Once all tests have completed successfully, the `&SELFTEST-\*` parameters can be removed from the EKMF agent's options. This will stop the EKMF agent from running the self-tests every time it starts, and it will disable the corresponding modify commands that runs the self-tests on demand.

These self-tests should be run for each EKMF agent, to verify that each EKMF agent has been configured correctly, and that the given user can log on to all EKMF agents.

A typical `&SELFTEST-CLIENT` output will look like the following:

```
KMG839I <<< START TESTING CLIENT ACCESS FOR EKMFCLT
 - LOGON TO EKMF AGENT OK
```

```
 - AUDITOFF ALLOWED BY TASK AND USER
 - LOGON AS BROWSER OK
```

Note: Only the first test needs to pass. The other tests are not required for EKMF Web.

A typical **&SELFTEST-CSF** output will look like the following:

```
KMG829I <<< START TESTING ACCESS TO CSFSERV RESOURCES
KMG831I CSFENC OK
KMG831I CSFCRC OK
KMG831I CSFDEC OK
KMG831I CSFIQF OK
KMG831I CSFKDMR OK
KMG831I CSFKDMW OK
KMG831I CSFKDSL OK
KMG831I CSFKIM OK
KMG831I CSFKYT OK
KMG831I CSFKYT2 OK
KMG831I CSFMGN OK
KMG831I CSFMVR OK
KMG831I CSFKRR2 OK
KMG831I CSFKRC2 OK
KMG831I CSFKRW OK
KMG831I CSFKRW2 OK
KMG831I CSFKRD OK
KMG831I CSFOWH OK
KMG831I CSFPKG OK
KMG831I CSFPKI OK
KMG831I CSFPKX OK
KMG831I CSFSYI OK
KMG831I CSFSYI2 OK
KMG831I CSFDSV OK
KMG831I CSFPKRC OK
KMG831I CSFPKRR OK
KMG831I CSFPRR2 OK
KMG831I CSFPKRW OK
KMG831I CSFPKRD OK
KMG831I CSFSYX OK
KMG831I CSFDSG OK
KMG831I CSFEDH OK
KMG831I CSFRNGL OK
KMG831I CSFSAD OK
KMG831I CSFSAE OK
KMG830I >>> END TESTING ACCESS TO CSFSERV RESOURCES
```

EKMF Agent Operation

The following assumes that the EKMF PROCLIB procedure member is named 'EKMF'

Start the EKMF Agent with the START command:

```
START EKMF
```
or
```
S EKMF
```

Stop the EKMF Agent with the STOP or MODIFY command:

```
STOP EKMF
```
or
```
P EKMF
```
or
```
F EKMF,STOP
```

(Re-)Display the settings from startup:

```
F EKMF,DISPLAY,SETUP
```

Display the EKMF Agent module levels:

```
F EKMF,DISPLAY,LEVEL
```

Switch the setting of the KMGPARM variable &DEBUG:

```
F EKMF,DEBUG,ON
F EKMF,DEBUG,OFF
```

Switch the setting of the KMGPARM variable &TRACE-DATAIN:

```
F EKMF,TRACEIN,ON
F EKMF,TRACEIN,OFF
```

Switch the setting of the KMGPARM variable &TRACE-DATAOUT:

```
F EKMF,TRACEOUT,ON
F EKMF,TRACEOUT,OFF
```

Set the KMGPARM variables on for &DEBUG, &TRACE-DATAIN, &TRACE-DATAOUT:

```
F EKMF,TRACEALL,ON
```

Set the KMGPARM variables off for &TRACE-DATAIN, &TRACE-DATAOUT. &DEBUG setting not changed:

```
F EKMF,TRACEALL,OFF
```

Rerun one of the self tests defined by &SELFTEST-CSF, CLIENT or DB2:

```
F EKMF,SELFTEST,CSF
F EKMF,SELFTEST,CLIENT
```

WebSphere Liberty Configuration

Define the following environment variables now. You need them in later steps.

- `$WLP_USER_DIR` is the WebSphere Liberty user server directory (for example `etc/liberty`)
- `$EKMF_WEB_DIR` is the directory where SMP/E installed the application files, for example `/usr/lpp/ekmfweb/`

The first time that you run WebSphere Liberty as a started task, be aware that the Java executable that runs the server must be program-controlled. You must update the extended attributes of the Java executable to allow program control.

```
extattr +p $JAVA_HOME/bin/java
```

Copy the installation directory that contains the application files from the SMP/E install to the server directory you want to run from

```
cp -R $EKMF_WEB_DIR/ $WLP_USER_DIR/servers/ekmfweb
```

Liberty and Angel started task

WebSphere Liberty on z/OS provides two JCL procedure templates (PROCs) in the **/templates/zos/procs** sub directory of the Liberty installation directory. One template is for the server process, and the other is for the angel process:

- bbgzsrv for Liberty.
- bbgzangl for the angel process that is required for controlling access to system authorized services on the z/OS platform.

Administrators can copy the templates to a user location, and then customize them to start Liberty servers from the MVS console. For more details, refer to the WebSphere Liberty on z/OS documentation

Db2 Type 2 connection configuration

If you use the default, pre-configured type 2 connection to Db2, you must edit the **$/servers/ekmfweb/resources/properties/db2jcc.properties** file to match your Db2 environment. You may need to install the Db2 Connect package in your installation, and allow the started task user access to it. You do not need to provide the license for type 2 connections. Db2 requires the SYSSTAT package, which may need to be bound or rebound.

```
db2.jcc.ssid=PROVIDE
```

Optionally, depending on your environment, you may need to provide additional settings, e.g.:

```
db2.jcc.currentPackageSet=NULLID
db2.jcc.pkList=NULLID.*
```

Since EKMF Web uses a type 2 Db2 connection, the following lines, or similar depending on your system environment, need to be added to the z/OS started task for the Liberty server:

```
STEPLIB DD DSN=SYSDB2.DB2C10C.SDSNEXIT,DISP=SHR
        DD DSN=SYSDB2.DB2C10C.SDSNLOAD,DISP=SHR
        DD DSN=SYSDB2.DB2C10C.SDSNLOD2,DISP=SHR
```

Define data sources for the EKMF Web data sets database, the EKMF Web key database and EKMF Web audit database. Below is the default configuration in **$WLP_USER_DIR/servers/ekmfweb/server.xml**, which you can change if you want to. If so **jndiName** values also have to be updated in **$WLP_USER_DIR/servers/ekmfweb/apps/ekmf-rest-api/WEB-INF/web.xml**.

```
<dataSource jndiName="jdbc/EkmfDataSet" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            driverType="2"
            databaseName="${env.DATA_SET_DB_NAME}" (1)
            currentSchema="${env.DB_CURRENT_SCHEMA}"/> (2)
</dataSource>

<dataSource jndiName="jdbc/EkmfWeb" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            driverType="2"
            databaseName="${env.EKMF_API_DB_NAME}"
```

```
            currentSchema="${env.DB_CURRENT_SCHEMA}"/>
</dataSource>

<dataSource jndiName="jdbc/EkmfWebAudit" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            driverType="2"
            databaseName="${env.EKMF_API_DB_NAME}"
            currentSchema="${env.DB_CURRENT_SCHEMA}"/>
</dataSource>
```

1. The Db2 host name where the EKMF Web key repository databases are defined
2. The schema for accessing the EKMF Web key repository databases

You can manually substitute the values that you want in the xml file. Or you can define the values with environment variables.

Db2 Type 4 connection configuration

If you intend to connect to Db2 using a type 4 connection, you must ensure that Db2 Connect license file exists.

To connect to Db2 on z/OS, you must obtain a Db2 Connect license, **db2jcc_license_cisuz.jar**. You must place the license in the **$DB2_PRODUCT_PATH/jdbc/classes** directory with the remaining Db2 class files.

