z/OS

Security Server RACF Callable Services

Version 2 Release 1

SA23-2293-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 393.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this document

This document supports z/OS® (5650-ZOS) and contains information about the Resource Access Control Facility (RACF®), which is part of z/OS Security Server.

This document documents callable services provided by RACF. It contains:

- Information about using the callable services
- A description of each callable service
- Descriptions of the data areas used by the callable services
- A description of an installation exit that can be used in conjunction with the callable services

Intended audience

This document is intended for system programmers who are familiar with RACF concepts and terminology. They should also be familiar with MVS™ systems and with z/OS UNIX.

Where to find more information

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, including the z/OS Information Center, see z/OS Internet Library (http://www.ibm.com/systems/z/os/zos/bkserv/).

RACF courses

The following RACF classroom courses are available in the United States:

ES191 *Basics of z/OS RACF Administration*

BE870 *Effective RACF Administration*

ES885 *Exploiting the Advanced Features of RACF*

IBM® provides various educational offerings for RACF. For more information about classroom courses and other offerings, do any of the following:

- See your IBM representative
- Call 1-800-IBM-TEACH (1-800-426-8322)

Other sources of information

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

Internet sources

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

- **Online library**

  To view and print online versions of the z/OS publications, use this address:

  `http://www.ibm.com/systems/z/os/zos/bkserv/`
- **Redbooks®**

  The documents known as IBM Redbooks that are produced by the International Technical Support Organization (ITSO) are available at the following address:

  `http://www.redbooks.ibm.com`
- **Enterprise systems security**

  For more information about security on the S/390® platform, and z/OS, including the elements that comprise the Security Server, use this address:

  `http://www.ibm.com/systems/z/advantages/security/`
- **RACF home page**

  You can visit the RACF home page on the World Wide Web using this address:

  `http://www.ibm.com/systems/z/os/zos/features/racf/`
- **RACF-L discussion list**

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

  To subscribe to the RACF-L discussion and receive postings, send a note to:

  `listserv@listserv.uga.edu`

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

  `subscribe racf-l` *`first_name last_name`*

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

  `racf-l@listserv.uga.edu`
- **Sample code**

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

  To access this code from a Web browser, go to the RACF home page and select the "Resources" file tab, then select "Downloads" from the list, or go to http://www-03.ibm.com/systems/z/os/zos/features/racf/goodies.html.

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

  1. Log in as user **anonymous**.
  2. Change the directory, as follows, to find the subdirectories that contain the sample code or tool you want to download:

     `cd eserver/zseries/zos/racf/`

  An announcement will be posted on the RACF-L discussion list whenever something is added.

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

  – Try to get access by using a Web browser and the links from the RACF home page.

- Use a different FTP client. If necessary, use a client that is based on command line interfaces instead of graphical interfaces.
- If your FTP client has configuration parameters for the type of remote system, configure it as UNIX instead of MVS.

Restrictions

Because the sample code and tools are not officially supported,

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

How to send your comments to IBM

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

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 Security Server RACF Callable Services
  SA23-2293-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

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

If you have a technical problem

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

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

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Using the RACF callable services

The RACF security functions provided for use by z/OS UNIX and other products integrated with it are called as callable services. Normal installation applications using the services or functions of z/OS UNIX cannot call the RACF callable services directly. They must use the z/OS UNIX callable services instead.

Linkage conventions for the callable services

The linkage is created as follows:

- For non-IBM modules, or modules written in languages other than PL/X, the CALL statement must generate a V-type constant (VCON) with the module name of a stub routine for the requested service. The module names are defined as part of the callable services interface described in Chapter 2, "Callable services descriptions," on page 9. The VCON can be resolved by link-editing the control section (CSECT) with the stub routines provided as part of MVS's system authorization facility (SAF). There is a stub for each service.
- The linkage loads a function code indicating the service requested and calls the callable services router. The function codes that can be used are described in *z/OS Security Server RACF Data Areas*.
- The callable services router calls an installation exit (IRRSXT00 for 31 bit callers or IRRSXT0X for 64 bit callers).
- The RACF router invokes the requested service routine based on the function code.
- The service routine provides the requested function and returns to the SAF callable services router.
- The SAF callable services router calls the installation exit IRRSXT00 or IRRSXT0X a second time, sets the SAF return code, and returns to the caller.

