

z/OS



# RACF Password enveloping and LDAP event notification enhancements support

APAR OA03853

10/28/2003

Document last updated 4/26/04



# Contents

General information . . . . .	v
<b>Part 1. Overview . . . . .</b>	<b>1</b>
<b>Chapter 1. Overview . . . . .</b>	<b>3</b>
<b>Chapter 2. Software . . . . .</b>	<b>5</b>
<b>Part 2. Publication updates . . . . .</b>	<b>7</b>
<b>Chapter 3. Security administrator considerations . . . . .</b>	<b>9</b>
LDAP notification for user status changes . . . . .	9
LDAP notification occurs in real-time only . . . . .	10
RRSF considerations for applications exploiting this function . . . . .	10
Activating LDAP change notification . . . . .	10
Overview of password enveloping . . . . .	10
The PASSWORD.ENVELOPE Profile. . . . .	11
Signing hash algorithm and encryption strength used to create the password envelope . . . . .	11
The IRR.PWENV.KEYRING key ring . . . . .	12
The IRR.RADMIN.EXTRACT.PWENV Profile . . . . .	13
The NOTIFY.LDAP.USER profile . . . . .	13
Setting up password enveloping . . . . .	13
Preparing the system . . . . .	13
Preparing RACF . . . . .	13
Defining a local certificate authority certificate using RACF as the certificate authority . . . . .	14
Generating a X.509V3 certificate for RACF's use for user password enveloping. . . . .	14
Activating password enveloping. . . . .	18
Enabling RACF for heterogeneous password synchronization. . . . .	19
<b>Chapter 4. Migration considerations . . . . .</b>	<b>21</b>
Purpose . . . . .	21
Process . . . . .	21
Other . . . . .	21
<b>Chapter 5. Data area considerations . . . . .</b>	<b>23</b>
<b>Chapter 6. Callable services considerations . . . . .</b>	<b>25</b>
R_admin (IRRSEQ00): RACF Administration API . . . . .	25
Function . . . . .	25
Requirements . . . . .	25
RACF authorization . . . . .	25
Format . . . . .	26
Parameters . . . . .	26
Return and reason codes . . . . .	29
Usage notes . . . . .	29
Parameter list formats . . . . .	30
R_proxyserv (IRRSPY00): LDAP interface . . . . .	32
Function . . . . .	32
Requirements . . . . .	33

Linkage Conventions . . . . .	33
RACF Authorization . . . . .	33
Format . . . . .	33
Parameters . . . . .	33
Return and Reason Codes . . . . .	36
Parameter Usage . . . . .	37
Usage Notes. . . . .	37
Related Services . . . . .	37
<b>Chapter 7. Messages considerations . . . . .</b>	<b>39</b>
<b>Chapter 8. Macros and interfaces considerations . . . . .</b>	<b>43</b>
Updates to the RACF database templates . . . . .	43
Updates to database unload . . . . .	43
Class descriptor table updates . . . . .	43
Router table updates. . . . .	43
<b>Chapter 9. Command considerations . . . . .</b>	<b>45</b>
SET Command. . . . .	46
<b>Chapter 10. System programmer considerations . . . . .</b>	<b>49</b>
The RACF subsystem . . . . .	49
<b>Trademarks . . . . .</b>	<b>51</b>

---

## General information

This information applies to APAR OA03853.



---

## Part 1. Overview



---

## Chapter 1. Overview

This enhancement provides two separate functions:

- The ability to create LDAP change log entries in response to updates to RACF user profiles
- The ability to create recoverable user passwords

These two functions can be used separately, or they can be used in conjunction by an application such as IBM Directory Integrator to provide a heterogeneous password synchronization solution.



---

## Chapter 2. Software

The enhancements to this support require one of the following:

- z/OS™ Version 1 Release 4 Security Server (RACF) (HRF7707), and later releases, including APAR OA03853 and SAF APAR OA03854
- z/OS Version 1 Release 3 Security Server (RACF) (HRF7706), including APAR OA03853 and SAF APAR OA03854

For full exploitation of these enhancements, use the appropriate level from one of the following:

- z/OS Version 1 Release 4 Security Server (LDAP) (HRSL340) and later releases, including APAR OA03857
- z/OS Version 1 Release 3 Security Server (LDAP) (HRSL320) including APAR OA03857



---

## Part 2. Publication updates

The chapters in this part supplement the following books:

<b>Chapter</b>	<b>Supplements</b>
Chapter 3, "Security administrator considerations," on page 9	<i>z/OS Security Server RACF Security Administrator's Guide</i>
Chapter 4, "Migration considerations," on page 21	<i>z/OS Security Server RACF Migration</i>
Chapter 5, "Data area considerations," on page 23	<i>z/OS Security Server RACF Data Areas</i>
Chapter 6, "Callable services considerations," on page 25	<i>z/OS Security Server RACF Callable Services</i>
Chapter 7, "Messages considerations," on page 39	<i>z/OS Security Server RACF Messages and Codes</i>
Chapter 8, "Macros and interfaces considerations," on page 43	<i>z/OS Security Server RACF Macros and Interfaces</i>
Chapter 9, "Command considerations," on page 45	<i>z/OS Security Server RACF Command Language Reference</i>
Chapter 10, "System programmer considerations," on page 49	<i>z/OS Security Server RACF System Programmer's Guide</i>



---

## Chapter 3. Security administrator considerations

---

### LDAP notification for user status changes

RACF can be configured to create LDAP change log entries in response to changes to RACF user profiles. This provides an open, remote method of change notification. An LDAP client can read the LDAP change log, detect updates to RACF users, and retrieve RACF user entries using only LDAP interfaces.

Event notification, through the creation of an LDAP change log entry, is controlled by the NOTIFY.LDAP.USER resource in the new RACFEVNT class. If RACFEVNT is active, and the NOTIFY.LDAP.USER resource is protected (by either a discrete or generic profile), then an LDAP change log entry will be created for the following types of user updates.

1. Password changes, regardless of the interface used, as long as the new password is enveloped (see *Overview of password enveloping* section for a description of the password enveloping function, including the conditions under which a password is enveloped)
2. Updates to a user's revoke status (that is, changes to the FLAG4 field in the USER profile), regardless of the interface used
3. User additions made using the ADDUSER command
4. User modifications made using ALTUSER and PASSWORD commands
5. User deletions made using the DELUSER command

NOTIFY.LDAP.USER acts on a system-wide basis.

**Note:**

1. The RACF panels and the R\_admin interface issue TSO commands internally, so these interfaces are included. Note that the RACF panels may generate multiple commands while processing a user profile, and this may result in multiple change log entries.
2. Changes to group connection information are not currently supported. These changes are normally performed using the CONNECT and REMOVE commands. The UACC and AUTHORITY operands of the ALTUSER command also update group connection information, and these are also not supported.
3. Other RACF commands that update user profiles (e.g. RACDCERT, RACLINK) are not currently supported.
4. An application which updates user profiles through means other than TSO commands can create its own change log entry using a new function of the R\_Proxyserv callable service (IRRSPY00), documented in *z/OS Security Server RACF Callable Services*.

The LDAP change log entry will contain information such as the change initiator, the affected user, the type of update (add, modify, or delete), and the time and date of the change. It will not contain a list of fields which were changed, nor will it contain the new values for these fields.

In the case of a password change, a change log entry will only be created if the user's password was enveloped (see *Overview of password enveloping* section). This change log entry will contain the 'changes' attribute identifying the password field as the changed field. The 'changes' attribute will not contain the actual

password value, but will contain a value of `**ComeAndGetIt**`, denoting that there is an encrypted password envelope which can be subsequently retrieved.

If other RACF profile fields are changed in the same request which updates the password (e.g. `ALTUSER SOMEUSER PASSWORD(NEWPASS) RESUME`), then two change log entries are created: one to describe the password update, and another to describe the non-password update).

See LDAP documentation, APAR OA03857, for information on the change log.

## LDAP notification occurs in real-time only

If the LDAP server is unavailable at the time the RACF change occurs, then that change log entry is lost. There is currently no queuing mechanism whereby a change notification can be retried at a later point in time. This does not affect the RACF database itself; LDAP notification is attempted only after the RACF database has been updated.

## RRSF considerations for applications exploiting this function

Applications which exploit LDAP change log entries for registry synchronization must take network topology into account when propagating locally initiated RACF changes to other z/OS/RACF systems in the network. In particular, if RACF is configured in an RRSF network, and user or password updates are being kept in sync across RRSF nodes, then application deployment must include consideration of which propagation mechanism is used for specific types of changes to specific systems. Neglecting the interaction of the various propagation mechanisms could result in an unending cascade of updates for the same RACF change. For example, for an RRSF network which fully mirrors user profile and password updates, an LDAP based propagation mechanism should only communicate with a single RRSF node, and let that node propagate the change to other RACF nodes. Further, this RACF node should be the only node configured to perform LDAP event notification for user updates.

## Activating LDAP change notification

- Define the RACFEVNT class profile named NOTIFY.LDAP.USER.  
`RDEFINE RACFEVNT NOTIFY.LDAP.USER`  
A generic profile could also be used.
- Activate the RACFEVNT class.  
`SETROPTS CLASSACT(RACFEVNT)`

---

## Overview of password enveloping

RACF can be configured to save user passwords such that the clear text can be recovered by an authorized application. This ability can be restricted to a subset of your users. When an eligible user's password is changed, the new password is encrypted under a public key which is contained within a key ring associated with the RACF subsystem address space identity. The encrypted password is then stored in the user's profile. When an application requests the password, RACF decrypts the password, and then encrypts it in PKCS#7 format for recipients whose digital certificates have been placed on the same RACF key ring. The application can then decrypt this 'password envelope' using their private key.

The R\_Admin callable service (IRRSEQ00) provides the interface by which an application can retrieve the password envelope. See *z/OS Security Server RACF Callable Services* for interface documentation, including a description of the envelope structure.

For the most part, any new password will be enveloped for an eligible user, with the following exceptions:

- Initial ADDUSER passwords
- When the new password is the same as the current password
- When the ALTUSER or PASSWORD command is used to change the password, and the new password is equal the user's default group name
- When an application uses RACROUTE or ICHEINTY (as opposed to a RACF command) to set the password, and the password contains characters which would not be accepted by the RACF commands. RACF commands only accept the characters 'A'-'Z', '0'-'9', and the variant characters X'5B' (typically '\$'), X'7B' (typically '#'), and X'7C' (typically '@')
- When an application uses RACROUTE or ICHEINTY to set the password and specifies ENCRYPT=NO

**Note:** When the password enveloping function is used, approximately 280 bytes of storage will be used in the USER profile to contain the enveloped password.

## The PASSWORD.ENVELOPE Profile

The PASSWORD.ENVELOPE profile in the RACFEVNT class controls whether new passwords are enveloped for a given user. If the user whose password is being changed has at least READ access to this resource, then the new password is enveloped. Thus, you can use this profile to subset the users whose passwords will be enveloped. For example, you can exclude sensitive user IDs from password enveloping, if necessary.