Db2 related configuration is referenced in the **$WLP_USER_DIR/servers/ekmfweb/includes/server-db2-zos.xml** file, as shown here:

```
<library id="db2.lib">
    <fileset dir="${env.DB2_PRODUCT_PATH}/jdbc/classes" includes="db2jcc.jar
db2jcc_license_cisuz.jar "/>
    <fileset dir="${env.DB2_PRODUCT_PATH}/jdbc/lib" includes="libdb2jcct2zos_64.so
"/>
</library>
```

Define data sources for the EKMF Web data sets database, the EKMF Web key database and EKMF Web audit database. For type 4 connections, you will need to manually edit the **$WLP_USER_DIR/servers/ekmfweb/server.xml** file. Please do not modify the **jndiName** values; if you do so, you will have to manually edit **$WLP_USER_DIR/servers/ekmfweb/apps/ekmf-rest-api.war/WEB-INF/web.xml** and repackage the WAR file. If you choose this route, please make sure to add the environment variables to **server.env** file.

```
<dataSource jndiName="jdbc/EkmfDataSet" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            databaseName="${env.EKMF_API_DB_NAME}" (1)
            serverName="${env.DB_SERVER_NAME}" (2)
            portNumber="${env.DB_SERVER_PORT}" (3)
            driverType="2"
            currentSchema="${env.DB_CURRENT_SCHEMA}"/> (4)
</dataSource>

<dataSource jndiName="jdbc/EkmfWeb" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            databaseName="${env.EKMF_API_DB_NAME}"
            serverName="${env.DB_SERVER_NAME}"
            portNumber="${env.DB_SERVER_PORT}"
            driverType="2"
```

```
            currentSchema="${env.DB_CURRENT_SCHEMA}"/>
</dataSource>

<dataSource jndiName="jdbc/EkmfWebAudit" jdbcDriverRef="DB2T2"
    type="javax.sql.ConnectionPoolDataSource">
        <properties.db2.jcc
            databaseName="${env.EKMF_API_DB_NAME}"
            serverName="${env.DB_SERVER_NAME}"
            portNumber="${env.DB_SERVER_PORT}"
            driverType="2"
            currentSchema="${env.DB_CURRENT_SCHEMA}"/>
</dataSource>
```

1. The Db2 host name where the EKMF Web key repository databases are defined
2. The server url of the EKMF Web key repository databases
3. The port number of the EKMF Web key repository databases
4. The schema for accessing the EKMF Web key repository databases

You can manually substitute the values that you want in the xml file. Or you can define the values with environment variables.

Configure TLS

Define the TLS configuration for the EKMF Web application. Below is the default configuration in **`$WLP_USER_DIR/servers/ekmfweb/server.xml`**, which you can change based on your own certificates defined in the section for Server Certificate.

```
<!-- SSL -->
<sslDefault sslRef="EkmfWebSSLConfig" />
<ssl id="EkmfWebSSLConfig"
     keyStoreRef="EkmfWebKeyStore"
     trustStoreRef="EkmfWebTrustStore"
     clientAuthentication="false"
     sslProtocol="TLSv1.2" (1)
     serverKeyAlias="EkmfWebLibertyServer" (2)
     securityLevel="HIGH" />
<keyStore id="EkmfWebKeyStore"
          location="safkeyring:///EkmfKeyRing" (3)
          type="JCERACFKS"
          password="password" (4)
          fileBased="false"
          readOnly="true" />
<keyStore id="EkmfWebTrustStore"
          location="safkeyring:///EkmfKeyRing"
          type="JCERACFKS"
          password="password"
          fileBased="false"
          readOnly="true" />
```

1. Setting TLS protocol to version 1.2
2. This is the personal certificate you defined in the Server Certificate section. If your naming is different than the default, please edit the file.
3. This is the name of the key ring you defined. If your naming is different than the default, please edit the file.
4. SAF does not use a password, but the Liberty keystore code requires it. This is just a dummy placeholder.

Environment Variables

Edit the **$WLP_USER_DIR/servers/ekmfweb/server.env** file (**this file is in EBCDIC!**) to match your environment. The contents of the **server.env** file that is provided are as follows:

```
WLP_SKIP_MAXPERMSIZE=true
JAVA_HOME=/usr/lpp/java/J8.0_64  # (1)

LOG_DIR=/var/log/ekmfweb # (2)

LIBPATH=/usr/lpp/db2c10/db2c10/jdbc/lib
DB2_PRODUCT_PATH=/usr/lpp/db2c10/db2c10

EKMF_API_DB_NAME=DB2F # (3)
DATA_SET_DB_NAME=DB2F # (4)
#DB_SERVER_NAME=ekmf-web.example.com # (5)
#DB_SERVER_PORT=446 # (6)
DB_CURRENT_SCHEMA=KRYTESTO # (7)
EKMF_CREATE_DB_VIEWS=true (8)

HTTP_PORT=-1 # (9)
HTTPS_PORT=443 # (10)
HOST=example.com # (11)

EKMF_OAUTH_CLIENT_ID=<add-a-value> (12)
#EKMF_OAUTH_CLIENT_SECRET=<add-a-value> (13)
#EKMF_START_VIEW=<add-a-value> (14)
#EKMF_DESCRIPTION=<add-a-value> (15)
#EKMF_LOCALE=<add-a-value> (16)
#EKMF_API_BASE_PATH=<add-a-value> (17)
#EKMF_OAUTH_STS_SERVER=<add-a-value> (18)
#EKMF_OAUTH_POST_LOGOUT=<add-a-value> (19)
#EKMF_OAUTH_REDIRECT=<add-a-value> (20)

#loginform_placeholders_userId="SAF User ID" (21)
#loginform_placeholders_password="SAF Password" (22)
#loginform_customMessage="This is a Production instance" (23)

TRACE_SPEC="*=info" # (24)

JVM_ARGS=-Dhttps.protocols=TLSv1.2 (25)
```

1. Replace **/usr/lpp/java/J8.0_64** with the path where Java is installed.
2. Location of the logs produced by the WebSphere Liberty server.
3. Name of the database name/location, which contains the EKMF Web key repository tables.
4. Name of the database name/location, which contains the EKMF Web dataset tables.
5. (Optional) Network address (IP address or host name) of the database server.
6. (Optional) The port of the database server.
7. The schema used for the databases.
8. (Optional) When **EKMF_CREATE_DB_VIEWS** is set to **true**, the EKMF Web application will automatically create any missing DB views. Omitting the variable defaults to **false**.
9. Port 80 is the default port for HTTP communications. Use an available port in your environment or **-1** if you wish to disable non-HTTPS access.
10. Port 443 is the default port for secure HTTPS communications; use an available port in your environment.
11. The name of the host where the server is available.
12. The public client id of the OAUTH Server.
13. (Optional) The private client secret of the client id. Only used if the OpenID Connect uses the **authorization_code** flow rather than the **implicit** flow. This is set in the grantType parameter of the **openidConnectClient** block in the **server-open-id-provider.xml** file.

14. (Optional) The default starting page for EKMF Web. The default start view is the **`Key templates list`** view. The following views can be configured as the default start view:
    - templates - Key templates list view (default)
    - keys - Key list view
    - keystores - Keystore list view
    - datasets - Data sets dashboard view
    - audit_log - Audit log viewer
15. (Optional) The value is shown to the left above the Hamburger Menu icon.
16. (Optional) The locale value, although for now setting this will not change anything. Will be automatially composed, if not set.
17. (Optional) Where the application is deployed. Adjust **`example.com:9443`** to match the host name/IP address of your environment.
18. (Optional) The OpenID Connect endpoint. If you're using the defaults, ensure that **`example.com:9443`** is the same as in the option above.
19. (Optional) Where the OpenID Connect server redirects after it logs off. If you use defaults, be sure to change it to be the same host name/IP address as in the value for **`api_base_path`**.
20. (Optional) Where the OpenID Connect redirects after successful logon. If you use defaults, be sure to change it to be the same host name/IP address as in the value for **`api_base_path`**, along with the context-root that is configured below.
21. (Optional) The value is shown in the login form to indicate which kind of user id to use. Default value if not specified is **`Your email address`**.
22. (Optional) The value is shown in the login form to indicate which kind of pw to use. Default value if not specified is **`Password`**.
23. (Optional) The value is shown in the login form as an optional customizable comment. It could state that the user is connecting to a Production system for instance.
24. (Optional) For debugging purposes. When you communicate with support, you'll be asked to change the value of this variable and restart the server.
25. Setting SSL protocol to version 1.2

Configure the front-end application

This is optional. Edit the **`$WLP_USER_DIR/servers/ekmfweb/includes/server-front-end.xml`** file to adjust the **`context-root`** property to specify where the EKMF Web application will be available to your users. For example, if your host name is **`www.example.com`** and the context root is **`ekmf-web`**, the application will be available at **`https://www.example.com/ekmf-web`**.

```
<server description="EKMF Web Front End">
    <application
    id="ekmf-web-front-end"
    location="ekmf-web-front-end"
    name="ekmf-web-front-end"
    type="war"
    context-root="/" />  (1)
</server>
```

1. Where the EKMF Web application will be available to your users. For example, if your host name is **`www.example.com`** and the context root is **`ekmf-web`**, the application will be available at **`https://www.example.com/ekmf-web`**.

Installing allowed features

Various functionality of the EKMF Web application is related to a set of feature codes you can buy. Currently the following features are supported:

#0332 Pervasive Encryption Key Management (dataset encryption) #0333 Cloud Key Management #0334 API Access

You can only enable these features if you deploy the corresponding jar file delivered with your installation package.