Working with return and reason codes

All return and reason codes are in decimal.

Work area (WORK)

When a module calls a RACF callable service, it must provide the address of a work area. The work area is a 1024-byte structure that is used by SAF, RACF, and the SAF exit routine IRRSXT00. IRRSXT00 can use the first 152 bytes of the area. The first 16 bytes are preserved from the pre-RACF exit invocation to the post-RACF exit invocation and can be used to pass parameters.

For the mapping of the work area, see *z/OS Security Server RACF Data Areas*. For information about IRRSXT00, see Chapter 3, "Installation exits," on page 353.

File security packet (IFSP)

Security-relevant data for files in the z/OS UNIX file system is kept in a file security packet (IFSP) structure owned by RACF. The IFSP is stored in the file system as part of the attributes associated with a file. When a file is created, the IFSP is created by the **makeFSP** or the **make_root_FSP** callable service. The **makeFSP** service returns an IFSP to the file system, which writes it with other

attributes of the file. On subsequent accesses to the file, the file system reads the IFSP and passes it to other callable services. The file system deletes the IFSP when the file is deleted.

The IFSP is a fixed-size 64-byte area. It is written to storage as part of the PFAR for the file and its size cannot be changed.

The file system manages the storage for the IFSP. The **makeFSP** service fills in the data, and other callable services use or modify the data in the area provided by the file system.

The IFSP data can be examined by users other than the security product. The IFSP is mapped by macro IRRPIFSP. Others should not use this mapping to create or directly modify the IFSP, and should not make their own security or audit decisions based on the contents of the IFSP.

The IFSP contains the following data:

- Control block ID
- Version number
- z/OS UNIX user identifier (UID) of the owner of the file
- z/OS UNIX group identifier (GID) of the group owner of the file
- Mode bits:
  - Owner permission bits
  - Group permission bits
  - Other permission bits
  - S_ISUID, S_ISGID, and S_ISVTX bits
- User audit options for the file
- Auditor audit options for the file
- Security label (SECLABEL) of the file

For the mapping of the file security packet, see *z/OS Security Server RACF Data Areas*.

Security credentials (CRED)

The security credentials (CRED) structure is used in the z/OS UNIX file system to pass data from the logical file system (LFS) through the physical file system (PFS) to the RACF callable services.

The CRED is built by the LFS, and is created for each system call entry to the LFS. The CRED is used for all vm_ops called (and most RACF callable service calls by the PFS) for the system call. The CRED is not kept across multiple LFS system calls.

The CRED contains:

- **User information:** a user type field that indicates whether the caller is a standard z/OS UNIX process known to RACF, or a system function that is not a process.

  Functions that accept a system caller process the request as if the caller is a superuser. If an audit record is written, the user z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) values in the record are set to -1.

- **Audit data:** data known by the LFS that must be passed through the PFS to the RACF callable services for auditing. This data is:
  - **Audit function code:** a code that identifies the system call being processed. The audit function codes are described in *z/OS Security Server RACF Data Areas*.
  - **Name flag:** a flag used on path resolution calls to **ck_access** to indicate whether the first or second file name is being checked.
  - **Requested path name:** the path name the user passed on the system call. For link, vlink, rename, and vrename, this is the old path name. When the caller of lookup is getcwd, ioctl, or ttyname, this field is not completed.
  - **File name:** the part of the requested path name currently being checked. This may be part of the path name or may be part of a symbolic link encountered when resolving the path name. The first directory checked in a path name resolution is either the root directory (**/ROOT**) or the current working directory (**/CWD**). The names **/ROOT** and **/CWD**-the only file names that contain a slash (/)-are provided to indicate these directories in the audit record. This field is included only in audit records produced by **ck_access**.