**Note:**

- Generic characters may not be used in this profile name.
- The enveloped password will neither be displayed by LISTUSER nor unloaded by IRRDBU00, although IRRDBU00 will indicate the presence of a password envelope for a given user with a YES/NO field. See Chapter 8, "Macros and interfaces considerations," on page 43 for more information.
- There will be no SMF records created as a result of failed access checks to this resource. Audit options in the profile can be used to log successes, and thus maintain a history of whose passwords have been enveloped.
- If the user cannot be verified (RACROUTE REQUEST=VERIFY), then a new password will not be enveloped. For example, if a revoked user's password is changed by an administrator, the new password will not be enveloped. If you want the new password to be enveloped in this scenario, specify RESUME on the same ALTUSER command in which the PASSWORD keyword is used to specify the new password.

## Signing hash algorithm and encryption strength used to create the password envelope

Both the signing hash algorithm and encryption strength are configurable attributes. The APPLDATA of the PASSWORD.ENVELOPE profile is used to specify the signing hash algorithm used to sign the PCKS#7 password envelope, and the

encryption strength used when encrypting the password envelope. The syntax of the APPLDATA string consists of a character string indicating the signing hash algorithm, followed by a forward slash("/"), followed by a string indicating the encryption strength.

The following values are allowed for the signing hash algorithm:

- MD5
- SHA1
- Default=MD5

The following values are allowed for the encryption strength are:

- STRONG
- MEDIUM
- WEAK
- Default=STRONG

It is recommended that you use the strongest encryption possible. If you are forced to use weaker encryption, for example, due to export regulations, then protect yourself against offline attacks by carefully controlling access to the RACF database, and any other repository in which password envelopes may be stored after being retrieved from RACF.

Table 1. Encryption strength values

Value	Data encryption method
STRONG	Triple DES. A 168-bit encryption key
MEDIUM	DES. A 56-bit encryption key
WEAK	RC2. A 40-bit encryption key

**Note:** Strong encryption may not be available on all customer systems depending on government export regulations. See *z/OS System SSL Programming* for more information.

If the APPLDATA is not specified in the profile, the defaults are taken as noted above. If an empty qualifier exists in the APPLDATA, then the default value will be taken for that qualifier. For example, if the APPLDATA is specified as 'SHA1', then SHA1 will be used as the signing hash algorithm, and triple DES will be used as the encryption algorithm. If the APPLDATA is specified as '/MEDIUM', then MD5 will be used as the signing hash algorithm, and DES will be used as the encryption algorithm.

If the APPLDATA is specified incorrectly, an error message will be issued to the console the next time a user who is eligible for password enveloping changes his password, or the next time an application requests the retrieval of a password envelope, and the defaults will be used.

The APPLDATA can be changed at any time.

## The IRR.PWENV.KEYRING key ring

IRR.PWENV.KEYRING is the name of a key ring which is associated with the identity of the RACF subsystem address space (RASP). It contains a certificate with private key for the RASP itself. This certificate is used to encrypt a user's new

password. It is also used to decrypt the stored password when a PKCS#7 envelope is retrieved by an authorized application. The contents of the returned envelope are signed using this certificate.

IRR.PWENV.KEYRING also contains certificates of all the principals who are intended to retrieve a user's changed password from RACF. A changed password is encrypted using the public keys contained within these certificates. RACF will encrypt passwords for up to 20 certificates on this key ring.

## The IRR.RADMIN.EXTRACT.PWENV Profile

IRR.RADMIN.EXTRACT.PWENV is a FACILITY class resource which authorizes the retrieval of the enveloped password from RACF using a new extension to the R\_admin callable service (IRRSEQ00). Audit options in the profile can be used to log successes, and thus maintain a history of whose passwords have been retrieved, and by whom. Failures may also be logged. The logstring identifies the user whose password has been retrieved.

## The NOTIFY.LDAP.USER profile

If the NOTIFY.LDAP.USER profile is defined, then an LDAP change log entry will be created when a user's password is enveloped. See the "LDAP notification for user status changes" on page 9 section for more details.

---

## Setting up password enveloping

There are several RACF setup steps which must be performed in order for recipients to be able to retrieve users' password changes. As you are initially configuring and testing this function, check the console for error messages. As RACF detects errors during the password enveloping process, they will be reported to the console, not to the end user who is initiating a password change.

## Preparing the system

Password enveloping uses the System SSL DLLs to implement the creation of the password envelope. The System SLL DLLs must be available to the RACF subsystem address space and must be APF authorized in order for password enveloping to function properly. By default, the System SSL DLLs are located in *hlq*.SGSKLOAD, where *hlq* is the high level qualifier for System SSL as defined by the installation. To make the libraries available to RACF, add the *hlq*.SGSKLOAD either to LINKLST or to the STEPLIB of the RACF subsystem address space PROC. If accessed via the linklist, the linklist must either be authorized (LNKAUTH=LNKLST specified in the IEASYSxx parmlib member) or the libraries must be explicitly APF authorized. If accessed via a STEPLIB, the libraries must be APF authorized and DISP=SHR specified.

## Preparing RACF

- If you have not already updated the RACF database templates during installation of the PTF, run IRRMIN00 with PARM=UPDATE. If using z/OS V1R3 or z/OS V1R4, you must re-IPL the system. If using z/OS V1R5, re-IPLing is not required.
- Add an OMVS segment to the RASP user ID and to its default group (the output of RACF's SET LIST command will identify the subsystem user ID). It is required to establish the LE run-time environment required to use the System SSL functions. This step may not be necessary if you have set up a default UNIX identity (by implementing the BPX.DEFAULT.USER profile in the FACILITY class), but you should not set up the default UNIX identity just for the password

enveloping function. You can specify any UID value you choose. For example, if the RACF subsystem is running under a user ID of RACFSUB, whose default group is STCGRP:

```
ALTUSER RACFSUB OMVS(AUTOUID HOME(/) PROGRAM(/bin/sh))
ALTGROUP STCGRP OMVS(AUTOGID)
```

**Note:** This example takes advantage of the AUTOUID and AUTOGID keywords to automatically assign a unique UID and GID. See the *z/OS RACF Security Administrator's Guide* for instructions on how to enable automatic UID/GID assignment. Alternatively, a unique UID can be explicitly assigned using the UID keyword, and a GID can be explicitly assigned using the GID keyword.

- If the RACF subsystem identity does not have the TRUSTED or PRIVILEGED attribute, it will require the necessary FACILITY class authorization in order to extract certificates from a key ring (certificate setup is described below):

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACFSUB) ACCESS(READ)
```

You may already be protecting this resource, perhaps with a generic profile. Modify this step as appropriate to your environment.

**Recommendation:** Assign the TRUSTED attribute to the RASP user ID.

## Defining a local certificate authority certificate using RACF as the certificate authority

If you will be using RACF as your CA, you will need to create a certificate authority certificate, if you have not already done so. The following command creates a certificate authority (or signer) certificate. This example creates a certificate called **RACFCA** to be used when creating subsequent certificates, such as the certificate which RACF will use during the password enveloping process. This command may be modified, specifically **subjectsdn**, **organization**, and **country**, to reflect the naming structure and conventions used at your installation.

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('RACFCA') O('ibm') C('us'))
WITHLABEL('RACFCA') NOTAFTER(DATE(2020-12-31)) ICSF
```

**Note:** Certificates signed with this local certificate authority certificate show an issuer of cn=RACFCA,o=ibm,c=us when listed with the RACDCERT LIST command.

**Note:** We recommend the use of ICSF to store private keys, and this example reflects that recommendation. If you do not use ICSF, then omit this keyword from the command.

**Note:** RACF will sign the password envelope so that a recipient can verify that the envelope signature was created using RACF's certificate (which is created in the next step). If the recipient also wants to check the veracity of RACF's certificate, it will need this CA certificate to do so. In this case, the CA certificate must be exported to a key ring known to the recipient application and marked as trusted.

## Generating a X.509V3 certificate for RACF's use for user password enveloping

- A digital certificate containing a private key must be generated for RACF's use. This certificate and its associated private key will be used in the RACF password enveloping process. In the following example, RACDCERT commands are used

to generate a certificate for the RACF address space, whose RACF userid is **RACFSUB**. This certificate is signed by RACF, which is the certificate authority in the context of this example. The local CA certificate is identified by the label **RACFCA**.

```
RACDCERT ID(RACFSUB) GENCERT
    SUBJECTSDN(CN('RACF AddrSpace System 1')O('ibm')C('us'))
    WITHLABEL('RASP1')SIGNWITH(CERTAUTH LABEL('RACFCA'))
    KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
    NOTAFTER(DATE(2020-12-31)) ICSF
```

**Note:** The actual user ID for your subsystem is defined by setting up a started task identity using either the started procedures table (ICHRIN03), or by defining a profile in the STARTED class. See the System Programmer's Guide for documentation on setting up the RACF subsystem. If you change the user ID under which the RACF subsystem runs, then you will need to create and populate the key ring for this new identity as described below.

**Note:** The label 'RASP1' is arbitrary; a different string can be used if desired.

**Note:** It is recommended that the use of ICSF to store private keys, and this example reflects that recommendation. If you do not use ICSF, then omit this keyword from the command.

**Note:** After performing the setup, avoid changing RACF's private key. If you do, then RACF will not be able to build PKCS#7 envelopes for existing passwords (since the passwords were encrypted under the old public key, they cannot be decrypted under the new private key). Normal operation will resume as users subsequently change their passwords.

- Create a RACF key ring named **IRR.PWENV.KEYRING**. Note that the name of the key ring is case sensitive.

```
RACDCERT ID(RACFSUB) ADDRING(IRR.PWENV.KEYRING)
```

- Connect RACF's certificate to the key ring as the default certificate.

```
RACDCERT ID(RACFSUB) CONNECT(LABEL('RASP1') RING(IRR.PWENV.KEYRING)
    DEFAULT USAGE(PERSONAL))
```

RACF's certificate must be defined as the default certificate.

Using these examples will create a certificate which is TRUSTED. However, it is always a good idea to verify this by listing the certificate using RACDCERT LIST. This also applies to the certificates which are created in the following steps. If the certificate is not TRUSTED, use the following command to mark it as TRUSTED.

```
RACDCERT ID(RACFSUB) ALTER (LABEL('RASP1')) TRUST
```

At the time a user's password is changed, if the user is eligible for enveloping (see below) then the user's new password is encrypted under the public key of the default certificate only, and stored back in the user's USER profile. When an application requests the envelope, the password is decrypted using the private key of the default certificate, and a PKCS#7 password envelope is created for all of the recipients on the key ring. (Note that any user with READ access to the RACF database, or who can perform a RACROUTE REQUEST=EXTRACT, may be able to obtain the default certificate's private key and thence a user's encrypted password. As always, you should protect the RACF database and its copies against inappropriate access. Further, you should verify that applications retrieving passwords do so using the R\_admin interface).

- During the RACF password enveloping process, RACF encrypts data which can only be recovered by the intended recipient of that data. An intended recipient, such as the identity of the IBM Directory Integrator process which may be

running off of the z/OS platform, is identified by a X.509V3 certificate. Certificates that identify intended recipients of RACF password envelopes must be loaded in to the key ring IRR.PWENV.KEYRING which is associated with the user ID assigned to the RACF subsystem address space. Note that these certificates are only used to encrypt password information; they are not used to bind to LDAP or to authenticate to RACF. Generally speaking, you should only permit trusted application identities (not humans) to recover user passwords.