Java Options

Edit **`$WLP_USER_DIR/servers/ekmfweb/jvm.options`** (**this file is in EBCDIC!**) to match your environment. Confirm that you have already defined the **`java.util.prefs.userRoot`** Java property. The user ID that you use to run the server must have **`write`** access to that directory.

```
-Djava.util.prefs.userRoot=/usr/lpp/IBM/ekmfweb/V2R0
```

Configuring JCCA

JCCA that is distributed with EKMF Web needs to be properly configured for your environment. The files, located at **`$WLP_USER_DIR/servers/ekmfweb/resources/lib/jcca`**, are 31/64 bit z/OS Dynamic Load Libraries. These libraries are used to interface with the native hardware drivers for the IBM Crypto Express cards on IBM Z. You will need to create symbolic links to the library that matches your ICSF level. The file names used by EKMF Web are **`libjcca16.so`** for 31-bit and **`libjcca16_64.so`** for 64-bit. EKMF Web is delivered with symbolic links for these two file names pointing to the jcca resource files for HCR77D0. All delivered Dynamic Load Libraries have extended attributes set to allow program control in case your system has a different level of ICSF.

Example with a specific ICSF

This section shows the commands that you run to establish links to ICSF FMID HCR77D0, an Integrated Cryptographic Service Facility (ICSF) with that function modification identifier (FMID):

For 31-bit:

```
cd $WLP_USER_DIR/servers/ekmfweb/resources/lib/jcca
ln -s libjcca_zos_HCR77D0_31.so.1.6.6.0 libjcca16.so
```

For 64-bit:

```
cd $WLP_USER_DIR/servers/ekmfweb/resources/lib/jcca
ln -s libjcca_zos_HCR77D0_64.so.1.6.6.0 libjcca16_64.so
```

EKMF PE database

The EKMF PE database is a feature to monitor the actual usage of PE (Pervasive Encryption) of z/OS data sets.

The EKMF Agent package includes:

- The KMGPEUTL extract utility to get information from the catalog and volumes.
- JCL and Db2 load statements to load the EKMF PE database with KMGPEUTL output.

The EKMF PE database is designed to hold only the most current/newest extract as the Db2 LOAD replaces the data.

Note: This is an optional feature and has no influence on the EKMF Agent task or the EKMF key database.

KMGPEUTL -- Extract data set information from the catalog

The KMGPEUTIL is a compiled rexx which executes the catalog search IGGCSI00 API to gather information about data set names, types and encryption status.

SKMGSQL(KMGJCPEU) is a sample JCL to execute the utility.

The utility would normally only be needed to executed on one LPAR in a sysplex to be able to gather information for all applicable data sets in the sysplex.

JCL to run KMGPEUTL:

```
//KMGPEUTL EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPROC DD DISP=SHR,DSN=KMG.SKMGEXEC
//STEPLIB DD DISP=SHR,DSN=KMG.SKMGMOD0
//EXTRACT DD DISP=(,CATLG),DSN=KMG.PE.LIST01,
// UNIT=SYSALLDA,
// DCB=(RECFM=VB,LRECL=9999,BLKSIZE=23200),
// SPACE=(CYL,(5,5),RLSE)
//FILTER DD *
ALL **
//SYSTSIN DD *
%KMGPEUTL +
IGGTRACE(OFF) +
VTOCTEST(ON)
```

Parameter	Description
IGGTRACE(value)	For debug, default is OFF. When ON, then the output from the IGGCSI00 API is listed in SYSTSPRT
VTOCTEST(value)	Default is OFF. When ON then each data set is examined on the first volume listed in the catalog and its also determined if the data set is a PDS or not. For PDSE data sets it is determined if it's a record format U data set. Record format U data sets is assumed to contain program object and thus marked as not encryptable, while other record format PDSE data sets are marked encryptable (depending on the support included in z/OS 2.4 and also added in z/OS 2.2 / 2.3). When ONDEBUG then additional DEBUG messages are written in SYSTSPRT NOTE1: ON or ONDEBUG requires that KMGPETSV is authorized and defined as AUTHTSF in IKJTSOxx. NOTE2: If value is OFF, then the information about the PDSE record format is not available, and all PDSE data sets will appear as encryptable.
DD-name	**Description**
//EXTRACT	Contains the KMGPEUTL output. This is data used to load
//FILTER	This can limit the number of data set names to collect information from. Each line must contain 2 strings: First string is filter types used as CSIDTYPS for IGGCSI00. The CSIDTYPS supported is ABCGH and is the same is specifying ALL. Second string is the data set name filter CSIFILTK. See IBM product manual: "DFSMS Managing Catalogs" for details.

Example of using //FILTER:

```
//FILTER DD *
ALL PROD.LEDGER.**
ALL PROD.WEBBANK.**
```

The above will limit the search for data sets matching either **PROD.LEDGER.**** or **PROD.WEBBANK.**** and makes sense if these are the only qualifiers setup to use pervasive encryption.

The following security requirements need to be fulfilled:

DD-name	Description
KMGPETSV AUTHTSF	The KMGPEUTL calls KMGPEVCK when VTOCTEST(ON) or VTOCTEST(ONDEBUG) is specified. KMGPEVCK calls the authorized KMGPETSV which needs to be specified as AUTHTSF in IKJTSOxx.
CSFSERV CSFOWH	The KMGPEUTL calls CSNBOWH and needs access to CSFSERV resource CSFOWH
CATALOG ACCESS	The KMGPUTL needs READ access to the catalogs searched.

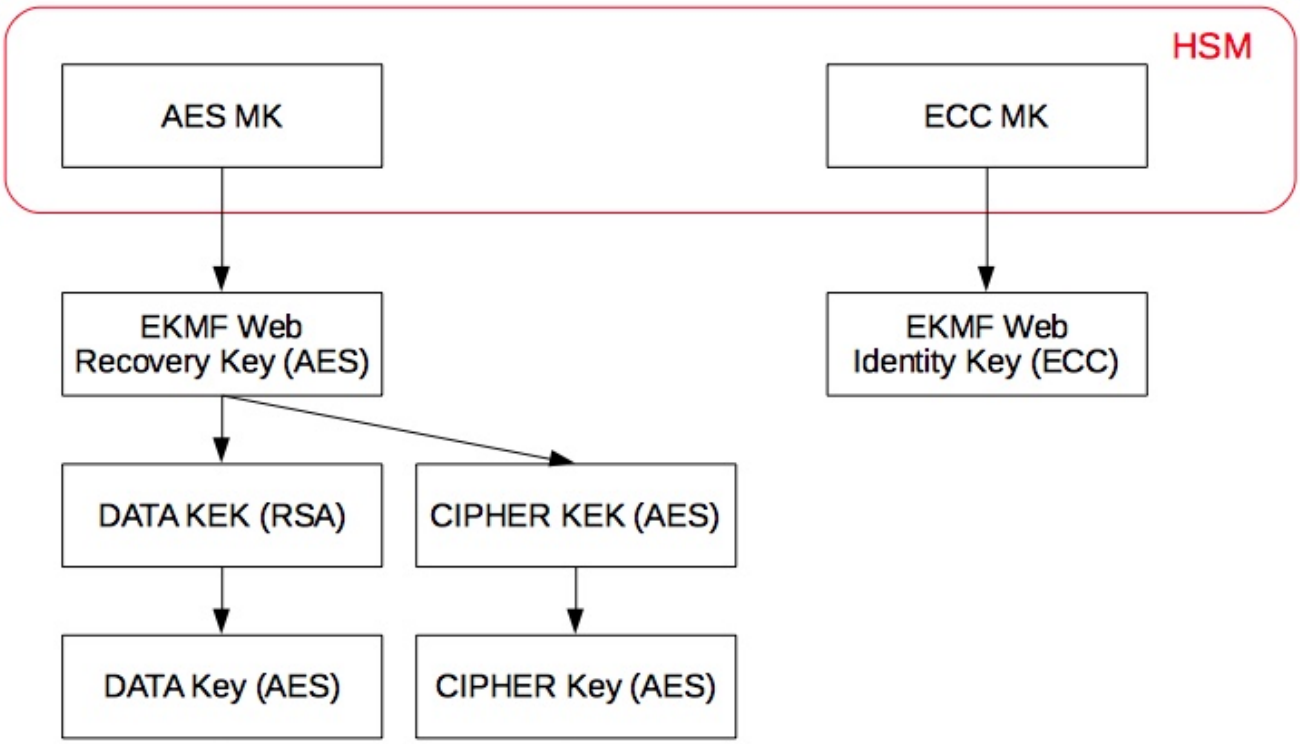
Load the EKMF PE tables

SKMGSQL(KMGJCPEL) contains the sample JCL to load the KMGPEUTL extract into the EKMF PE database. Edit the JCL and the input LOAD statements in SKMGSQL(KMGIPEL).

The DB2LOAD step normally ends with condition code 04 due to table spaces being IN COPY PENDING. This is repaired in the RUNSTATS step.

EKMF Web key hierarchy setup

EKMF Web requires the setup of a key hierarchy as shown in the following picture:



Note: All keys are always handled within the HSM. The master keys reside within the HSM itself, while all other keys are either in the EKMF Web key repository, local keystore, or both.