    This field contains the file name of:
    - The directory being checked on calls from lookup.

      When the caller of lookup is getcwd, ioctl, or ttyname, this field is not completed.
    - The parent directory of the object identified by the path name for calls for mkdir, mknod, vcreate, open(new file), rename, vrename, rmdir, symlink, vsymlink, unlink, and vremove.
    - The object identified by the path name for calls for open(old file), opendir, link, vlink, and utime.
    - **Second path name:** for rename, vrename, link, and vlink, this is the new path name passed on the system call. For symlink and vsymlink, this is the content of the symlink. For mount and unmount, this is the data set name of the shared file system data set being mounted or dismounted.
    - **Second file name:** this is the same as the file name above, except that it is for the second part of the path name being checked. This field contains the file name of:
      - The directory being checked on calls from lookup
      - The parent directory of the object identified by the new path name for calls for link, vlink, rename, and vrename.
  - **Access Control List information:** pointers to ACL buffers are used for the ck_access, makeFSP, and R_Setfacl callable services.
  - **Security Label:**
    - used to pass the security label to be set for the file or directory to R_setfsecl
    - used in a system CRED to pass the security label to set when the directory's security label is SYSMULTI for makeFSP
    - used in a system CRED to indicate that a directory requires a SYSMULTI security label to pass an access request for ck_access.
    - an ACEE pointer, used to pass the ACEE address of a user performing a socket call in SRB mode from UNIX System Services to the IP stack, a read-only security label, used to pass the resource security label for read-only file systems to ck_access.
  - **File System Name:** the name of the file system containing the specified file name(s). If supplied by LFS and the FSACCESS class is RACLISTed, RACF verifies the user has access to a matching profile defined in that class.

The CRED structure is mapped by the IRRPCRED mapping macro.

For the mapping of the CRED, see *z/OS Security Server RACF Data Areas*.

File identifiers

Programming interface information

Part of the audit data for file access is a file identifier. The file identifier is a 16-byte token that uniquely identifies a file while it is mounted on the system.

End Programming interface information

Programming interface information

>If the file system is unmounted and remounted, that file identifier may change. A change in file identifiers can be detected in the audit trail by matching the mount audit records with the same file system name and comparing the file identifiers for the root directory.

End Programming interface information

File type and file mode values

Programming interface information

A mode value is input to z/OS UNIX chmod, open, creat, mkdir, and umask, and output by z/OS UNIX stat and fstat. The mode value is defined as a mode_t data type and consists of a one-byte file type and three bytes for the file modes. The file mode specifies the permission bits and the S_ISUID, S_ISGID, and S_ISVTX bits for a file.

End Programming interface information

Programming interface information

The z/OS UNIX macro BPXYMODE defines the mode_t values as:

- Bits 0–7: file type, mapped by z/OS UNIX macro BPXYFTYP
- Bits 8–13: reserved
- Bits 14–31: available to the security product:
  - Bits 14–19: reserved
  - Bit 20: S_ISUID (set user ID on execution)
  - Bit 21: S_ISGID (set group name on execution)
  - Bit 22: S_ISVTX (keep loaded executable in storage)
  - Bits 23–25: S_IRWXU (owner class mask)
    - Bit 23: S_IRUSR (read permission)
    - Bit 24: S_IWUSR (write permission)
    - Bit 25: S_IXUSR (search (if directory) or execute (otherwise) permission)
  - Bits 26–28: S_IRWXG (group class mask)
    - Bit 26: S_IRGRP (read permission)
    - Bit 27: S_IWGRP (write permission)

- Bit 28: S_IXGRP (search (if directory) or execute (otherwise) permission)

– Bits 29–31: S_IRWXO (other class mask)
  - Bit 29: S_IROTH (read permission)
  - Bit 30: S_IWOTH (write permission)
  - Bit 31: S_IXOTH (search (if directory) or execute (otherwise) permission)

End Programming interface information

Programming interface information

The system call services pass the mode parameter from the caller of the system call to the RACF callable service or from the RACF callable service to the caller of the system call. The system call service can change the file type but does not change the file mode bits.