Certificates for intended recipients may be created by RACF, and exported to off platform processes for instance. The creation of the certificates may be accomplished using the following RACDCERT commands which generate certificates for **IDI1** and **APP2**, who in this example are the identities of processes which will be authorized to retrieve RACF password envelopes. These certificates are signed with the local CA (RACF) certificate that is identified by the label **RACFCA**.

```
RACDCERT ID(IDI1) GENCERT
  SUBJECTSDN(CN('IBM Directory Integrator Server 1') O('ibm')
  C('us')) WITHLABEL('IDI1') SIGNWITH(CERTAUTH LABEL('RACFCA'))
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
```

```
RACDCERT ID(APP2) GENCERT
  SUBJECTSDN(CN('Application Server 2') O('ibm')
  C('us')) WITHLABEL('APP2') SIGNWITH(CERTAUTH LABEL('RACFCA'))
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
```

The KEYUSAGE attributes HANDSHAKE, DATAENCRYPT and DOCSIGN must be specified for the certificates.

You may also create certificates directly on the recipient platform and import them into RACF. Any key management system can be used to create the recipient key pair and certificate, as long as it can export certificates in an industry standard format understood by RACF's RACDCERT command.

The following example uses Keytool, which is shipped with many Java Virtual Machines (JVMs), including the one shipped with IBM Directory Integrator.

You may not need to perform all of the steps below if you already have a key management infrastructure. The examples below assume you are starting new.

Create certificates:

Use of RSA as a key algorithm is required. Keytool may use a default of DSA, which does not work.

```
keytool -genkey -alias IDI1 -keypass xxxxxx -storepass xxxxxx -keystore recipient.jks -dname
  "CN=(IBM Directory Integrator Server 1),O=(ibm),C=(us)" -keyalg RSA
keytool -genkey -alias APP2 -keypass xxxxxx -storepass xxxxxx -keystore recipient.jks -dname
  "CN=(Application Server 2),O=(ibm),C=(us)" -keyalg RSA
```

The commands above will create two self-signed certificates. For example purposes, this is sufficient. In a production environment, you may wish to use something other than a self-signed certificate.

Export certificates:

The rfc keyword specifies base64 encoded output.

```
keytool -export -alias IDI1 -file IDI1.b64 -keystore recipient.jks -storepass xxxxxx -rfc
keytool -export -alias APP2 -file APP2.b64 -keystore recipient.jks -storepass xxxxxx -rfc
```

The following example uses IBM Key Management, running on Windows 2000. The commands are slightly different from those in the previous example, but the procedure is the same.

Create key database:

```
gsk5cmd.exe -keydb -create -db recipient.kdb -pw xxxx
```

Create certificates:

```

gsk5cmd.exe -cert -create -db recipient.kdb -pw xxxx -label IDI1 -dn
"CN=(IBM Directory Integrator Server 1),O=(ibm),C=(us)"
gsk5cmd.exe -cert -create -db recipient.kdb -pw xxxx -label APP2 -dn
"CN=(Application Server 2),O=(ibm),C=(us)"

```

The commands above will create two self-signed certificates. For example purposes, this is sufficient. In a production environment, you may wish to use something other than a self-signed certificate.

Export certificates:

```

gsk5cmd.exe -cert -extract -db recipient.kdb -pw xxxx -label
"IDI1" -target IDI1.b64 -format ascii
gsk5cmd.exe -cert -extract -db recipient.kdb -pw xxxx -label
"APP2" -target APP2.b64 -format ascii

```

At this point, using either example above, the contents of the certificates are contained in the files IDI1.b64 and APP2.b64.

You should copy the certificates to the host system. Because the certificates were exported in base 64, they can be cut and pasted into a host file, using a text editor. If you use FTP, transfer them using the ASCII option, not binary. The files should start with '-----BEGIN CERTIFICATE-----' and end with '-----END CERTIFICATE-----' when viewed on the host. For this example, the files are copied to CERT.IDI1.TEXT and CERT.APP2.TEXT.

Import the certificates in RACF using RACDCERT:

```

RACDCERT ID(IDI1) add(cert.IDI1.TEXT) trust withlabel('IDI1')
RACDCERT ID(APP2) add(cert.APP2.TEXT) trust withlabel('APP2')

```

If you created self-signed certificates, RACF will warn that the Certificate Authority is not defined to RACF, but will properly import the certificates.

- Connect the recipient certificates to the key ring. RACF will encrypt the password for no more than 20 recipient certificates:

```

RACDCERT ID(RACFSUB) CONNECT(ID(IDI1) LABEL('IDI1'))
RING(IRR.PWENV.KEYRING) USAGE(PERSONAL))

RACDCERT ID(RACFSUB) CONNECT(ID(APP2) LABEL('APP2'))
RING(IRR.PWENV.KEYRING) USAGE(PERSONAL))

```

RACF constructs the PKCS#7 password envelope at the time the envelope is requested. So, if you add a certificate for a principal, that principal will be able to decrypt the password for any eligible user whose current password has already been changed (assuming the principal has the authorization which is described in the next step). Likewise, if a certificate is removed from the key ring, the principal will not be able to decrypt any password envelopes, even if the password was changed when the certificate was on the ring, and even if the authorization described in the next step has not been removed for the principal.

You should export RACF's certificate to the recipient key database. This is used to verify the signature of password envelopes as they are being decrypted to ensure they came from RACF. You need to export both the RACF CA certificate and the RACF address space certificate. These were created above.

Start with the CA certificate:

```

racdcert certauth export(label('RACFCA')) DSN(CERT.RACFCA.TEXT) FORMAT(certB64)
racdcert id(RACFSUB) export(label('RASP1')) DSN(cert.RASP.TEXT) FORMAT(certb64)

```

These files must be transferred to the recipient system. Use FTP (ASCII mode) or simple cut and paste to create files racfca.cert and rasp.cert. Then import the files:

Using the keytool command:

```
keytool -import -alias RACFCA -file racfca.cert -keystore recipient.jks -storepass xxxxxx
keytool -import -alias RASP1 -file rasp.cert -keystore recipient.jks -storepass xxxxxx
```

Using gsk5cmd:

```
gsk5cmd.exe -cert -add -db recipient.kdb -pw xxxx -label
"RACFCA" -file racfca.cert -format ascii
gsk5cmd.exe -cert -add -db recipient.kdb -pw xxxx -label
"RASP1" -file rasp.cert -format ascii
```

- Authorize these same principals to the new R\_admin function which retrieves the password envelope from RACF:

```
RDEFINE FACILITY IRR.RADMIN.EXTRACT.PWENV UACC(NONE)
```

```
PERMIT IRR.RADMIN.EXTRACT.PWENV CLASS(FACILITY) ID(IDI1 APP2)
ACCESS(READ)
```

Failed access attempts to this resource will be logged by default. The logstring of the audit record will contain the user ID whose password envelope is being retrieved. Given the sensitive nature of this function, IBM recommends that you log successful accesses as well. From a user with the RACF AUDITOR attribute:

```
RALTER FACILITY IRR.RADMIN.EXTRACT.PWENV AUDIT(ALL(READ))
```

Assuming the FACILITY class is already ACTIVE and RACLISTed, refresh the class.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

## Activating password enveloping

This procedure involves defining resources in the new RACFEVNT class, and activating the class.

- Define the profile which controls password enveloping. This profile is called PASSWORD.ENVELOPE. The APPLDATA of this profile is used to specify the signing hash algorithm used to sign the PCKS#7 password envelope, and the encryption strength used when encrypting the password envelope. For example, to specify the strongest signing and encryption:

```
RDEFINE RACFEVNT PASSWORD.ENVELOPE UACC(NONE) APPLDATA('MD5/STRONG')
```

- Enable password enveloping for users whose passwords are to be encrypted for the intended recipients whose digital certificates were set up above. This is done by permitting a given user or group to PASSWORD.ENVELOPE.

```
PERMIT PASSWORD.ENVELOPE CLASS(RACFEVNT) ID(USER1 USER2 GROUPA GROUPB)
ACCESS(READ)
```

The profile may be given UACC(READ) to activate system-wide password propagation, or you can permit specific users and groups. However, assigning a UACC(READ) to the PASSWORD.ENVELOPE profile will also enable the enveloping of passwords for highly privileged users (i.e. system SPECIAL users, or users which are defined for emergency recovery purposes). This approach is not recommended, unless sensitive user IDs are specifically permitted to this resource with ACCESS(NONE). (This is not necessary for PROTECTED user IDs.) Ultimately of course, this is up to the policy of the individual customer. If a non-global approach is taken, keep in mind that over time, one must consider how newly added users and groups fit into the access scheme, so that users' passwords are properly enveloped or not. In particular, you may want to have intended group connections in place for new users before their initial logons when they must change their passwords. Also, keep in mind that if a user is connected to several groups, his effective authority is the highest allowed by any of his groups (assuming he is not specifically permitted by user ID, in which case this authority overrides that granted by any of his groups). For example, if

list-of-groups processing (SETOPTS GRPLIST) is active, and user BOB is connected to groups GROUP1 and GROUP2, and GROUP1 is permitted with ACCESS(NONE), and GROUP2 is permitted with ACCESS(READ), and BOB is not explicitly permitted, then BOB's effective access to PASSWORD.ENVELOPE is READ, and BOB's password will be enveloped.

**Note:** If you use the UACC(READ) approach, be aware that when you have users with the RESTRICTED attribute that they must be explicitly permitted to the PASSWORD.ENVELOPE profile, by user ID or by group name, if you want their passwords to be enveloped.

- Optionally, an LDAP change log entry can be created whenever a user's new password is enveloped. This will be a required step if you use an application like IBM Directory Integrator to implement a heterogeneous password synchronization solution. See the "LDAP notification for user status changes" on page 9 section for details on LDAP event notification
- Activate the function by activating the RACFEVNT class. It can be RACLISTed to improve performance, but this is not required.  
SETOPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)
- You must stop and restart the RACF subsystem address space after defining the PASSWORD.ENVELOPE profile and activating the RACFEVNT class. If the RACF subsystem address space is already up and running when you configure password enveloping and you do not stop and restart the address space, it will not have the proper environment set up to perform the function, and will fail any requests to envelope passwords.

## Enabling RACF for heterogeneous password synchronization

Before implementing this function, you must carefully weigh the risks vs. the benefits. RACF has always implemented one-way encryption when storing a user's new password. This implementation makes it impossible for even a system administrator to obtain a user's password once that user has changed his initial logon password. This implementation protects the user against unauthorized use of his password, and increases system accountability. Implementing this function will make it possible for an authorized user or application to retrieve a user's current password. Subsequent use of this password will result in a loss of accountability: who actually entered the user ID and password and is now working under the user's identity? An administrator currently has the capability of simply changing the user's password, and then logging on as that user. However, in this case, the user can at least detect the fact that his password has changed.

Looking at a wider view, IBM Directory Integrator will be utilizing this new function in order to implement a heterogeneous password synchronization solution. While password sync is a topic somewhat outside the scope of RACF, you should be aware that some believe password sync to be a security exposure in that it reduces the security of your enterprise to its weakest link. For example, z/OS has traditionally been viewed as a high security platform, in much part due to the security of user passwords. In an heterogeneous environment, a password sync application can open up a z/OS system to a successful hack on any other platform, such as Windows or UNIX.