Disaster recovery key setup

EKMF Web requires the secure creation of the EKMF Web Disaster Recovery Key (DRK). The DRK must be created in such a way that it is recoverable. The recommended approach is to store the DRK in parts, such that it can be reconstructed from these key parts.

- Setting up the hierarchy keys by using EKMF Workstation
- Setting up the hierarchy keys by using Trusted Key Entry (Trusted Key Entry)
- If you are only setting up EKMF Web in a test, demonstration or Proof of Concept environment, you can use the DRK tool which will create a fixed recovery key. This option is only for demonstration or testing purposes and must not be used for production.

Note: If TKE isn't available, key parts and their corresponding key check values can be created and calculated using ICSF and then recorded on paper. Note that key part creation for the EKMF Web DRK using ICSF **isn't considered secure**. Keeping track of the paper copies of the key parts and storing them securely will be critical to prevent data loss in case the CKDS is corrupted. Each key part should be stored separately, such that no single individual has access to all parts. Entry of these key parts to build the DRK will require the use of the ICSF API.

Setting up hierarchy keys by using EKMF Workstation

If you have EKMF Workstation, you should use it to create the hierarchy keys. You can use the provided key templates for EKMF Web as the basis for your setup. Import the key templates in EKMF Web and generate the keys as you normally do. EKMF Web automatically picks up the key labels from the repository when the keys are activated. It searches for the newest active keys created by the following key templates:

- EKMF Web Recovery Key: `AES-W010`
- KEK for AES DATA Keys: `RSA-W050`
- KEK for AES CIPHER Keys: `AES-W011`

For EKMF Web users, these templates can be imported into the workstation using the `EKMF_WEB_HIERARCHY.xml` file included in the EKMF Web release package.

Setting up hierarchy keys by using Trusted Key Entry (Trusted Key Entry)

When using TKE (recommended, especially for production systems), these key parts can be securely generated and stored on smart cards. Subsequently, the key parts can be loaded from the smart cards and can installed in the CKDS keystore using ICSF. Restoring the DRK follows the same key loading process using the existing key parts on smart cards.

The key must be an `IMPORTER` created with the following keywords:

```
IMPORT,GEN-IMEX,GEN-PUB,TRANSLAT,WR-AES,WR-RSA,WR-KEK
```

We also recommend limiting the key's export options:

```
NOEX-SYM,NOEX-RAW,NOEXUASY,NOEXAASY,NOEX-DES,NOEX-AES,NOEX-RSA
```

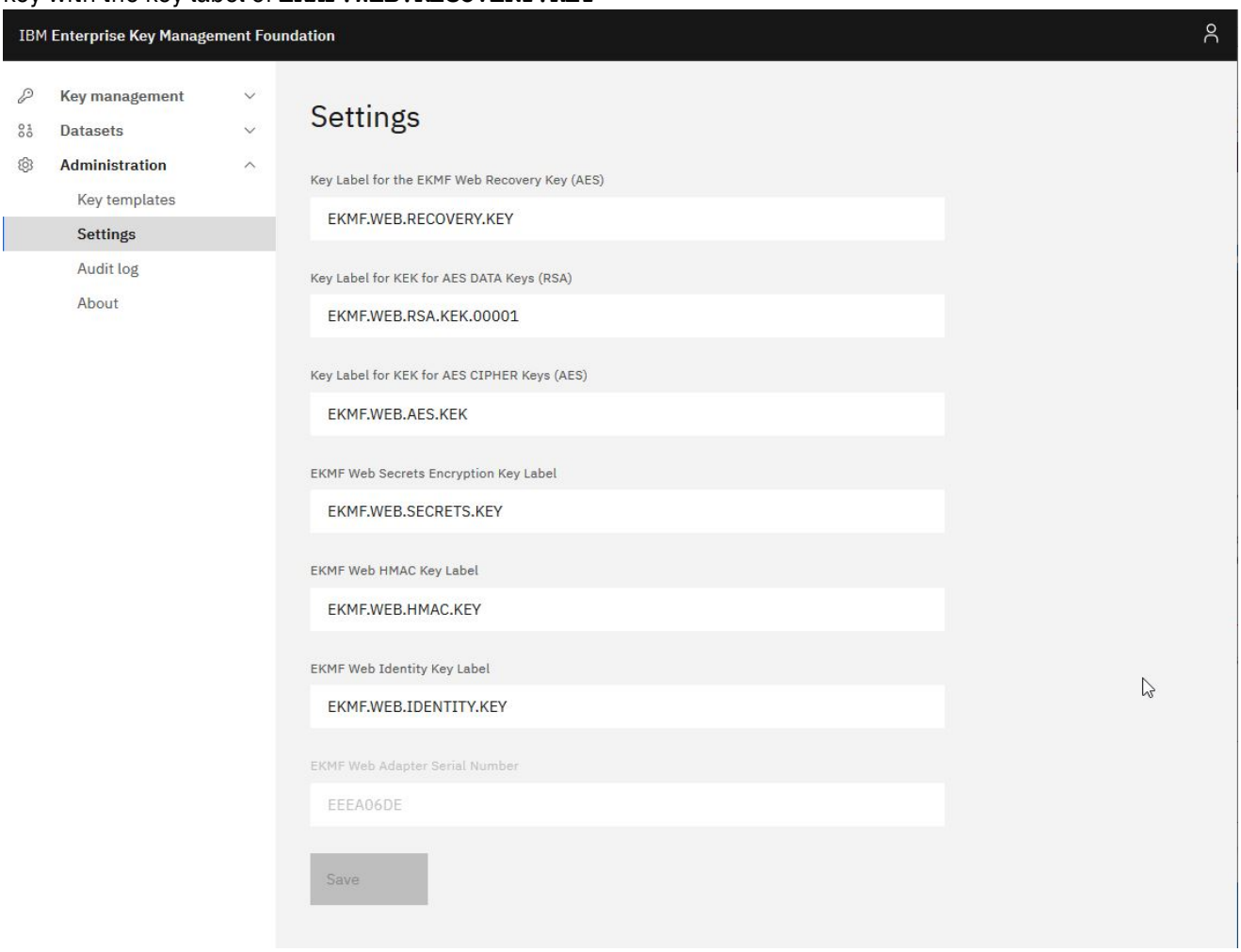
Select a label based on your naming convention, generate the key and import it into the local keystore using ICSF, so you can use it in the Settings.

Setting up hierarchy keys by using the EKMF Web DRK utility

EKMF Web provides a utility that can be used to create a known, fixed-value DRK that can be used for initial testing and isolated proof of concepts. The tool will create a 256 bit AES Importer key with the key value 3DF4B30AE33ED6E22EEBA06359512ACB2B7F9AA006105943C93E8BF30EACF700 and ENC-ZERO key check value '5D7DDC'. In case the DRK is lost from the CKDS, rerunning the utility will recreate the key and allowing recovery of the keys stored in the EKMF repository. As the key value is known, this utility is only appropriate to use for functional testing and proof of concepts. A setup based on this key should not be considered secure. Do not use this key in production.

Remaining setup in EKMF Web

The key label of the recovery key needs to be set in EKMF Web in the Settings window, as shown below for a key with the key label of **`EKMF.WEB.RECOVERY.KEY`**



Whichever way you do the setup, there are five basic steps for setting up the remaining key hierarchy and other required key labels. You simply fill in any missing key labels and keys will be generated upon saving the Settings:

1. You set up the hierarchy keys:
    - **`Key Label for the EKMF Web Recovery Key (AES)`**, the top-level AES **`IMPORTER`** key that enciphers the RSA KEK (Key Encryption Key based on the Rivest-Shamir-Adleman algorithm), as well as the AES KEK (Key Encryption Key based on the Advanced Encryption Standard algorithm)
    - **`Key Label for KEK for AES DATA Keys (RSA)`**, a 4096-bit RSA key that enciphers AES **`DATA`** keys in the database

- **`Key Label for KEK for AES CIPHER Keys (AES)`**, a 256-bit AES key that enciphers AES **`CIPHER`** keys in the database. You can rotate your KEKs by simply updating the labels; they will be auto-generated

2. You set up the EKMF Web Secrets Encryption Key:
   - **`Key label for the EKMF Web Secrets Encryption Key (AES)`**, a 256-bit AES key that enciphers secrets stored in the database. It's auto-generated upon start up with key label **`EKMF.WEB.SECRETS.KEY`**. This is for future use, but has to be set up to save the remaining settings. In case the default key label does not match your key label policies, you can simply write a different key label in the input field
3. You set up the EKMF Web HMAC Key:
   - **`EKMF Web HMAC Key Label`**, a 512-bit AES key used for MAC verification of all other keys stored in the Db2 repository
4. You set up the EKMF Web Identity Key:
   - **`EKMF Web Identity Key Label`**, a 521-bit EC key that identifies the EKMF Web server with the {AGENTS}
5. The cryptographic adapter serial number is auto-generated by EKMF Web. EKMF Web employs the adapter serial number to enable communication with EKMF Agents and to display appropriate information in the audit log. (You do not modify this value in EKMF Web. The value is stored as a Java preference on the server)

Note: All fields on the Settings page have to be filled out, even though not all keys are used depending on your context.