End Programming interface information

Programming interface information

Some RACF callable services test the file type to determine if the file is a directory. The **makeFSP** service sets the file type to "directory" if the file is a directory and sets it to zero otherwise.

End Programming interface information

IPC security packet (IISP)

Interprocess communication (IPC) requires RACF to do authorization and permission checking. IPC facilities of the z/OS UNIX system allow two or more distinct processes to communicate with each other. RACF protects this environment so that only those processes with the correct authority can communicate.

Interprocess communication consists of message queueing, semaphores, and shared memory segments used by application programs. Each function requires a security action by z/OS UNIX, which RACF performs to allow a secure environment to exist.

The IPC security packet (IISP) contains data needed to make security decisions. It is built when a new ID for an IPC key is created and is saved in memory by the kernel. The IISP is used in place of a profile in the RACF database to contain information about the IPC key's owner and access rights.

The **makeISP** service initializes the IPC security packet (IISP) for a new IPC key with the creator's user and group identifiers (UID and GID), the owner's UID and GID, the mode bits, the IPC key, and the IPC ID.

The **ck_IPC_access** service determines whether the current process has the requested access to an IPC key. The IISP of the key is passed with this request. The **ck_IPC_access** service is called separately for each IPC key.

For the z/OS UNIX IPC_SET command, the **R_IPC_ctl** service modifies the owner's UID, owner's GID, and mode bits in the IISP for the IPC key if the

authority is correct. For the z/OS UNIX IPC_RMID command, the **R_IPC_ctl** service checks the authority of the current process to determine whether the resource can be removed.

The IISP consists of two parts, the root and the extension. The root is mapped by macro IRRPIISP. The root contains a pointer to the extension, which is mapped by the z/OS UNIX mapping macro BPXYIPCP. Other products can read the IISP for reporting purposes using the IRRPIISP and BPXYIPCP mapping macros.

The IISP root contains the following data:

- Control block ID
- Version number
- ALET of the IPCP
- Address of the IPCP (mapped by z/OS UNIX macro BPXYIPCP)
- IPC key
- IPC ID
- Security label (SECLABEL)

For the mapping of the IPC security packet, see *z/OS Security Server RACF Data Areas*.

Interprocess communications permission (BPXYIPCP)

```
         BPXYIPCP   ,
** BPXYIPCP: Interprocess Communications Permission
**  Used By: MCT, MGT, SCT, SGT, QCT, QGT
IPC_PERM            DSECT ,      Interprocess Communications
IPC_UID             DS    F      Owner's effective user ID
IPC_GID             DS    F      Owner's effective group name
IPC_CUID            DS    F      Creator's effective user ID
IPC_CGID            DS    F      Creator's effective group name
IPC_MODE            DS    XL4    Mode, mapped by BPXYMODE
IPC#LENGTH   EQU   *-IPC_PERM  Length of Interprocess Control block
* Key:
IPC_PRIVATE         EQU   0      Private key.
* Mode bits:                    Map over S_TYPE in BPXYMODE
IPC_CREAT           EQU   1      Create entry if key does not exist.
IPC_EXCL            EQU   2      Fail if key exists.
* Flag bits - semop, msgrcv, msgsnd:
IPC_NOWAIT          EQU   1      Error if request must wait.
* Control Command:
IPC_RMID            EQU   1      Remove identifier.
IPC_SET             EQU   2      Set options.
IPC_STAT            EQU   3      Access status.
* CONSTANTS WHICH MAP OVER BYTE S_TYPE, SEE BPXYMODE
** BPXYIPCP End
```

IPC security credentials (CREI)

The IPC security credentials (CREI) structure is used in the z/OS UNIX IPC system to pass data from the kernel to RACF.

The CREI is built by the kernel, and is created for each system call entry to RACF.

The CREI contains:

- **User information:** a user type field that indicates whether the caller is a standard z/OS UNIX process known to RACF, or a system function that is not a process.