On the other hand, password sync is often viewed as a boon to usability. These days, a given user may have many accounts on many different systems, and managing the different passwords, which may have different size and content restrictions, as well as different expiration dates, can be a major headache and productivity detractor. And in fact, this complexity can itself lead to a loss of overall

| security in that users may be tempted to write passwords on paper, or use easy to  
| guess passwords, in an attempt to manage the complexity.

| Other solutions currently exist which perform password synchronization in which  
| z/OS is a participating platform. Such applications hook into RACF exits to intercept  
| password changes. The PKCS#7 password enveloping function, in conjunction with  
| LDAP event notification, simply provides a cleaner way for such applications to  
| subscribe to password changes, but does not necessarily in and of itself provide  
| security which is any better or any worse than that already in place by such  
| applications. Ultimately, it is the application and the application vendor which you  
| must trust to maintain the security of RACF passwords from the point in which they  
| have been intercepted (for example, by never sending the clear text password  
| across a network, and by protecting any repository which might contain clear text  
| passwords).

| It is the individual's decision to evaluate password synchronization software, and  
| also the security administrator to enable PKCS#7 password enveloping in support  
| of such software. Part of deploying such software is insuring proper user education  
| and network security. RACF has done its best to minimize the risk associated with  
| this function with several implementation features:

- | • As with all new RACF enhancements, this function is not enabled by default. The  
| security administrator must set it up before any passwords are enveloped and  
| retrievable.
- | • The use of public key cryptography protects passwords as they are sent across  
| the network, and makes them available only to those principals who the security  
| administrator has authorized, by virtue of placing their certificates on a RACF key  
| ring.
- | • The ability to use RACF key rings can keep the trust policy totally within the  
| jurisdiction of the z/OS security officer. Furthermore, the encryption/retrieval of  
| password envelopes is fully dynamic with respect to key ring changes. (The  
| password envelope is created on-demand when it is requested using the current  
| contents of the key ring.) Thus, if the private key of a recipient has become  
| compromised, the removal of the recipient's certificate from the key ring  
| immediately makes password envelopes indecipherable by whoever stole the  
| recipient's private key. Thus, even if the thief is able to masquerade as the  
| intended recipient, and can bind to LDAP and retrieve the password envelope  
| under the recipient's identity, he will still not be able to decipher it using the  
| stolen private key.
- | • Profiles in the RACFEVNT class establish password enveloping policy. This  
| policy allows you to scope password enveloping to a subset of RACF users, thus  
| allowing you to exclude sensitive user IDs at your discretion.
- | • Policy changes are accountable through the creation of SMF records for every  
| privileged operation associated with this function (changes to the key ring,  
| changes to the enveloping policy, and the actual retrieval of the password  
| envelope from RACF).

---

## Chapter 4. Migration considerations

---

### Purpose

See Chapter 1, "Overview," on page 3 for support information.

---

### Process

This support might affect the following areas of RACF<sup>®</sup> processing.

Area	Considerations
Administration	The new RACFEVNT class contains profiles which control the password enveloping and LDAP notification functions.
Application Development	R_admin (IRRSEQ00) callable service has been updated to provide an interface for authorized applications to retrieve an encrypted PKCS#7 envelope containing a user's password.  R_Proxyserv (IRRSPY00) callable service has been updated to provide an interface for authorized applications to create their own LDAP change log entries for updates made using interfaces other than the RACF commands.
Auditing	None.
Customization	None.
General User	None.
Operations	During RACF subsystem initialization, RACF will detect if the password enveloping function is configured and if so, will invoke UNIX services to initialize itself as a UNIX process. Thus, the OMVS kernel must be initialized, and if it is not, RACF subsystem initialization will wait for OMVS initialization to complete. As a result, the RACF subsystem address space may initialize later in the IPL sequence than it otherwise would have if password enveloping were not configured.  In addition, if password enveloping is configured, the RACF subsystem can now be affected by an OMVS shutdown. A password enveloping operation will wait for OMVS to be restarted. If enough password changes are made while the OMVS kernel is unavailable, the available tasks in the RACF subsystem will be exhausted, affecting other RACF address space functions which would otherwise not have been affected by an OMVS shutdown. However, OMVS shutdowns should not be performed while work is occurring on the system. See <i>z/OS UNIX System Services Planning</i> for details on shutting down OMVS.  The SET command has been updated to allow tracing of the System SSL functions used by the password enveloping function.
Interfaces	Refer to Chapter 9, "Command considerations," on page 45, Chapter 5, "Data area considerations," on page 23, Chapter 8, "Macros and interfaces considerations," on page 43, and Chapter 7, "Messages considerations," on page 39.

---

### Other

The USER profile for each user eligible for password enveloping, will have approximately 280 bytes of space increase.



## Chapter 5. Data area considerations

The following supplements the COMP section for z/OS RACF data areas:

Table 2.

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	PRXY	
0	(0)	ADDRESS	4	PRXY_PARM_ALET@	Address of ALET for remaining parameters
4	(4)	ADDRESS	4	PRXY_FUNC@	Address of 2 byte function code. Constants for the function codes are supplied below
8	(8)	ADDRESS	4	PRXY_LDAP_HOST@	Address of an area containing a 4 byte length followed by an EBCDIC URL for the LDAP BIND
12	(C)	ADDRESS	4	PRXY_BIND_DN@	Address of an area containing a 4 byte length followed by an EBCDIC DN for the LDAP BIND
16	(10)	ADDRESS	4	PRXY_BIND_PW@	Address of an area containing a 4 byte length followed by an EBCDIC password for the LDAP BIND
20	(14)	ADDRESS	4	PRXY_USERID@	Address of 9 byte area containing a 1 byte length followed by up to 8 EBCDIC characters for a host user ID
24	(18)	ADDRESS	4	PRXY_USERDN@	Address of an area containing a 4 byte length followed by an EBCDIC string naming the base DN of an LDAP subtree
28	(1C)	ADDRESS	4	PRXY_RESULTS@	Address of a pointer to the results
		1... ....		PRXY_LAST_PARM	Variable length parameter list. This is the last parameter
32	(20)	ADDRESS	4	PRXY_FUNC_VERSION@	Address of a 4 byte version number for the function specific parameter list

Table 2. (continued)

Offsets					
Dec	Hex	Type	Len	Name (Dim)	Description
36	(24)	ADDRESS	4	PRXY_FUNC_PARMLIST@	Address of the function specific parameter list
40	(28)	ADDRESS	4	PRXY_LDAP_ERROR@	Address of an area in which to return an LDAP error message
		1... ..		PRXY_LAST_PARM2	Last parameter for second version of the variable length parameter list

**Note:** The following table includes new callable service function code constants for R\_Proxyserv and R\_admin:

Table 3. Callable service new function codes

Len	Type	Value	Name	Description
1	Decimal	24	ADMN_XTR_PWENV	Extract PKCS#7 encrypted password envelope
2	Decimal	3	PRXY_CHANGELOG	Create LDAP change log entry

---

## Chapter 6. Callable services considerations

---

### R\_admin (IRRSEQ00): RACF Administration API

#### Function

The **R\_admin** service enables applications to construct a function code/data field name parameter list to manage RACF profiles within the RACF database. Alternately, a RACF TSO administrative command image can be passed to manage RACF user profile information. Output, if any, which resulted from RACF's processing of the profile admin request is returned to the caller in virtual storage. This callable service does **not** support all RACF command functions. For a list of the commands that are **not** executed through this service, see the Usage Notes section.

The exact format (spacing and order) of the data in the command output or messages does not form a programming interface. No programs should depend on the exact format of this data.

#### Requirements

<b>Authorization:</b>	Any PSW key in supervisor or problem state
<b>Dispatchable unit mode:</b>	Any task
<b>Cross memory mode:</b>	PASN = HASN = SASN
<b>AMODE:</b>	31
<b>RMODE:</b>	Any
<b>ASC mode:</b>	Primary only
<b>Recovery mode:</b>	Recovery must be provided by caller
<b>Serialization:</b>	Enabled for interrupts
<b>Locks:</b>	No locks held
<b>Control parameters:</b>	The parameter list and the work area must be in the primary address space.

#### RACF authorization

For the function codes which cause a RACF command to execute in the RACF address space, the command will run under the authority of a RACF user ID. The RACF user ID can come from one of a number of sources, and is searched for in the following order:

- The RACF\_userID parameter
- The ACEE\_ptr parameter
- The user ID associated with the current task control block (TCB)
- The user ID associated with the current address space (ASXB)

For problem state callers only, READ authority to the resource IRR.RADMIN.(*command-name*) in the FACILITY class is required to execute the RACF TSO command *command-name* using the ADMN\_RUN\_COMD function code of R\_Admin. This is in addition to any authority checks done by the command itself. The resource must be defined using the full command name even if the abbreviated version of the command name is used with R\_Admin. (e.g., lu joeuser would require READ authority to IRR.RADMIN.LISTUSER.) Generic profiles can be used.

READ authority to the resource IRR.RADMIN.EXTRACT.PWENV in the FACILITY class is required to extract the enveloped RACF password, using the ADMN\_XTR\_PWENV. If this access check is being audited, the logstring will identify the user ID whose password envelope was extracted.

## Format

```
CALL IRRSEQ00 (Work_area,  
              ALET, SAF_return_code,  
              ALET, RACF_return_code,  
              ALET, RACF_reason_code,  
              Function_code,  
              Parm_list,  
              RACF_userID,  
              ACEE_ptr,  
              Out_message_subpool,  
              Out_message_strings  
              )
```

## Parameters

### **Work\_area**

The name of a 1024-byte work area for SAF and RACF usage. The work area must be in the primary address space.

### **ALET**

The name of a word containing the ALET for the following parameter. Each parameter must have an ALET specified. Each ALET can be different. The word containing the ALETs must be in the primary address space.

### **SAF\_return\_code**

The name of a fullword in which the SAF router returns the SAF return code. These return codes are found in Table 8 on page 29.

### **RACF\_return\_code**

The name of a fullword in which the service routine stores the return code. These return codes are found in Table 8 on page 29.

### **RACF\_reason\_code**

The name of a fullword in which the service routine stores the reason code. These reason codes are found in Table 8 on page 29.

### **Function\_code**

The name of a one-byte field that specifies the function (administration request) that RACF is to perform. The function code may have one of the values found in Table 4 on page 27.

### **Parm\_list**

The name of a variable marking the start of the input parameter list. The mapping macro IRRPCOMP contains a definition of the parameter list for each of the values of function\_code. To find parameter list mappings for the values of function\_code, see Table 5 on page 28.

### **RACF\_userID**

The name of a 9 byte area that consists of a 1 byte length field followed by the userID, which can be up to eight characters. If not specified, the length must equal zero. Otherwise, the user ID must be specified in upper case. If specified, the RACF command executed through R\_admin will run under the authority of this userID.

### ACEE\_ptr

The name of a fullword containing the address of the ACEE of the user under whose identity the RACF administrative request runs. The user ID is extracted from the ACEEUSER field. The ACEE itself is not used for subsequent authority checking for the request. If the caller does not specify an ACEE, this area must contain binary zeros. If both an ACEE and a user ID are passed into this service, the user ID is used.

### Out\_message\_subpool

The name of a one-byte field that specifies the subpool used to obtain storage for output messages that are returned. Problem state callers are limited to subpools 1 thru 127.