Installation reference material

The following pages contain a set of references.

Database Customization Reference

The KMG.SKMGSQL data set has a naming convention for the member names, offering an easy overview of the content of the members.

The naming convention is as follows:

Member starting with	Description
KMGR	Rexx execs or input to rexx execs
KMGI	Included in one or more JCL samples in this library
KMGJ	JCLs
KMGCS	Create table space
KMGCT	Create table
KMGDS	Drop table space
KMGDT	Drop table

Selected SKMGSQL members:

Member	Description
KMGIMDEF	Include member for dropping, defining of storage group and database, edit carefully

Member	Description
KMGISQLI	Include member for Db2 SQL ID
KMGJCAUD	JCL for TSSAUDIT table (re)-creation
KMGJCELT	JCL for TKMGELT table (re)-creation
KMGJCGRP	JCL for TKMGGRPG/M tables (re)-creation
KMGJCILD	JCL for adding key hierarchy letter
KMGJCPRM	JCL for TSSPARAM table (re)-creation
KMGJCSQL	JCL for first time users creating the EKMF database and all tables
KMGJCUK7	JCL for T_XTEMPLATESPURE and T_XUKDS7 table (re)-creation
KMGJCXC	JCL for T_XCONFIG table (re)-creation
KMGJCXCE	JCL for T_XCERTS table (re)-creation
KMGJCXKE	JCL for T_XKEVENTS table (re)-creation
KMGJCXLO	JCL for T_XLOCKS table (re)-creation
KMGJCXPO	JCL for T_XPOSTBOX table (re)-creation
KMGJCXPY	JCL for T_XPOLICY table (re)-creation
KMGJCXR	JCL for T_XREPORTER table (re)-creation
KMGREDIT	ASIS sample for KMGRUPD rexx exec to edit members of SKMGSQL
KMGRUPD	ASIS rexx to execute KMGREDIT EDIT macros on all SKMGSQL members

Edit the **KMGI***, **KMGC*** and **KMGD*** members to suit your installation naming standards before submitting the **KMGJC*** JCL. The physical table names may be changed to customer standards. However the Db2 view names must be kept unchanged.

All the Db2 tables must be permanent tables. Even though a Db2 table is not accessed, the content must be maintained, and even though a Db2 table may be empty, the table definition should not be erased.

Using the sample JCL that is delivered in the SKMGSQL data set, is optional and may be replaced by procedures already defined for the installation.

ICSF and ACP configuration reference

The following reference table lists

- ICSF services
- Corresponding resources in the CSFSERV class in RACF
- Description of the service
- Corresponding Access Control Point (ACP) number(s) in the crypto adapter, if applicable
- Which of the **<task-user>** needs access
  - A - if the agent needs access
  - L - if the Liberty server needs access
- ACP enablement status based on the ICSF Programmer's Guide, using the following abbreviations:
  - AE - Always enabled, cannot be disabled
  - ED - Enabled by default
  - DD - Disabled by default
  - SC - Usage of this access control point requires special consideration

CSFSERV Resource	ICSF Service	Service Description	ACP	Access for	ACP status

CSFSERV Resource	ICSF Service	Service Description	ACP	Access for	ACP status
CSFCRC	CSFCRC	Coordinate KDS Administration		A	
CSFDEC	CSNBDEC	Decipher	000F	A	ED
CSFDSG	CSNDDSG	Digital Signature Generate	0100	A, L	ED
CSFDSV	CSNDDSV	Digital Signature Verify	0101 01FF	A, L	ED
CSFEDH	CSNDEDH	ECC Diffie-Hellman	0360 0362 0367	A, L	ED
CSFENC	CSNBENC	Encipher --DES	000E	A	ED
CSFIQF	CSNBIQF	ICSF Query Facility		A	
CSFKDMR	CSNBKDMR	Key Data Set Metadata Read		A	
CSFKDSL	CSNBKDSL	Key Data Set Metadata Read		A	
CSFKGN2	CSNBKGN2	Key Generate2	00EB 00EC 00EA	L	ED
CSFKGN	CSNBKGN	Key Generate	008C 00D7 008E	L	ED
CSFKIM	CSNBKIM	Key Import	0012	A	ED
		DATAM Key Management Control	0275	A	ED
CSFKRC	CSNBKRC	Key Record Create	n/a	L	
CSFKRC2	CSNBKRC2	Key Record Create2	n/a	A, L	
CSFKRD	CSNBKRD	Key Record Delete		A	
CSFKRR	CSNBKRR	Key Record Read	n/a	L	
CSFKRR2	CSNBKRR2	Key Record Read2	n/a	A, L	
CSFKRW	CSNBKRW	Key Record Write	n/a	A, L	
CSFKRW2	CSNBKRW2	Key Record Write2	n/a	A, L	
CSFKTB2	CSNBKTB2	Key Token Build2	n/a	L	
CSFKYT	CSNBKYT	Key Test	001D	A, L	AE
CSFKYT2	CSNBKYT2	Key Test2 -- AES, CMACZERO	0021	A, L	ED

CSFSERV Resource	ICSF Service	Service Description	ACP	Access for	ACP status
		Key Test2 -- AES, ENC-ZERO	0022	A, L	ED
		Key Test2 -- DES, CMACZERO	001D	A, L	AE
CSFKYTX	CSNBKYTX	Key Test Extended	001D	L	AE
CSFMGN	CSNBMGN	MAC Generate	0010	A	ED
CSFMVR	CSNBMVR	MAC Verify	0011	A	ED
CSFOWH	CSFPOWH	One-Way Hash Generate	n/a	A, L	
CSFPKB	CSNDPKB	PKA Key Token Build	n/a	L	
CSFPKG	CSNDPKG	PKA Key Generate	0103	A, L	ED
CSFPKI	CSNDPKI	PKA Key Import	0104	A, L	ED
		PKA Key Import -- Import an external trusted block	003A *1)	A	
CSFPKRC	CSNDKRC	PKDS Record Create	n/a	A, L	
CSFPKRD	CSNDKRD	PKA Key record Delete		A	
CSFPKRR	CSNDKRR	PKDS Record Read	n/a	A, L	
CSFPKRW	CSNDKRW	PKDS Record Write	n/a	A, L	
CSFPKX	CSNDPKX	PKA Public Key Extract	n/a	A, L	
CSFPRR2	CSNDRR2	PKA Key record Read2		A	
CSFRNG	CSNBRNG	Random Number Generate	008E	L	ED
CSFRNGL	CSNBRNGL	Random Number Generate	n/a	A, L	
CSFSAD	CSNBSAD	Symmetric Algorithm Decipher	012B	A, L	ED
CSFSAE	CSNBSAE	Symmetric Algorithm Encipher	012A	A, L	ED
CSFSYI	CSNDSYI	Symmetric Key Import - AES, PKCSOAEP, PKCS-1.2	012E	A, L	ED

CSFSERV Resource	ICSF Service	Service Description	ACP	Access for	ACP status
		Symmetric Key Import - AES, ZERO-PAD	012F	A, L	ED
		Symmetric Key Import - Allow wrapping override keywords	0144	A, L	ED
		Symmetric Key Import - DES, PKA92 KEK	0235	A, L	ED
		Symmetric Key Import - DES, PKCS-1.2	0106	A, L	ED
		Symmetric Key Import - DES, ZERO-PAD	023D	A, L	ED
CSFSYI2	CSNDSYI2	Symmetric Key Import2 - AESKW	0329	A, L	ED
		Symmetric Key Import2 - Allow wrapping override keywords	02B9	A, L	ED
		Symmetric Key Import2 - AESKWCV	02B4	A	ED
		Symmetric Key Import2 - AES,PKOAEP2	00FD	A, L	ED
		Symmetric Key Import2 - HMAC,PKOAEP2	00F4	A	ED
CSFSYX	CSNDSYX	Symmetric Key Export	0130 00FC 0131 0327	A, L	ED

Note 1*) - Only needed if your installation needs to install trusted blocks from the EKMF Workstation.

Agent configuration option reference for KMGPARM

This chapter describes the options that affect the EKMF Agent operation. The options are passed to the EKMF Agent via the KMGPARM DD-name in the EKMF procedure.

The sample KMG option member **KMGOPCRY** is included in **KMG.SKMGSAMP**.

All values for these parameters must be defined by the z/OS systems programmer and the EKMF administrator together, taking into account both what is available and possible and the required level of security. Also these values may change over time, as the installation project proceeds, and if new cryptographic hardware is defined in the system.