Functions that accept a system caller process the request as if the caller is a superuser. If an audit record is written, the user z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) values in the record are set to -1.

- **Audit data:** data known by the kernel that needs to be passed through the IPC system to the RACF callable services for auditing. This data includes an **audit function code,** which identifies the system call being processed. The audit function codes are described in *z/OS Security Server RACF Data Areas*.
- **IPC key:** the key of the IPC service that is being checked.
- **IPC identifier:** the identifier of the IPC service that is being checked.

The CREI structure is mapped by the IRRPCREI mapping macro.

For the mapping of the CREI, see *z/OS Security Server RACF Data Areas*.

Chapter 2. Callable services descriptions

This chapter describes the RACF callable services. The services appear in alphabetic order. Table 1 lists each callable service's intended users.

Table 1. Intended use of RACF callable services

Callable service	For use by
"ck_access (IRRSKA00): Check access" on page 11	z/OS UNIX file system or z/OS UNIX servers
"ck_file_owner (IRRSKF00): Check file owner" on page 15	z/OS UNIX file system or z/OS UNIX servers
"ck_IPC_access (IRRSKI00): Check IPC access" on page 17	MVS BCP or z/OS UNIX task-level processes
"ck_owner_two_files (IRRSC200): Check owner of two files" on page 20	z/OS UNIX file system and z/OS UNIX servers.
"ck_priv (IRRSKP00): Check privilege" on page 22	z/OS UNIX file system, MVS BCP, or z/OS UNIX servers
"ck_process_owner (IRRSKO00): Check process owner" on page 25	MVS BCP or z/OS UNIX task-level processes
"clear_setid (IRRSCS00): Clear set ID" on page 27	z/OS UNIX file system or z/OS UNIX servers
"deleteUSP (IRRSDU00): Delete USP" on page 29	MVS BCP or z/OS UNIX servers
"R_GetInfo (IRRSGI00): Get security server fields" on page 224	Enterprise identity mapping
"getGMAP (IRRSGM00): Get GID-to-Group-Name mapping" on page 31	MVS BCP
"get_uid_gid_supgrps (IRRSGE00): Get UIDs, GIDs, and supplemental groups" on page 33	z/OS UNIX file system
"getUMAP (IRRSUM00): Get UID-to-User-ID mapping" on page 36	MVS BCP
"initACEE (IRRSIA00): Initialize ACEE" on page 38	z/OS kernel on behalf of servers that use pthread_security_np servers or __login, or MVS servers that do not use z/OS UNIX services
"initUSP (IRRSIU00): Initialize USP" on page 55	MVS BCP or z/OS UNIX servers
"makeFSP (IRRSMF00): Make IFSP" on page 58	z/OS UNIX file system or z/OS UNIX servers
"makeISP (IRRSMI00): Make IISP" on page 62	MVS BCP or z/OS UNIX task-level processes
"make_root_FSP (IRRSMR00): Make root IFSP" on page 64	DFSMS/MVS or z/OS UNIX servers
"query_file_security_options (IRRSQF00): Query file security options" on page 67	z/OS UNIX file system
"query_system_security_options (IRRSQS00): Query system security options" on page 69	MVS BCP
"R_admin (IRRSEQ00): RACF administration API" on page 71	Tivoli®
"R_audit (IRRSAU00): Provide an audit interface" on page 112	z/OS UNIX file system, MVS BCP, or z/OS UNIX servers
"R_auditx (IRRSAX00 or IRRSAX64): Audit a security-related event" on page 115	Enterprise identity mapping
"R_cacheserv (IRRSCH00): Cache services" on page 125	Policy Director
"R_chaudit (IRRSCA00): Change audit options" on page 149	z/OS UNIX file system or z/OS UNIX servers

Table 1. Intended use of RACF callable services (continued)