### Out\_message\_strings

The name of a fullword in which the service routine stores the address of output data, if applicable. It is the responsibility of the caller to free the output storage.

For SETROPTS functions (ADMN\_XTR\_SETR and ADMN\_UNL\_SETR), see *z/OS Security Server RACF Callable Services* for the output mapping.

For ADMN\_XTR\_PWENV (extract encrypted password envelope), see *Output message block* section in this document for the output mapping.

For the remaining functions, see *z/OS Security Server RACF Callable Services* for the mapping of the output message block returned by this service. For the format of each message entry, see *z/OS Security Server RACF Callable Services*. The message entry is a repeating data structure. In it, the entry length and message text repeats if more than one message is present. See also the IRRPCOMP data area in *z/OS Security Server RACF Data Areas*.

## Function code values

Table 4 shows the function code values in the mapping macro IRRPCOMP.

Table 4. Function Code Values in Mapping Macro IRRPCOMP

Function Code	Value	Description
ADMN_ADD_USER	X'01'	Add a user to the RACF database
ADMN_DEL_USER	X'02'	Delete a user from the RACF database
ADMN_ALT_USER	X'03'	Alter a user's RACF database profile
ADMN_LST_USER	X'04'	List the contents of a user's RACF database profile
ADMN_RUN_COMD	X'05'	Run a RACF command image
ADMN_ADD_GROUP	X'06'	Add a group to the RACF database
ADMN_DEL_GROUP	X'07'	Delete a group from the RACF database
ADMN_ALT_GROUP	X'08'	Alter a group's RACF database profile
ADMN_LST_GROUP	X'09'	List a group's RACF database profile
ADMN_CONNECT	X'0A'	Connect a single user to a RACF group
ADMN_REMOVE	X'0B'	Remove a single user from a RACF group
ADMN_ADD_GENRES	X'0C'	Add a general resource profile to the RACF database
ADMN_DEL_GENRES	X'0D'	Delete a general resource profile from the RACF database
ADMN_ALT_GENRES	X'0E'	Alter a general resource's RACF database profile
ADMN_LST_GENRES	X'0F'	List a general resource's RACF database profile
ADMN_ADD_DS	X'10'	Add a data set profile to the RACF database

Table 4. Function Code Values in Mapping Macro IRRPCOMP (continued)

Function Code	Value	Description
ADMN_DEL_DS	X'11'	Delete a data set profile from the RACF database
ADMN_ALT_DS	X'12'	Alter a data set's RACF database profile
ADMN_LST_DS	X'13'	List a data set's RACF database profile
ADMN_PERMIT	X'14'	Permit a user or group to a RACF profile
ADMN_ALT_SETR	X'15'	Alter SETROPTS information
ADMN_XTR_SETR	X'16'	Extract SETROPTS information in R_admin format
ADMN_UNL_SETR	X'17'	Extract SETROPTS information in SMF data unload format
ADMN_XTR_PWENV	X'18'	Extract PKCS#7 encrypted password envelope

### Parameter list mappings by function code

Table 5 shows where to find the parameter list mappings for the values of function\_code.

Table 5. Parameter List Mappings for Function\_Code Values

For function_code(s)	Refer to
ADMN_ADD_USER, ADMN_DEL_USER, ADMN_ALT_USER, ADMN_LST_USER, ADMN_XTR_PWENV	User administration chapter in z/OS Security Server RACF Callable Services
ADMN_RUN_COMD	Running RACF commands chapter in z/OS Security Server RACF Callable Services
ADMN_ADD_GROUP, ADMN_DEL_GROUP, ADMN_ALT_GROUP, ADMN_LST_GROUP	Group administration chapter in z/OS Security Server RACF Callable Services
ADMN_CONNECT ADMN_REMOVE	Group connection administration chapter in z/OS Security Server RACF Callable Services
ADMN_ADD_GENRES, ADMN_DEL_GENRES, ADMN_ALT_GENRES, ADMN_LST_GENRES, ADMN_ADD_DS, ADMN_DEL_DS, ADMN_ALT_DS, ADMN_LST_DS, ADMN_PERMIT	Resource profile administration chapter in z/OS Security Server RACF Callable Services
ADMN_ALT_SETR	Parameter list format for SETROPTS administration chapter in z/OS Security Server RACF Callable Services
ADMN_XTR_SETR	Input parameter list is ignored
ADMN_UNL_SETR	Input parameter list is ignored

### Output message block mapping

The following mapping applies to command (for example, LISTUSER) output and informational and error messages. For 'extract' or 'unload' types of requests, the output blocks will be formatted as documented for those function codes.

Table 6. Mapping of Output Message Block

Offset	Length	Description
0	4	Next ADMIN output messages block, or zero if no additional blocks follow

Table 6. Mapping of Output Message Block (continued)

Offset	Length	Description
4	4	Eye catcher to aid in virtual storage dumps 'RMSG'
8	1	Storage subpool in which the block was obtained
9	3	Total block length
12	4	Offset to the first byte after the last message; This offset value is related to the first message.
16	1	Start of the first message

Table 7. Format of Each Message Entry

Offset	Length	Description
0	2	Length of this message text entry.
2	*	Variable message text.

The actual format of the output area is mapped by macro IRRPCOMP.

See *z/OS Security Server RACF Data Areas* for the actual format of the output area.

## Return and reason codes

IRRSEQ00 returns the following values in the reason and return code parameters:

Table 8. Return and Reason Codes

SAF Return Code	RACF Return Code	RACF Reason Code	Explanation
0	0	0	The service was successful. Output from the RACF command may be present. The Out_message_strings parameter should be interrogated by caller.
4	0	0	RACF is not installed.
4	4	0	For ADMN_XTR_PWENV, the user does not have a password envelope.
4	4	4	For ADMN_XTR_PWENV, the target user does not exist.
4	4	8	For ADMN_XTR_PWENV, a problem was encountered while retrieving the password envelope.
8	8	0	Incorrect function code
8	8	4	Input parameter list error

...

**Note:** Return and reason codes are shown in decimal.

## Usage notes

...

## Parameter list formats

The following information describes parameter list formats for running RACF commands and for all administration.

### Running RACF commands

For the ADMN\_RUN\_COMD (execute a RACF command) function code, the mapping associated with the function-specific parameter list is mapped as follows:

Table 9. Parameter List Format for Running a Command

Offset	Length	Description
0	2	Length of the RACF command string. Note, the length must not exceed 4096 characters.
2	*	Syntactically correct RACF TSO administration command string.

### User administration

For the ADMN\_ADD\_USER, ADMN\_DEL\_USR, ADMN\_ALT\_USER, ADMN\_LST\_USER, and ADMN\_XTR\_PWENV function codes, the mapping associated with the function specific parameter list is mapped as in Table 10.

Table 10. Parameter List Format for User Administration

Offset	Length	Description
0	1	Length of the user ID
1	8	Uppercase RACF user ID
9	1	Reserved
10	2	Output offset to the segment or field entry in error in relationship to the start of the Parm_list. Only applies to ADMN_ADD_USER and ADMN_ALT_USER requests when an "invalid request" error is returned to the caller.
12	2	Number of RACF profile segments
14	*	Start of first segment entry

**Note:** For ADMN\_DEL\_USER and ADMN\_XTR\_PWENV, no segment data is expected. The number of segments should be zero. If non-zero, any segment data present is ignored.

Each segment entry is mapped as in Table 11.

Table 11. Segment Entry Mapping

Offset	Length	Description
0	8	Profile segment name (left-justified, uppercase, and padded with blanks). For ADMN_LST_USER, any user profile segment as defined in the RACF database templates is accepted. For ADMN_ADD_USER and ADMN_ALT_USER, the following segments are supported: BASE, CICS, DCE, DFP, EIM, KERB, LANGUAGE, LNOTES, NDS, NETVIEW, OMVS, OPERPARM, OVM, PROXY, TSO, and WORKATTR.
8	1	Flag byte. Use Y to create or alter the segment. Use N to delete the segment.

Table 11. Segment Entry Mapping (continued)

Offset	Length	Description
9	2	Number of fields to update within a segment
11	*	First field for segment

...

**Output message block:**

Following is the mapping of the output message block returned by R\_admin for the ADMN\_XTR\_PWENV function code. The output storage is obtained in the subpool specified by the caller in the Out\_message\_subpool parameter.

Table 12. Output message block

Offset	Length	Description
0	4	Eye catcher to aid in virtual storage dumps: 'RXPW'
4	1	Subpool of this block
5	3	Total length of output buffer
8	0	Start of encrypted password envelope

**Contents of the encrypted password envelope:** The PKCS#7 standard defines the structure of encrypted content comprising a digital envelope which is intended for multiple recipients. RACF will first sign, and then envelope (encrypt) the password payload (defined below). The recipient will decrypt the envelope to obtain the payload. The recipient should also verify the signature of the envelope, although this is not necessary in order to obtain the payload.

For the structure of the password envelope, refer to the PKCS#7 standard, Version 1.5, which can be obtained at <http://www.rsasecurity.com/rsalabs/pkcs/index.html>.

PKCS#7 defines several data types using ASN.1 notation to describe them. Each type is contained in a ContentInfo type. ContentInfo simply identifies the contained data type with an object identifier (OID), followed by the actual data. See section 7 of the PKCS#7 standard.

On the outside of the structure is a ContentInfo containing the EnvelopedData content type. The EnvelopedData type is described in section 10 of the PKCS#7 standard. The EnvelopedData type is further broken down into subtypes such as RecipientInfos and EncryptedContentInfo. EncryptedContentInfo is broken down further into ContentType, ContentEncryptionAlgorithmIdentifier, and EncryptedContent. ContentType will be SignedData, and EncryptedContent, defined in the standard as OCTET STRING (which just means an arbitrary string of data), will be the ContentInfo containing data of type SignedData which is the output of the System SSL signing function. See section 9.1 of the PKCS#7 standard describing the SignedData type.

SignedData contains a field called contentInfo, which is the data being signed. This data can be of any of the types defined by the standard. The type we use is Data, so the contentInfo field within SignedData contains the ContentInfo for the Data type. The Data type is defined as OCTET STRING. This OCTET STRING is the payload that RACF is constructing as input to the whole envelope-creation process.

The payload contains password related information in BER-encoded ASCII format. Following is the ASN.1 notation describing the payload as constructed by RACF:

```
PasswordPayload ::= SEQUENCE {
    Version      INTEGER
    Expired      BOOLEAN
    Password     UTF8String
    Changetime   IA5String
    Language     IA5String OPTIONAL DEFAULT "ENU"
}
```

**Version** is the version number of the password payload. It will be set to 1.

**Expired** will be true if the new password is marked as expired at the time of the password change (for example, an ALTUSER command is used to change the password without specifying the NOEXPIRED operand). If Expired is true, the password must be changed the next time the user logs on.