Parameter	Values	Description
&CRYPTO-ENGINE	**`ICSF`** **`NONE`**	When the value is 'NONE', then the EKMF Agent can only serve the EKMF database, and link encryption is not available.
&IP-PORT		Specifies the IP port, on which the EKMF Agent receives requests
&DB2-USE	**`NO`** **`YES`**	Specifies whether the EKMF Agent acts as a EKMF database server. For EKMF Web this has to be **`'NO'`**, because the Liberty server will communicate with Db2 and not the Agent.
&SYS-CCSID	**IBM1047** IBM037 IBM1047 IBM1141 IBM1142 IBM1143 IBM1144 IBM1145 IBM1146 IBM1147 IBM1148 IBM1149 IBM273 IBM277 IBM278 IBM280 IBM284 IBM285 IBM290 IBM297 IBM420 IBM500 IBM871	The EBCDIC code page which the EKMF client will use, when connecting to the EKMF Agent. Tip: When running the bind -- SKMGSAMP(KMGJBPCK) or SKMGSAMP(KMGJBPCK) then in the output you can see message DSNT255I which indicates the code page EBCDIC(xxxxxx). This is likely to be the code page number you want to use.

Parameter	Values	Description
&SYS-ECCSIGN-PREFIX		This is a 1 - 50 character key label prefix that will be appended a '.' and the user ID of the EKMF Agent. The key label will be a 521 bit ECC signature key that the EKMF Agent will generate or use, if it already exists in the PKDS with the correct attributes. The EKMF Agent displays the SHA-256 value as 64 hex characters of the public key token in the EKMF job log. The EKMF workstation must specify this value to accept a connection to the EKMF Agent. For example: **`&SYS-ECCSIGN-PREFIX(EKMF.SYSTEM.ECCSIGN)`** and EKMF Agent user ID of **`'EKMF'`** results in a PKDS key label of **`EKMF.SYSTEM.ECCSIGN.EKMF`**. Ensure that the EKMF Agent has access to the key label in the **`CSFKEYS`** class. Job log, if the key label exists in PKDS already, and is a correct key: **`System ECC signature key (Existing PKDS entry): - LABEL :`** **`EKMF.SYSTEM.ECCSIGN.EKMF - SHA-256 : 468C5B773B21686C41CE310661B9D3CA3615AC4C624D7F98A2DC6533FFB4C495`** Job log, if the key label does not exist in PKDS: **`System ECC signature key (New PKDS entry): - LABEL : EKMF.SYSTEM.ECCSIGN.EKMF - SHA-256 : 468C5B773B21686C41CE310661B9D3CA3615AC4C624D7F98A2DC6533FFB4C495`**
&WS-AUTH	**`ON`** **`OFF`**	**`'ON'`**: then the EKMF workstation must have an ECC public signature with its SHA-256 value of its public key token, defined for the EKMF Agent. On the EKMF workstation the generated ECC signature key will display its SHA-256 values as 64 hex characters, and in RACF you must define an XFACILIT class profile of **`KMG.WS.<64hex-EKMSws-sha256>`**. **`'OFF'`**: EKMF workstation does not check if the ECC signature key of the EKMF workstation is trusted. However, the value of **`'OFF'`** can only be accepted if the EKMF Agent has READ access to the **`XFACILIT`** class profile **`KMG.WS.AUTHOFF`**.
&WEBCLIENT		This must be used to allow an EKMF Web to connect to the EKMF Agent as a client. The user id defined must be an already defined EKMF client user id or a new user id created for this purpose. The user id will only be assigned if Diffie-Hellman link encryption with **`&WS-AUTH(ON)`** is in place. The EKMF Agent will need access to KMG.WEBCLIENT.**`<client-user>`** and KMG.LG.**`<64-character-hex-fingerprint>`** in the **`XFACILIT`** class
&DEBUG	**OFF** ON	Troubleshooting: Enable tracing information to be written to SYSOUT. 'ON' enables output, 'OFF' disables output. The trace information only specifies a command and the date and time of each command issued from the EKMF workstation. The information can be used to verify the connection from an EKMF workstation with timestamps of the request command flow. The proprietary command interface between the EKMF workstation and the EKMF Agent is not documented.

Parameter	Values	Description
&TRACE-DATAIN	**OFF** ON	Troubleshooting: The parameter is used to enable tracing information to be written to SYSOUT. The value 'ON' enables output, and the value 'OFF' disables output. The default value is 'OFF'. The trace shows the input data received from the EKMF workstation. Use only in debug situations together with IBM support.
&TRACE-DATAOUT	**OFF** ON	Troubleshooting: Enable tracing information to be written to SYSOUT. The value 'ON' enables output, and the value 'OFF' disables output. The trace shows the output data sent from the EKMF Agent to the EKMF workstation. Use only in debug situations together with IBM support.
&SYS-DISPLAY-TOKEN	**OFF** ON	Troubleshooting: When the value is 'ON', the public key token of the key label from **`&SYS-ECCSIGN-PREFIX`** and **`&SYS-RSAKEK-PREFIX`** is displayed in the EKMF Agent job log.
&SYS-RSAKEK-PREFIX		This is a 1 - 50 character key label prefix that will be appended a '.' and the user ID of the EKMF Agent. The key label will be a 4096 bit RSA key management-only key that the EKMF Agent will generate or use, if it already exists in the PKDS with the correct attributes. The EKMF Agent displays the PKDS label of the private key. The EKMF workstation will receive the public key token together with the label of the private key. The EKMF workstation will use the public key to wrap AES data keys for the EKMF Agent to import. Example: **`&SYS-RSAKEK-PREFIX(EKMF.SYSTEM.RSAKEK)`** and EKMF Agent user ID of **`'EKMF'`** results in a PKDS key label of **`EKMF.SYSTEM.RSAKEK.EKMF`**. Job log, if the key label exists in PKDS already, and is a correct key: **`System RSA KEK key (Existing PKDS entry): - LABEL : EKMF.SYSTEM.RSAKEK.EKMF`** Job log, if key label does not exists in PKDS: **`System RSA KEK key (New PKDS entry): - LABEL : EKMF.SYSTEM.RSAKEK.EKMF`** Ensure that the EKMF Agent has access to the key label in the **`CSFKEYS`** class.
&KDS-ADMNREAD	**NO** YES	It is recommended to set this parameter to **`'YES'`**. When specifying 'YES', the z/OS agent will use the rule array 'ADMNREAD' for the CSNBKRR2 and CSNDKRR2 services. The idea of using 'ADMNREAD' is that an administrator through the EKMF Agent can browse the CKDS and PKDS keys, without having ICSF setting the metadata field 'Last used reference date'. If specifying 'YES', and ICSF is not updated to support the 'ADMNREAD' rule, then either the EKMF Agent will end with an abend 806 for module CSNDKRR2, or return an ICSF error 8/2012 for using CSNBKRR2. Support was added for 'ADMNREAD' in ICSF with APAR OA49503.
&SELFTEST-CSF	**OFF** ON	When ON is specified then the EKMF Agent user id checks its access for the required CSFSERV resource. This is done when EKMF Agent starts. It also enables the modify command **`SELFTEST,CSF`** to be issued and then the tests will be done again.

Parameter	Values	Description
&SELFTEST-CLIENT	**<userid>**	Can be used to verify if a defined user id has access to logon to the EKMF Agent. When a user id is specified, then tests are done to see if: User id can logon to the EKMF Agent User id can disable the use of SMF logging (AUDITOFF access). User id can logon to the EKMF Agent via the browser. It also enables the modify command **SELFTEST,CLIENT** to be issued and then the tests will be done again.
&ECDH-AESKEKONLY	**OFF** ON	When 'ON' then no DES KEK is derived for the ECDH link session. An ECDH installed DES key must use the ECDH AES KEK and a DESUSECV external token to import a DES key. The EKMFws must have the support for change 20416 implemented. EKMFws level 9.2.1. For EKMF the recommended value is **'ON'**, for EKMF Workstation it's **'OFF'**
&ECDH-DOUBLEO	**OFF** ON	Normally this should not be specified -- and ignored for &ECDH-AESKEKONLY(ON). The only situation where 'ON' should be used is when the EKMFws level is below 9.1.4 and the ECDH KEK created for Diffie-Hellman install is required to be a DOUBLE-O length KEK instead of a DOUBLE length KEK. Note: With the EKMFws level 9.1.4 and later a TRIPLE-O length KEK is created if the ICSF/HW supports triple length DES keys, otherwise a DOUBLE-O length KEK is created.

{: caption="KMGPARM options" caption-side="top"}

Messages and troubleshooting

The following sections contain troubleshooting information and messages.