Callable service	For use by
"R_chmod (IRRSCF00): Change file mode" on page 152	z/OS UNIX file system or z/OS UNIX servers
"R_chown (IRRSCO00): Change owner and group" on page 154	z/OS UNIX file system or z/OS UNIX servers
"R_datalib (IRRSDL00 or IRRSDL64): OCSF data library" on page 158	MVS BCP or z/OS UNIX servers
"R_dceauth (IRRSDA00): Check a user's authority" on page 183	MVS BCP
"R_dceinfo (IRRSDI00): Retrieve or set user fields" on page 187	MVS BCP
"R_dcekey (IRRSDK00): Retrieve or set a non-RACF password" on page 191	MVS BCP
"R_dceruid (IRRSUD00): Determine the ID of a client" on page 195	z/OS UNIX servers or MVS BCP
"R_exec (IRRSEX00): Set effective and saved UIDs/GIDs" on page 199	MVS BCP or z/OS UNIX task-level processes
"R_fork (IRRSFK00): Fork a process" on page 201	MVS BCP or z/OS UNIX task-level processes
"R_GenSec (IRRSGS00 or IRRSGS64): Generic security API interface" on page 204	MVS BCP
"R_getgroups (IRRSGG00): Get/Set supplemental groups" on page 219	MVS BCP or z/OS UNIX servers
"R_getgroupsbyname (IRRSUG00): Get groups by name" on page 221	MVS BCP
"R_IPC_ctl (IRRSCI00): Perform IPC control" on page 230	MVS BCP or z/OS UNIX task-level processes
"R_kerbinfo (IRRSMK00): Retrieve or set security server network authentication service fields" on page 233	For use by the z/OS Network Authentication Service
"R_PgmSignVer (IRRSPS00): Program Sign and Verify" on page 239	The z/OS program binder and the z/OS loader
"R_proxyserv (IRRSPY00): LDAP interface" on page 308	Policy Director
"R_ptrace (IRRSPT00): Ptrace authority check" on page 316	MVS BCP or z/OS UNIX task-level processes
"R_setegid (IRRSEG00): Set effective GID, set all GIDs" on page 318	MVS BCP
"R_seteuid (IRRSEU00): Set effective UID, set all UIDs" on page 320	MVS BCP
"R_setfacl (IRRSCL00):Unix access control lists" on page 322	z/OS UNIX file system or z/OS UNIX servers
"R_setfsecl (IRRSSB00): Security label" on page 326	MVS BCP or z/OS UNIX servers
"R_setgid (IRRSSG00): Set group name" on page 328	MVS BCP
"R_setuid (IRRSSU00): Set z/OS UNIX user identifier (UID)" on page 330	MVS BCP
"R_ticketserv (IRRSPK00): Parse or extract" on page 332	Parse or extract
"R_umask (IRRSMM00): Set file mode creation mask" on page 340	MVS BCP or z/OS UNIX servers
"R_usermap (IRRSIM00): Map application user" on page 342	z/OS application servers

Table 1. Intended use of RACF callable services (continued)

Callable service	For use by
"R_writepriv (IRRSWP00): Write-down privilege" on page 350	MVS BCP

Note: In a server environment, work can be processed for more than one user in an address space. Callable services marked for use by z/OS UNIX servers provide task-level support for server applications. Callable services marked as having support for task-level processes use task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process. All other callable services assume that there is only one user per address space and provide only address-space-level support.

ck_access (IRRSKA00): Check access

Function

The **ck_access** service determines whether the current process has the requested access to the element (directory or file) of a pathname whose IFSP, and ACL if it exists, is passed. It is called separately for each element.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user or any task if system user type is specified