**Password** is the value of the new password, in lower case.

**Changetime** is a character string of decimal numbers in the format *yyyymmddhhiss.uuuuuuZ* (relative to GMT) where

- *yyyy* is year
- *mm* is month
- *dd* is day
- *hh* is hour
- *ii* is minutes
- *ss* is seconds
- *uuuuuu* is micro seconds
- 'Z' is a character constant meaning that this time is based on ZULU time, also known as GMT

**Language** is the 3 character language code which RACF has used in order to determine the UTF-8 code points for the variant characters. This is for diagnostic purposes. Currently, RACF assumes the language is U.S. English ('ENU'). This may result in RACF propagating a different password than may be expected by a given user using a given keyboard and code page. If so, users should avoid using variant characters in passwords when RACF is participating in a password synchronization network. For example, a person in the United Kingdom may enter the pound sterling symbol as a character in a new password. This is represented as X'5B' which RACF will accept. When RACF envelopes this password assuming U.S. English, the UTF-8 code point for '\$' will be used. If this password is propagated to another system, and the person tries to logon to that system using the same keystrokes he used to change his password in RACF, the password will be rejected.

...

---

## R\_proxyserv (IRRSPY00): LDAP interface

### Function

The **R\_proxyserv** SAF callable service invokes the LDAP component of the Security Server for z/OS to obtain data which resides in an LDAP directory. Invokers are not required to be LE-enabled.

## Requirements

<b>Authorization:</b>	Any PSW key in supervisor or problem state
<b>Dispatchable unit mode:</b>	Task of user
<b>Cross memory mode:</b>	PASN = HASN
<b>AMODE:</b>	31
<b>RMODE:</b>	Any
<b>ASC mode:</b>	Primary or AR mode
<b>Recovery mode:</b>	ESTAE. Caller cannot have a FRR active
<b>Serialization:</b>	Enabled for interrupts
<b>Locks:</b>	No locks held
<b>Control parameters:</b>	The parameter list and the work area must be in the primary address space. The words containing the ALETs must be in the primary address space.

## Linkage Conventions

The parameter list for this callable service is intended to be variable length to allow for future expansion. To allow for this, the last word in the parameter list must have a 1 in the high-order (sign) bit.

## RACF Authorization

For callers not running in system key or supervisor state, the use of R\_proxyserv is authorized by the resource IRR.RPROXYSERV in the FACILITY class. The application server must be running with a RACF user or group ID that has at least READ authority to this resource. If the class is inactive, or the resource is not defined, only servers running with a system key or in supervisor state may use the R\_proxyserv service.

**Note:** If using function code 3, the caller must be in system key or supervisor state.

## Format

```
|          CALL IRRSPY00 (Work_area,  
|                          ALET, SAF_return_code,  
|                          ALET, RACF_return_code,  
|                          ALET, RACF_reason_code,  
|                          ParmALET,  
|                          Function_code,  
|                          LDAP_host,  
|                          Bind_DN,  
|                          Bind_PW,  
|                          Host_userID,  
|                          Base_DN,  
|                          Result_entries  
|                          Function_parmlist_version,  
|                          Function_parmlist,  
|                          LDAP_error_string  
|          )
```

## Parameters

### Work\_area

The name of a 1024-byte work area for SAF. The work area must be in the primary address space.

**ALET**

The name of a fullword which must be in the primary address space and contains the ALET for the remaining parameters.

**SAF\_return\_code**

The name of a fullword in which the SAF router returns the SAF return code.

**RACF\_return\_code**

The name of a fullword in which the service routine stores the return code.

**RACF\_reason\_code**

The name of a fullword in which the service routine stores the reason code.

**ParmALET**

The name of a fullword containing the ALET for the following parameter. Each parameter must have an ALET specified. Each ALET can be different. The words containing the ALETs must be in the primary address space.

**Function\_code**

The name of a half word (2 byte) area containing the Function code. The function code has one of the following values:

X'0001'	Return the distinguished name (DN) of the user identified by the input host user ID.
X'0002'	Return the Policy Director Authorization Services attributes for the specified base DN.
X'0003'	Create an LDAP change log entry for a RACF user profile update

**LDAP\_host**

This is optional, see Usage Notes. The name of an area that consists of a 4 byte length field followed by the string of EBCDIC characters which identify the URL of the LDAP server which the z/OS LDAP Server is to contact when acting as a proxy for this request. The maximum length of this string is 1023 bytes. Uppercase and lowercase characters are allowed, but no significance is attached to the case. See *z/OS Security Server LDAP Server Administration and Use* for more information about LDAP URLs and the *z/OS Security Server RACF Command Language Reference* for more information about how to define this information in the RACF database.

**Bind\_DN**

This is optional, see Usage Notes. The name of an area that consists of a 4 byte length field followed by the string of EBCDIC characters which represent the distinguished name (DN) that the z/OS LDAP Server is to use when acting as a proxy for this request. The maximum length of this string is 1023 bytes. Both uppercase and lowercase characters are allowed. See *z/OS Security Server LDAP Server Administration and Use* for more information about LDAP distinguished names and the *z/OS Security Server RACF Command Language Reference*, for more information about how to define this information in the RACF database.

**Bind\_PW**

This is optional, see Usage Notes. The name of an area that consists of a 4 byte length field followed by the string of EBCDIC characters which represent the password that the z/OS LDAP Server is to use when acting as a proxy for this request. The maximum length of this string is 128 bytes. Both uppercase and lowercase characters are allowed. See *z/OS Security Server LDAP Server Administration and Use* for more information about LDAP passwords and the ,

*z/OS Security Server RACF Command Language Reference* for more information about how to define this information in the RACF database.

**Host\_userID**

The name of a 9 byte area that consists of a 1 byte length field followed by up to 8 EBCDIC characters. Uppercase and lowercase characters are allowed, but no significance is attached to the case. If not specified, the length must equal 0.

**Base\_DN**

The name of an area that consists of a 4 byte length field followed by the string of EBCDIC characters which represent the base DN of an LDAP subtree. The maximum length of this string is 1023 bytes.

**Result\_entries**

The name of a 4 byte area which contains the address of the output area from this service, if any. All character fields in the output area are represented in EBCDIC. The caller is responsible for releasing output area storage, via FREEMAIN or STORAGE RELEASE macro invocation. Offset values in the output area are all relative to the beginning of the output area. See *z/OS Security Server RACF Callable Services* for the *Result\_entries output area* table.

...

**Function\_parmlist\_version**

The name of a 4 byte input value which contains the version number for the Function\_parmlist input field. The contents of this field must be set to binary zero.

**Function\_parmlist**

The name of an area containing data specific to a given value of Function\_code. Not every function code will require a function specific parameter list. If there is no parameter list required for a given function code, then this parameter is ignored. The specific mappings are provided in the IRRPCOMP macro.

The function specific parameter lists are formatted as listed below. All parameters are required unless otherwise noted.

Table 13. Parameter list for function code 3

Offset	Length	Type	Name	Description
0	0	Structure	PRXY_F3_PLIST	Function specific plist for function 3
0	1	Unsigned	PRXY_F3_OPTYPE	Operation type: X'00' - Add X'01' - Delete X'02' - Modify
1	1	Bitstring	PRXY_F3_FLAGS	Request flags
		1... ....	PRXY_F3_PWUPD	Reserved for use by the security product. Not for application use. This bit should be set to zero by applications using this interface.
		.111 1111		Reserved for future use. These bits must be set to 0.

Table 13. Parameter list for function code 3 (continued)

Offset	Length	Type	Name	Description
2	8	Character	PRXY_F3_CLASS	RACF class name, padded to the right with blanks. Currently, the only class name supported is USER.
10	2	Unsigned	PRXY_F3_PROFLEN	Length of profile being changed. Must adhere to length requirements for the class name in PRXY_F3_CLASS.
12	4	Address	PRXY_F3_PROFNAME@	Address of profile name being added, altered, or deleted.
16	8	Character	PRXY_F3_INITIATOR	The user ID who initiated the RACF profile change. If this field contains binary zeros, then R_proxyserv will use the identity of the caller.
24	22	Character	PRXY_F3_DATETIME	The GMT time of the update in the format yyymmddhhii.ss.uuuuuuZ where yyyy is the year mm is the month dd is the day hh is the hours ii is the minutes ss is the seconds uuuuuu is the microseconds 'Z' is constant  If this field contains binary zeroes, R_Proxyserv will use the current date and time.

#### LDAP\_error\_string

The name of an area that consists of a 4 byte length field followed by 256 characters. This field is filled in when LDAP returns an error message. The length will be updated with the actual length of the LDAP error string or zero. This field is used for function code 3.

## Return and Reason Codes

IRRSPY00 returns the following values in the reason and return code parameters:

SAF Return Code	RACF Return Code	RACF Reason Code	Explanation
...			
8	8	32	Function not supported for problem state caller.

...

## Parameter Usage

Function	Function Code	LDAP_host	Bind_DN	Bind_PW	Host_user_ID	Base_DN	Result_entries	Function_parm_list_version	Function_parm_list	LDAP_error_string
Return the base DN for the specified host UID	X'0001'	In <sup>1</sup>	In <sup>1</sup>	In <sup>1</sup>	In	N/A	Out	N/A	N/A	N/A
Return the Policy Director attributes for the specified base DN	X'0002'	In <sup>1</sup>	In <sup>1</sup>	In <sup>1</sup>	N/A	In	Out	N/A	N/A	N/A
Create an LDAP change log entry	X'0003'	N/A	N/A	N/A	N/A	N/A	N/A	In	In	Out

## Usage Notes

1. This service is intended for use by z/OS application servers, which are not executing in a Language Environment (LE). It allows z/OS application servers to perform limited LDAP queries which retrieve information from a directory information tree (DIT). Note, however, that LE-enabled applications may also exploit this service, should they choose to do so.
2. The R\_proxyserv service requires an instance of the z/OS LDAP Server or OS/390 LDAP Server on each physical z/OS or OS/390 instance (whether in a sysplex datasharing configuration or not) and each of these LDAP Server instances must be configured to support PC call and the extended operations backend. See the *z/OS Security Server LDAP Server Administration and Use* for information about configuring this support
3. The parameter list for this callable service is intended to be variable length to allow for future expansion. To allow for this, the last word in the parameter list must have a 1 in the high-order (sign) bit. If the last word in the parameter list does not have a 1 in the high-order (sign) bit, the caller receives a parameter list error. For function codes 1 and 2, the first parameter that can have the high-order bit on, ending the parameter list, is the *Result\_entries* parameter. For function code 3, the first parameter that can have the high-order bit on, ending the parameter list, is the **LDAP\_error\_string** parameter.
4. ...

## Related Services

None



## Chapter 7. Messages considerations

**IRR417I UNABLE TO COMMUNICATE WITH THE RACF SUBSYSTEM. IEFSSREQ RETURN CODE IS** *return-code*.

**Explanation:** A RACF function requiring the RACF subsystem was unable to communicate with the subsystem. Report this message to your system programmer. The possible functions, and the effect of the failure are as follows:

1.

**RRSF password synchronization**

RACF attempted to process a password change request for a user. The password change has been made on this system's RACF database. However, if associations exist with another user on this or a remotely connected system, the passwords will not be synchronized.

2.

**PKCS#7 password enveloping**

RACF attempted to process a password change request for a user. The password change has been made on this system's RACF database. However, a PKCS#7 password envelope was not created and stored in the user's profile for this password change. An existing envelope may exist for the user's previous password.

3.

**LDAP event notification**

A user profile was changed, and a change log entry was supposed to have been created in LDAP. However, the change log entry was not created. Any applications which rely on the LDAP change log for RACF event notifications will not be aware of the change which was made.