Troubleshooting

In case the EKMF Web application doesn't start as expected, looking in the **/var/log/ekmfweb/messages.log** file or in some cases in your web browser log should indicate what is missing or possibly incorrectly configured.

Clear cache and cookies

One action which is sometimes required is to clear the browser cache and cookies, when moving from one release to the next.

The Liberty recovery logs and work area need to be cleared

In some cases the following directories are blocking for the Liberty server recovery function. If so simply delete them as they are autocreated on startup:

- **$WLP_USER_DIR/servers/ekmfweb/logs**
- **$WLP_USER_DIR/servers/ekmfweb/tranlog**
- **$WLP_USER_DIR/servers/ekmfweb/workarea**

RACF related issues

If connection to the EKMF Agent cannot be established and an error description

```
KMGPARM &WEBCLIENT not defined in EKMF Agent task or user not authorized.
```

is shown in **messages.log** file it's indirectly a missing RACF authorisation. The *EKMF Agent Configuration and Operation Guide* (SC28-2024-00), describes the need for defining the &WEBCLIENT parameter when using EKMF Web. The parameter has to define which user ID EKMF Web will use as Agent client id (and granted correct RACF access) to connect to the EKMF Agent. If it's undefined a default user is assumed, which most likely doesn't have the correct access and the resulting error is:

```
E Error during connection setup. KMGPARM & WEBCLIENT not defined in EKMF Agent
task or user not authorized.
```

The solution is to define the parameter as described in the *EKMF Agent Configuration and Operation Guide* (SC28-2024-00), restart the EKMF Agent and restart the EKMF Web Liberty Server.

Another RACF error is logged in **messages.log** as:

```
37,4,0, DKMS Host User is NOT (RACF) permitted to invoke host task (KMGPTRAN)
(Corresponding to the selected IP address/IP port).
```

This is because the user ID defined to run the Liberty Server is not granted **READ** access to the RACF profile **KMG.EKMF.KMGPRACF.<task-user>** in class FACILITY where <task-user> is the user ID running the EKMF Agent task. Grant this access and restart the Liberty Server to solve the issue.

Timezone issues

An error that can be shown in browser log is related to having different timezones on the workstation and the z/OS. No error shows on the screen, but observing the browser log, will show the following warning:

```
authorizedCallback Validation, iat rejected id_token was issued too far away from
current time
authorizedCallback, token(s) validation failed, resetting
```

This happens in case the machine where the web browser runs has a different time compared to the z/OS system where the EKMF Web application is installed. Adjust machine to automatically use the correct timezone and ensure time is correctly set, then try to log on again.

JCCA errors in /var/log/ekmfweb/messages.log file

When attempting to generate the Web Identity Key an error occurs, which in the /var/logs/ekmfweb/messages.log is shown partly as:

```
JCCA Exception msg : jcca16_64 (Not found in java.library.path)
JCCA Exception String : java.lang.UnsatisfiedLinkError: jcca16_64 (Not found in
java.library.path)
```

This is either because the proper ICSF FMID has not been linked to libjcca16_64.so or because the libjcca16_64.so file was not in the expected directory. Verify that the link command was indeed executed in the correct directory and that both the FMID file and the libjcca16_64.so file were in this same directory.

Performance issues

In case the EKMF Web application seems to run slower than expected the following can be attempted:

- In case you have unpacked the .war files, pack them again with ZIP to have a compressed .war file
- move the compressed .war files to a **READ ONLY** mounted ZFS (on $WLP_USER_DIR/servers/ekmfweb/apps)

- add the following options to `$WLP_USER_DIR/servers/ekmfweb/jvm.options` file:
  - `-Xcompressedrefs`
  - `-Xquickstart`
  - `-Xjit:noResumableTrapHandler`

Agent messages

The following list describes the messages issued by the EKMF Agent. The messages are written to the SYSOUT DD name for the started task. Selected messages are also written to the system log as WTO messages, usable for automated operation. The WTO messages are issued with routing code 2, 9 and 11.

The message number followed by 'I' signifies an informational message, 'E' signifies an error message and 'W' signifies a warning message.

Message ID	Description
KMG001I	EKMF Agent initialization has completed. This is a WTO message. When this message is issued, the initialization of the EKMF Agent started task is completed.
KMG002E	EKMF Agent abnormal termination. This is a WTO message. This message is issued when an error has occurred during initialization of the EKMF started task, or an unrecoverable error has occurred after initialization. Other messages may be issued describing the actual error. Inspect the system log and the SYSOUT for the EKMF started task to retrieve additional information.
KMG003E	Failure to access TSSPARAM Db2 table. SQL Code: xx. SQLCA Data Structure: xxxxxx. The SQL error code is described in the manual 'Messages and codes' for DB2. Typical errors are bind errors and access to the Db2 plan/package errors.
KMG004I	EKMF Agent termination due to DB2 termination. This is an expected message when Db2 stops. Restart the EKMF Agent again when Db2 is available.
KMG007E	TCP/IP Error. Socket Function: xxxxxx. Socket Error: xx. The socket function is the sub-function to the z/OS TCP/IP callable service EZASOKET. A description of the socket error (xx) can be found in the document 'IP Application Programming Interface Guide, appendix B' (z/OS Communication Server). The socket error corresponds to the error number (errno) from the documentation. The socket error '48', address and TCP/IP port in use, may occur, when the EKMF Agent is stopped and started within a short interval (0 - 15 seconds). Wait some seconds before restarting the task.
KMG008W	TCP/IP Time-out on reading data from client. An incomplete request has been received from a client. This message will, for example,. occur if a client is sending data to the EKMF Agent, but is not using the EKMF proprietary communication protocol correctly. This message could indicate an unauthorized attempt to access the EKMF Agent.
KMG009W	TCP/IP Connection closed by client. Could be the EKMF client abruptly ending, or some network layer stopping the connection.

Message ID	Description
KMG010W	Invalid transaction - Length not numeric. This message will, for example, occur if a client is sending data to the EKMF Agent, but is not using the EKMF proprietary communication protocol correctly. This message could indicate an unauthorized attempt to access the EKMF Agent.
KMG011W	Length too short/long. Length: xxxx A request from a client is received. The length of the request is either too short or too long for the the EKMF proprietary communication protocol. This message could indicate an unauthorized attempt to access the host.
KMG012W	TCP/IP Time-out on reading data from client. An incomplete request has been received from a client. This message will, for example, occur if a client send data to the EKMF Agent; but is not using the EKMF proprietary communication protocol, or an EKMF client has abruptly ended. This message could indicate an unauthorized attempt to access the host.
KMG013W	TCP/IP connection closed by client. Could be the EKMF client abruptly ending, or some network layer stopping the connection.
KMG014W	Time-out during sending data to client. It was not possible for the EKMF Agent to reply to a request from a client.
KMG015E	Failed to access ICSF - CSFIQF function failed. Return code: xx Reason code: yy. The return code and reason code can be found in the ICSF Application Programmer's Guide, Appendix A.
KMG018I	EKMF Agent terminated by command. This is a WTO message. The started task has been purged, or canceled, or stopped upon request from the operator command.
KMG019W	Link Encryption is not required. This message indicates that the SAF FACILITY resource KMG.EKMF.LNKCRYOFF is permitted to the EKMF Agent user ID. The current status of the link encryption between the EKMF client and the EKMF Agent is not required and is not determined by the EKMF client. See also the message KMG020I.
KMG020I	Link encryption is required. Link encryption between the EKMF workstation and the EKMF Agent is required. The EKMF client user cannot connect to the EKMF Agent without link encryption. See also the message KMG019W.
KMG021E	Failed to open KMGPARM DD. Return code: xx. The EKMF Agent failed to open the options data set. The return code specifies the I/O return code for the Open command.
KMG022E	Failed to read from KMGPARM DD. Return code: xx. The EKMF Agent to read from the option data set. The return code specifies the I/O return code for the read command
KMG023W	Audit is not enforced. The EKMF Agent user ID is permitted to FACILITY resource KMG.EKMF.AUDITOFF. If the EKMF client user ID is also permitted, then the client user can disable logging to SMF for EKMF events.
KMG024I	Audit is enforced. The EKMF Agent user ID is NOT permitted to FACILITY resource KMG.EKMF.AUDITOFF. The EKMF client user ID cannot disable logging to SMF for EKMF events.
KMG025E	AUDITOFF query returned unexpected: error-type, RACF rc, RACF rs.