**System Action:** The system continues processing.

**System Programmer Response:** The return code indicated in this message reflects the return code from the MVS IEFSSREQ subsystem interface. The return code may be one of these values:

Code	Explanation
4	The subsystem does not support this function.
8	The subsystem exists, but is not active.
12	The subsystem is not defined in the IEFSSNxx parmlib member.
16	The function has not completed. This is a disastrous error.
20	The SSOB or SSIB have invalid lengths or formats.
24	The SSI has not been initialized.

A return code of 4, 16, 20 or 24 indicates a RACF code problem. Report this message to the IBM support center.

A return code of 8 or 12 indicates an installation or RACF subsystem configuration problem. See *z/OS Security Server RACF Migration* for installation considerations and *z/OS Security Server RACF System Programmer's Guide* for configuration considerations for the RACF subsystem.

**Destination:** Descriptor code is 3. Routing codes are 9 and 11.

**IRRC130I SYSTEM SSL FUNCTION x RETURNED ERROR CODE nnn DURING OPERATION NUMBER opcode WHILE PROCESSING THE PASSWORD ENVELOPE FOR USER name.**

**Explanation:** An unexpected error was detected when using System SSL functions to create a PKCS#7 password envelope containing the new password for user *name*.

**System Action:** The system continues processing.

**System Programmer Response:** Use the following table to identify the problem:

Table 14.

x	System SSL function	nnn	Possible cause
'02'X	gsk_open_keyring	'03353009'X	IRR.PWENV.KEYRING not defined, or specified in incorrect case, or not owned by the RACF subsystem user ID
		'03353017'X	The RACF subsystem does not have the trusted or privileged attribute, and does not have at least READ authority to IRR.DIGTCERT.LISTRING in the FACILITY class
'04'X	gsk_get_default_key	'0335300E'X	The certificate for RACF was not added to the key ring as the DEFAULT certificate

Table 14. (continued)

x	System SSL function	nnn	Possible cause
'40'X	gsk_make_enveloped_data_content	'03353033'X	No recipient certificates have been added to the key ring, or the certificates do not have TRUST status
		'03353026'X	A certificate was created without the KEYUSAGE value of HANDSHAKE.

See if the source of the problem can be determined by reading the documentation for the failing API and return code in *System SSL Programming*. If the problem continues, contact the system support center. If more diagnostic data is required, enable System SSL tracing by issuing the subsystem SET TRACE(SYSTEMSSL) command, then have the user attempt to change the password again. System SSL trace records will be created in a z/OS UNIX file named /tmp/gskssl.racf.pid.trc, where *pid* is the process identifier of the RACF task that invoked System SSL. Look for the trace record corresponding to the failing API. See the *z/OS Security Server RACF Command Language Reference* for details on the SET command. See *System SSL Programming* for information on the System SSL APIs and on collecting trace records.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC131I RACF ENCOUNTERED AN R\_PROXYSEV ERROR WHILE ATTEMPTING TO CREATE AN LDAP CHANGE LOG ENTRY FOR AN UPDATE TO USER *name*. SAF RETURN CODE=SAF-return-code, RACF RETURN CODE=return-code, RACF REASON CODE=reason-code.**

**Explanation:** RACF attempted to create an LDAP change log entry for an update to user *name*. The R\_ProxyseV callable service (IRRSPY00) is used to communicate with LDAP. The service failed with the return codes shown. The LDAP change log entry was not created.

**System Action:** The system continues processing.

**System Programmer Response:** Look up the return codes in *z/OS Security Server RACF Callable Services* and correct the problem.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC132I RACF ENCOUNTERED AN UNEXPECTED PARSE RETURN CODE *nn* WHILE PROCESSING AN IRRLOG00 COMMAND.**

**Explanation:** While processing an update, RACF encountered return code *nn* from the IKJPARS service. This can happen when the RACFEVNT class is active, and RACF was either creating a PKCS#7 password envelope for a user, or was attempting to create an LDAP change log entry describing the update to a user. The IRRLOG00 command is created and sent to the RACF subsystem by the RACF database manager. The parse operation failed for this command. The user being changed cannot be identified because the user ID is contained within the IRRLOG00 command.

**System Action:** The user profile was updated successfully on the RACF database, but the enveloping or change log operation did not occur. The system continues processing.

**System Programmer Response:** Contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC133I RACF ENCOUNTERED INCORRECT APPLDATA SYNTAX IN THE PASSWORD.ENVELOPE PROFILE WHILE PROCESSING USER *name* DEFAULT VALUES ARE USED.**

**Explanation:** While processing a password update for user *name*, an error was encountered while interpreting the APPLDATA string in the PASSWORD.ENVELOPE profile in the RACFEVNT class. The APPLDATA is used to specify the signing hash algorithm and encryption strength to use when building a PKCS#7 password envelope for a user.

**System Action:** RACF uses the default values of MD5 for the signing hash algorithm and triple DES for encryption.

**Security Administrator Response:**

Correct the APPLDATA. See the *z/OS Security Server RACF Security Administrator's Guide* for details on defining the PASSWORD.ENVELOPE profile.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC134I RACF ENCOUNTERED AN ICHEINTY ERROR WHILE ATTEMPTING TO PROCESS THE PASSWORD ENVELOPE FOR USER *name*. OPERATION=optype, RETURN CODE=return-code, REASON CODE=reason-code.**

**Explanation:** RACF attempted an ICHEINTY *optype* ('DELETE', 'STORE', or 'EXTRACT') on the password envelope for user *name*, but an unexpected error

occurred. The contents of the password envelope are not current with respect to the user's LOGON password.

**System Action:** The system continues processing.

**System Programmer Response:** Make sure the RACF database templates are current. If this is not the problem, contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC135I RACF ENCOUNTERED AN EXTRACT ERROR FOR PROFILE *profile-name* IN CLASS *classname* WHILE PROCESSING USER *name*. RETURN CODE=*return-code*, RACF RETURN CODE=*racf-return-code*, RACF REASON CODE=*racf-reason-code*.**

**Explanation:** A RACROUTE REQUEST=EXTRACT was attempted but an unexpected return code was encountered. RACF was processing a change log request, or a password envelope request, or both for user *name*. Neither function succeeded.

**Note:** Due to an internal method being used, *return-code* will not be a SAF return code as documented in the *z/OS Security Server RACROUTE Macro Reference* (though *racf-return-code* and *racf-reason-code* will match a documented combination).

**System Action:** The system continues processing.

**System Programmer Response:** Contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC136I RACF ENCOUNTERED A RACROUTE REQUEST=AUTH ERROR WHILE PROCESSING USER *name*. RETURN CODE=*return-code*, RACF RETURN CODE=*racf-return-code*, RACF REASON CODE=*racf-reason-code*.**

**Explanation:** A RACROUTE REQUEST=AUTH was attempted but an unexpected return code was encountered. RACF was attempting to check a user's eligibility for PKCS#7 password enveloping, by checking the user's access to PASSWORD.ENVELOPE in the RACFEVNT class. The user's password was not enveloped, and if LDAP event notification was active, no LDAP change log entry was created.

**Note:** Due to an internal method being used, *return-code* might not be a SAF return code as documented in the *z/OS Security Server RACROUTE Macro Reference* (though *racf-return-code* and *racf-reason-code* will match a documented combination).

**System Action:** The system continues processing.

**System Programmer Response:** Contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC137I RACF RECEIVED CEEPIPI RETURN CODE *rc* FROM FUNCTION *fcn* WHILE PROCESSING USER *name*. CEEPIPI RESPONSE CODE *hi-resp low-resp*, REASON CODE *hi-reas low-reas*, FEEDBACK CODE *hi-feed low-feed*.**

**Explanation:** An error was encountered during PKCS#7 envelope processing in the LE interface used to set up the environment necessary for the execution of C language code. The function *fcn* identifies the internal C language function that was being invoked. The high-order four bytes and low-order four bytes of the response, reason, and feedback codes returned by CEEPIPI are displayed separately in hexadecimal. The user's password was not enveloped, and if LDAP event notification was active, no LDAP change log entry was created.

**System Action:** The system continues processing.

**System Programmer Response:** Contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC138I RACF ENCOUNTERED AN UNEXPECTED PKCS#7 ENVELOPING ERROR WHILE PROCESSING USER *name*. R15=*contents*, OPERATION CODE=*opcode*, RC1=*rc1*, RC2=*rc2*, RC3=*rc3*.**

**Explanation:** An unexpected error was encountered during PKCS#7 envelope processing for user *name*. The various diagnostic values are displayed. The user's password was not enveloped, and if LDAP event notification was active, no LDAP change log entry was created.

**System Action:** The system continues processing.

**System Programmer Response:** Contact the customer support center.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC139I THE NUMBER OF PASSWORD RECIPIENT CERTIFICATES ON IRR.PWENV.KEYRING EXCEEDS THE MAXIMUM OF 20. THE KEY RING IS OWNED BY THE RACF SUBSYSTEM ID *user*.**

**Explanation:** A PKCS#7 password envelope was being processed by the RACF subsystem in response to a request to the R\_admin (IRRSEQ00) callable service. RACF only supports up to 20 recipients, each of which is identified by a certificate on the

IRR.PWENV.KEYRING key ring. This key ring is owned by the identity under which the RACF subsystem is running, displayed in the message as *user*.

**System Action:** The password has been enveloped for only the first 20 certificates encountered (not including RACF's certificate, which is the default certificate on the key ring). The system continues processing.

**Security Administrator Response:**

To avoid having this message displayed every time a password envelope is requested, you can remove some certificates from the key ring using the RACDCERT command. You can see the contents of the key ring by issuing the following command:

```
RACDCERT ID(user) LISTRING(IRR.PWENV.KEYRING)
```

**Note:** The key ring name is case sensitive, and so must be typed in upper case.

Proper authority will be required in order to issue this command. See the *z/OS Security Server RACF Command Language Reference* for details on the RACDCERT command.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRC141I THE PASSWORD ENVELOPING FUNCTION CANNOT BE PERFORMED FOR USER *user1*. A PROPER UNIX SYSTEM SERVICES ENVIRONMENT DOES NOT EXIST FOR THE RACF SUBSYSTEM RUNNING UNDER USER ID *user2*.**

**Explanation:** RACF attempted to build a PKCS#7 password envelope for user ID *user1*, but was unsuccessful because the RACF subsystem is not running as a UNIX process. The most likely cause of this failure is one of the following:

1. The subsystem address space identity (identified in the message as *user2*) does not have an OMVS segment. In this case, message IRRB023I should have been issued during subsystem initialization.
2. The password enveloping function was activated (by defining the PASSWORD.ENVELOPE profile in the RACFEVNT class, and activating the class), but the RACF subsystem address space was not stopped and restarted.

**System Action:** The password enveloping function was not performed. The system continues processing.

**Security Administrator Response:**

In the case of number 1 above, define the RACF subsystem user ID as a z/OS UNIX user by defining an OMVS segment for its USER profile, and for its default group profile. See the RACF Security Administrator's Guide for more details on defining UNIX users. The RACF subsystem address space must be stopped and

restarted after the OMVS information has been defined.

In the case of number 2 above, stop and restart the RACF subsystem address space. RACF will recognize that password enveloping has been configured, and will invoke the proper UNIX services to dub itself as a UNIX process.

In order to avoid interruptions in services provided by the address space, it should be restarted during a period of low activity. See the *z/OS RACF System Programmer's Guide* for information on the RACF subsystem.

**Destination:** Descriptor code is 6. Routing code is 2.

---

**IRRB023I SYSTEM SERVICE *service* FAILED WITH RETURN CODE *return-code*, REASON CODE *reason-code*.**

**Explanation:** RACF invoked a UNIX service to dub the RACF subsystem address space as a UNIX process, but the service failed with the return and reason codes specified. RACF only invokes these services during subsystem initialization if a RACF function which requires UNIX (e.g. password enveloping) has been enabled. The most likely cause of this failure is that the subsystem address space identity does not have an OMVS segment. If this is the case, you might see an ICH408I message in addition to IRRB023I.

**Note:** If the RACF subsystem address space has the PRIVILEGED attribute, then the ICH408I message will not be displayed because audit records are not created for PRIVILEGED tasks

**System Action:** The RACF subsystem continues to initialize. Only those functions which require UNIX system services will be unavailable.

**Security Administrator Response:**

If the subsystem address space identity is not defined as a UNIX user, then define an OMVS segment for its USER profile, and for its default group profile. See the *z/OS RACF Security Administrator's Guide* for more details on defining UNIX users. The RACF subsystem address space must be stopped and restarted after the OMVS information has been defined in order for the UNIX functions to become available. In order to avoid interruptions in services provided by the address space, it should be restarted during a period of low activity. See the *z/OS RACF System Programmer's Guide* for information on the RACF subsystem.

If the RACF subsystem address space identity is properly defined as a UNIX user, then consult *z/OS UNIX System Services Programming: Assembler Callable Services Reference* for the meaning of the return and reason codes

**Destination:** Descriptor code is 6. Routing code is 2.

## Chapter 8. Macros and interfaces considerations

### Updates to the RACF database templates

PWDENV

The field PWDENV is added to the user template (base segment):

Template	Field Being Described						
Field Name (Character Data)	Field ID	Flag 1	Flag 2	Field Length Decimal	Default Value	Type	
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
PWDENV	085	00	08	00000000	00	Bin	Internal form of enveloped RACF password.

### Updates to database unload

The field USBD\_PWDENV\_EXISTS is added to the user basic data record (0200).

Table 15. Update to user basic data record (0200)

Field Name	Type	Position		Comments
		Start	End	
USB_RECORD_TYPE	Int	1	4	Record type of the User Basic Data record (0200).
⋮	⋮	⋮	⋮	⋮
USB_ATTRIBS	Char	542	549	Other user attributes (RSTD for users with RESTRICTED attribute).
USB_PWDENV_EXISTS	Yes/No	551	554	Has a PKCS#7 envelope been created for the user's current password?

### Class descriptor table updates

Table 16.

Class		
RACFEVNT	POSIT=574	OTHER=ANY
	RACLIST=ALLOWED	MAXLNTH=246
	GENLIST=ALLOWED	DFTRETC=4
	RACLREQ=NO	DFTUACC=NONE
	OPER=NO	ID=1
	FIRST=ANY	

### Router table updates

The new entry class, RACFEVNT class, is added to the SAF router table (ICHRFR0X):

```
ICHRFR0X CLASS=RACFEVNT,ACTION=RACF
```



---

## Chapter 9. Command considerations

---

## SET Command

### Purpose

Use the SET command to:

- List information related to RRSF on the local node
- Specify the name of a member of the RACF parameter library to be processed by RACF
- Set tracing on or off for specified RACF subsystem facilities
- Enable and specify options for automatic direction

**Note:** You might find it useful to fill out the “Configuration Worksheet” in the *z/OS Security Server RACF System Programmer’s Guide* to help you determine the information you need to issue the SET command.

### Usage

The following table identifies the eligible options for issuing the SET command:

*Table 17. How the SET Command Can be Issued*

As a RACF TSO Command?	As a RACF Operator Command?	With Command Direction?	With Automatic Command Direction?	From the RACF Parameter Library?
No	Yes	No	No	Yes

**Note:** You must be logged on to the console to issue this command as a RACF operator command.

### Authorization

When issuing the SET command as a RACF operator command, you might require sufficient authority to the proper resource in the OPERCMDS class. See *z/OS Security Server RACF Security Administrator’s Guide* for further information.

## Format

The complete syntax of the command is:

```

subsystem-prefix SET
    [ AUTOAPPL([
        [ NOTIFY(notify-level(list-of-notify-users)) | NONOTIFY ]
        [ OUTPUT(output-level(list-of-output-users)) | NOOUTPUT ] ) ]
    | NOAUTOAPPL ]
    [ AUTODIRECT([
        [ NOTIFY(notify-level(list-of-notify-users)) | NONOTIFY ]
        [ OUTPUT(output-level(list-of-output-users)) | NOOUTPUT ] ) ]
    | NOAUTODIRECT
    ]
    [ AUTOPWD([
        [ NOTIFY(notify-level(list-of-notify-users)) | NONOTIFY ]
        [ OUTPUT(output-level(list-of-output-users)) | NOOUTPUT ] ) ]
    | NOAUTOPWD
    ]
    [ INCLUDE(member-suffix...) ]
    [ JESNODE(nodename) ]
    [ LIST ]
    [ PWSYNC([
        [ NOTIFY(notify-level(list-of-notify-users)) | NONOTIFY ]
        [ OUTPUT(output-level(list-of-output-users)) | NOOUTPUT ] ) ]
    | NOPWSYNC
    ]
    [ TRACE( {
        [ APPC | NOAPPC ]
        [ ASID(asid ... !*)|NOASID | ALLASIDS ]
        [ CALLABLE(ALL | NONE | TYPE(type ...)) ]
        | NOCALLABLE ]
        [ DATABASE({
            [ ALL | NONE ]
            [ ALTER | NOALTER ]
            [ ALTERI | NOALTERI ]
            [ READINOREAD ] }) ]
        | NODATABASE ]
        [ IMAGE | NOIMAGE ]
        [ JOBNAME(jobname ... !*) ]
        | NOJOBNAME | ALLJOBNAMES ]
        [ PDCALLABLE(ALL | NONE | TYPE(type ...)) ]
        | NOPDCALLABLE ]
        [ RACROUTE(ALL | NONE
            | TYPE(type ...) | NORACROUTE ]
        [ SYSTEMSSL | NOSYSTEMSSL ] } ) ]

```

## Parameters

...

### SYSTEMSSL | NOSYSTEMSSL

#### SYSTEMSSL

Use to trace RACF's use of z/OS System Secure Sockets Layer (SSL) services. The actual trace records are created by the System SSL component itself; RACF only requests that the trace records be created. SSL services are used by RACF to create PKCS#7 password envelopes for

## SET

users permitted to the PASSWORD.ENVELOPE profile in the RACFEVNT class. For more information, see *z/OS Security Server RACF Security Administrator's Guide*

When SET TRACE(SYSTEMSSL) is in effect, all trace functions are requested. The trace records are written to a UNIX file with the path name of /tmp/gskssl.racf.pid.trc, where *pid* is the UNIX process ID assigned to the RACF thread that attempted the password envelope operation (initial creation during password change, or retrieval by using the R\_admin callable service). It is difficult to determine the *pid* for a given password envelope operation because it exists transiently. To debug a reproducible problem, look at the trace files that already exist in /tmp starting with 'gskssl.racf', or delete all the trace files, initiate the password envelope operation, find the new file that was created, and look within that file for the trace data. For more information on creating trace records, see *z/OS System Secure Sockets Layer Programming*

### NOSYSTEMSSL

Deactivates tracing for RACF's use of System SSL services.

...

## Examples

The output of the SET LIST command has been updated:

```
RACFR12 IRRH005I (@) RACF SUBSYSTEM INFORMATION:
TRACE OPTIONS
- IMAGE
- NOAPPC
- SYSTEMSSL
- RACROUTE
  2 5 9
- NOCALLABLE
- NODATABASE
- ASID
  17
- NOJOBNAME
SUBSYSTEM USERID - RACFSUB
JESNODE (FOR TRANSMITS) - WCC
AUTOMATIC COMMAND DIRECTION IS *NOT* ALLOWED
AUTOMATIC PASSWORD DIRECTION IS *NOT* ALLOWED
PASSWORD SYNCHRONIZATION IS *NOT* ALLOWED
AUTOMATIC DIRECTION OF APPLICATION UPDATES IS *NOT* ALLOWED
RACF STATUS INFORMATION:
  TEMPLATE VERSION - OA03853
  DYNAMIC PARSE VERSION - HRF7709
```

Figure 1. Output for the SET LIST Command

---

## Chapter 10. System programmer considerations

The section on the RACF subsystem in Chapter 4. "Operating considerations" in *z/OS Security Server RACF System Programmer's Guide* is updated to reflect that LDAP change log notification and password enveloping require the RACF subsystem to be active.

---

### The RACF subsystem

The RACF subsystem enables remote RACF administration and password synchronization, provides an execution environment for most RACF commands and provides support for APPC persistent verification. Starting the subsystem is optional but recommended. It is not necessary for system IPL or most RACF functions, but it is required for the following functions:

- RACF remote sharing facility  
The RACF subsystem is required for the RACF remote sharing facility to be operational. For more information see Chapter 5, "RACF remote sharing facility (RRSF)".
- RACF commands as operator commands  
When the RACF subsystem is active, most RACF commands can be issued as operator commands. For more information, see "RACF operator commands"..
- R\_admin (IRRSEQ00) callable service  
When the RACF subsystem is active, it executes commands which are passed to it by R\_admin. Applications that use R\_admin, such as TME 10 User Administration, require the RACF subsystem to be active.
- RACF LU6.2 persistent verification  
The RACF subsystem provides a centralized data owner/data server environment for the signed-on lists used by RACF persistent verification. The lists are managed with the RACROUTE REQUEST=SIGNON macro. RACF also provides an execution environment for the RACF persistent verification operator commands, DISPLAY and SIGNOFF.
- LDAP change log notification  
When LDAP change notification is configured, the RACF subsystem calls the LDAP server to create LDAP change log entries when RACF user profiles are added, modified, or deleted.
- Password enveloping  
When PKCS#7 password enveloping is configured, the RACF subsystem creates an encrypted password envelope when passwords are changed for eligible users. The clear text password can be recovered by authorized recipients with certificates on a key ring owned by the RACF subsystem.

The RACF subsystem address space is identified as a standard MVS subsystem. The RACF subsystem provides the following services:

- Automatic start of the RACF subsystem at IPL time.
- Tailorability through startup parameters. The RACF subsystem reads startup parameters from the IEFSSNxx member of SYS1.PARMLIB and the PARM keyword on the EXEC statement in the subsystem procedure.
- Optional subsystem command identifiers. You can choose to use the MVS subsystem convention of assigning a unique subsystem prefix or you can use the unique subsystem name, followed by a blank, as the prefix for the RACF subsystem.

This unique subsystem name is defined in the IEFSSNxx member of SYS1.PARMLIB.

Only one RACF subsystem can run at a time. If you define more than one RACF subsystem with the same name in IEFSSNxx, only one starts. It is possible to define two RACF subsystems with different names, and start the second one after stopping the first, but this is not recommended. If you choose to do this, you must specify PARM=INITIAL on the MVS START command whenever you start a RACF subsystem that has a different name than the one that was previously running.

---

## Trademarks

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

IBM

MVS

RACF

z/OS

z/OS.e

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