Message ID	Description
KMG050E	KMGPRACF auth. Program not found in an auth library. The APF KMGPRACF authorized program not found in an authorized library. This is a WTO message.
KMG051E	APF auth. program KMGPRACF was invoked from non TSO/E. The APF authorized program KMGPRACF was invoked from a nonTSO/E environment. This is a WTO message.
KMG052E	Failed to call the APF authorized module KMGPRACF. This is a WTO message. See KMG053E also.
KMG053E	Error from TSO/E service facility routine TSOLNK/IKJEFTSR. Error type: 36 return code yy. The return code yy can be found in the manual TSO/E Programming Services for the TSO/E service facility IKJEFTSR.
KMG054E	Error from RACF/Security server. RACF return code : xx. The return code xx can be found in the RACF Messages and Codes. For return code 8 it will additionally display: Access to the resource not permitted to task user. Resource KMG.EKMF.KMGPRACF.
KMG055E	The module KMGPRACF is not running authorized. Check SYS1.PARMLIB(IKJTSOxx) or similar data set for registration of the APF authorized program. The table AUTHTSF must specify the KMGPRACF module. A refresh of the table is required whenever this table is changed. If the module name is present in the table, the refresh may not have been performed yet.
KMG057E	The TCP/IP network is down. Try to start the EKMF Agent again when the network is up.
KMG058E	The TCP/IP port is in use. Port: <port> Try to start the EKMF Agent again after a short while. Another task is using the port, or the port is not yet released by TCP/IP from a former instance of the EKMF Agent.
KMG101I	CSFCRC ICSF refresh OK for CKDS/PKDS This is a WTO message.
KMG102E	CSFCRC ICSF refresh failed for CKDS/PKDS – return-code, reason-code This is a WTO message.
KMG201E	TSOLNK-KMGPRACF Error. A=XX,B=YY,C=ZZ. An error is returned from the TSOLNK module. The XX code retrieved from the TSOLNK parameter 4, the YY code is retrieved from TSOLNK parameter 5, and the ZZ code is retrieved from parameter 6. The return code XX, YY and ZZ can be found in the manual TSO/E Programming Services for the TSO/E service facility IKJEFTSR.
KMG240I	EKMF Agent console interface ready The EKMF Agent is now ready to receive modify and stop commands from the MVS console.
KMG250I	ICSF FMID FROM CCVT : CCVTID/VER: xxxxxx CCVTFMID: HCRxxxx
KMG251I	TCP/IP host name is <host name>
KMG252E	Fail to retrieve ICSF CCVT information. TYPE=<RACF-ERROR-TYPE> RC=<RACF-RETURN-CODE> RE=<RACF-REASON-CODE>

Message ID	Description
KMG253I	No key hierarchy letter in the database SQLCODE = <sqlcode> Select PTSS_TEST_MODE from view VTSSPARAM where PTSS_BFC_NO = '00'. When not set, the SQLCODE will be 100. It is perfectly valid not to have a key hierarchy letter defined in the database, however, some features require it.
KMG254I	Key hierarchy letter = <key hierarchy letter> Select PTSS_TEST_MODE from view VTSSPARAM where PTSS_BFC_NO = '00'
KMG255E	SQLCODE = <rc> getting plan qualifier Module KMGPQIBM needs to be bound to the Db2 plan to get the qualifier from SYSIBM.SYSPLAN. Consider using the KMGPARM option &DB2-QUALIFIER instead, if the problem persists. Restart the EKMF Agent when either the &DB2-QUALIFIER is been specified or KMGPQIBM successfully retrieves the qualifier from the Db2 plan.
KMG0256I	Plan qualifier = <qualifier or schema> This is the table and view qualifier by which the EKMF database tabels and views are found.
KMG257E	Plan qualifier and creator is 0 or >24 chars. Use a qualifier with a maximum of 24 characters for the EKMF database tables and views Restart the EKMF Agent when either the &DB2-QUALIFIER is been specified or KMGPQIBM successfully retrieves the qualifier from the Db2 plan.
KMG258E	KMGPARM OPTION(S) FOR JDBC MISSING When &DB2-USE(YES) is specified then the EKMF Agent must be configured with: - &SYS-JDBC-LOCATION - &SYS-JDBC-PORT and/or &SYS-JDBC-SECPORT - &DB2-QUALIFIER or qualifier extracted through the plan using KMGPQIBM Restart the EKMF Agent when above is corrected.
KMG259E	Error getting enhanced wrapping status. - RETURN-CODE = <rc> - REASON-CODE = <rs>
KMG260I	KMG-LEVEL=<xxxxxx> <change-date> <FMID>. <xxxxxx> = KMG level of code in use, the level of KMGPTRAN. <change-date> = The date KMGTRAN was changed. <FMID> = the current FMID.
KMG261I	SYSINFO: - ZOS REL <vvrrmm> - Z SERIAL <serial> - PLEX <plex-name> - SYSTEM <system-name> - LPAR <lpar.name> - SMFID <smd-id>
KMG270I	Enhanced wrap for DES not available.
KMG271I	Enhanced wrap for DES available: - DEFAULT FOR INTERNAL TOKENS = <ECB/CBC> - DEFAULT FOR EXTERNAL TOKENS = <ECB/CBC> ECB is ORIGINAL, and CBC is ENHANCED.
KMG300E	DSNALI failed to connet to Db2 system <system>. RETURN CODE: xxxxxxxx REASON CODE: xxxxxxxx

Message ID	Description
KMG301E	DSNALI failed to open plan <plan> on Db2 system <system>. RETURN CODE: xxxxxxxx REASON CODE: xxxxxxxx
KMG303I	Db2 ppppyyymm VERSION xxxx. Information about the Db2 program level and version in use.
KMG304I	Old adapter session had RESERVE flag. Relieving ROLLBACK done, SQLCODE = xxxxxx
KMG310I	Db2 termination in progress TECB = xxxx. When the EKMF Agent is stopped by command or stops because Db2 stops, then Db2 the disconnect process will start.
KMG311I	DSNALI close abort successful.
KMG312E	DSNALI failed close RETURN CODE: <rc> REASON CODE: <rs>
KMG313I	DSNALI disconnect successful.
KMG314E	DSNALI failed disconnect RETURN CODE: <rc> REASON CODE: <rs>
KMG330I	Task termination without waiting modify interface termination. The EKMF decided not to wait for the MVS console interface to end. This can happen when system resources are busy. No further action is required.
KMG331I	Modify detach ended in error rc = <rc>
KMG332I	Possible abend like 33E can be expected and is harmless. Message comes together with KMG331I.
KMG400I	Received command : <cmd>. This is a WTO message. A valid command was entered, and the EKMF Agent to process
KMG401W	Received invalid command : <cmd>. This is a WTO message. The command received is invalid, and the EKMF Agent ignores the input.
KMG801I	Session <sess#> ended due to reserve timeout, user = <user ID>. A result of &RESERVE-WAIT-MIN in KMGPARM where the session was holding an EKMF reserve without activity
KMG802I	Db2 ROLLBACK done for reserve time out.
KMG803W	DBE ROLLBACK error = <sqlcode>.
KMG807I	Module/version list: A list of EKMF Agent modules and program levels from the SKMGMOD0 library.
KMG808I	EKMF settings: Lists output of KMGPARM options and configuration keys used.
KMG820I	&SELFTEST-DB2 not ON – test ignored - or - &SELFTEST-CSF not ON – test ignored -or - &SELFTEST-CLIENT not specified – test ignored
KMG821I	SELFTEST,DB2 cannot run when while an EKMF session has a reserve
KMG822I	SELFTEST,DB2 cannot run when &DB2-USE(NO)
KMG829I	Start testing access to CSFSERV resources
KMG830I	End testing access to CSFSERV resources
KMG831I	<csfserv-resource> OK

Message ID	Description
KMG832E	<csfserv-resource> No access (Permit the EKMF Agent user id to the resource)
KMG839I	Start testing client access for <user id>
KMG840I	End testing client access for <user id>
KMG849I	Start testing access to the EKMF tables
KMG850I	Test Db2 success commit
KMG851E	Test Db2 failed <action> (Check the error codes listed. For ERROR-TYPE = 3 the RETURN-CODE would be an SQL code)
KMG852I	All Db2 tests successful
KMG853I	Db2 qualifier from PLAN or &DB2-QUALIFIER is OK - or - <view name> view is OK
KMG854E	Db2 qualifier from PLAN or &DB2-QUALIFIER is in error - or - <view name> view error (Check the KMG851E message)
KMG855E	All Db2 tests done – errors occurred
KMG856I	SYSTEM EBCDIC SBCS CCSID = <ccsid> From Db2 GETVARIABLE('SYSIBM.SYSTEM_EBCDIC_CCSID') (This is most likely the EBCDIC codepage you want to specify in the EKMFws device configuration)
KMG857W	SYSTEM_EBCDIC_CCSID RETURNED NULL
KMG858I	ICSF CSFIQF ICSFST2 DATA: (Text messages showing master key status)
KMG859I	KMG859I DOUBLE-O suggested as DES ECDH KEK or KMG859I TRIPLE-O suggested as DES ECDH KEK
KMG860I	Db2 SQLCODE =' <sqlcode> ' at loc = <pgmname>/<loc> :SQLERRMC = <SQLERRMC> This message can give additional information to debug non trivial SQL codes.