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. If the audit function code in the CRED is access, the real z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) of the calling process are used on the authority checks. Otherwise, the effective UID and GID for the calling process are used.
2. If the calling user has auditor authority and the access requested is search, or the access requested is read for a directory, access is allowed. Security label checking is bypassed in this case. This lets an auditor set the auditor audit options on any file without requiring that the auditor be given search access rights to all directories.
3. If the CRED user type is system, IRRSKA00 allows any access except when the requested access is execute and no execute permission bits are set for the file. No UIDs are used in this case, because no process exists. If the user is not system, the security label is checked and the security label authorization checking will be performed. If the SECLABEL class is active, and both a system CRED and a directory FSP are received, and the system CRED contains a security label, the access request will fail with a security label failure if the directory's SECLABEL is not SYSMULTI. No other security label authorization checking is done when a system CRED is passed. No security label checking is performed for a system CRED, with the exception that if the SECLABEL class is active, and both a system CRED and a directory FSP are received, and the system CRED contains a security label, the access request will fail with a security label failure if the directory's SECLABEL is not SYSMULTI.
4. If the user does not have the RACF Auditor attribute, and a file system name was specified in the CRED, and the FSACCESS class is active and RACLISTed, RACF will check for a profile in the FSACCESS class that covers the file system name. If a matching profile is found and the user does not have at least UPDATE authority, access is denied. Otherwise, authorization is determined by subsequent checks.
5. If the SECLABEL class is active, security label checking is performed for any file or directory with a security label. The security label of the process will be evaluated by ck_access against the security label in the FSP of the object being accessed, according to the hierarchical security model used for mandatory access control. If MLFSOBJ is active, a failure will occur if the file or directory does not have a security label, unless the CRED contains a security label in the ROSeclabel field. This value is used to pass the resource security label for read-only file systems.
6. If the caller is not a superuser, the permission bits did not allow the requested access, *and* the audit function code is listed in Table 2 on page 13, an authorization check is performed on the corresponding resource in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

   If Table 2 does not result in a UNIXPRIV authorization check, the caller is not a superuser, the permission bits did not allow the requested access, the file is a directory, and requested access is search, a read authorization check is performed on the SUPERUSER.FILESYS resource in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

   If a matching ACL entry was encountered for the user, or for at least one of its groups, and the requested access was not granted, then substitute SUPERUSER.FILESYS.ACLOVERRIDE as the resource name in Table 2. If SUPERUSER.FILESYS.ACLOVERRIDE does not exist, the check is redriven for the SUPERUSER.FILESYS resource.

Table 2. UNIXPRIV class resource names used in ck_access

Audit function code	Resource name	Access required
OPEN (for read or search), OPENDIR, READLINK, STAT, REALPATH, LSTAT, EACCESS (for read), ACCESS (for read; if real, effective and saved match)	SUPERUSER.FILESYS	READ
OPEN (for write), EACCESS (for write), ACCESS (for write; if real, effective and saved match)	SUPERUSER.FILESYS	UPDATE
LINK, MKDIR, MKNOD, RENAME, RMDIR, SYMLINK, UNLINK	SUPERUSER.FILESYS	CONTROL

7. If the user being checked is a superuser, IRRSKA00 allows any access except when the requested access is execute and no execute permission bits are set for the file. The user is considered a superuser if the selected UID is 0 or if the ACEE indicates trusted or privileged authority. The superuser that has UID 0 will not automatically pass the security label check, however, the trusted or privileged will.
8. If the user is not system and is not a superuser, the permission bits and ACL (if one exists, and if the FSSEC class is active) for the file are checked to see if the access requested is allowed. If the selected UID matches the owner UID of the file, the owner permission bits are checked. If the UIDs don't match, the user ACL entries are checked. If the selected UID matches an ACL entry, the ACL entry bits are checked. If a matching ACL entry was not found for the user, the group bits and the group ACL entries are checked. The selected GID and supplemental GID are checked against the file owner GID and the group ACL entries, until a match is found which grants the requested access, or until all the GIDs have been checked. If no match was found, the other permission bits are checked, unless the user has the RESTRICTED attribute, the UNIXPRIV class is active, the resource named RESTRICTED.FILESYS.ACCESS is protected, and the user does not have at least READ access.
9. If the real, effective, and saved UID are the same, and if the real, effective, and saved GID are the same, UNIXPRIV will be checked for AFC_ACCESS.
10. For a detailed list of authorization steps for z/OS UNIX files and directories, see Appendix F, in the *z/OS Security Server RACF Security Administrator's Guide*.

Format

```
CALL IRRSKA00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Requested_access_code,
               ALET, FSP,
               ALET, File_identifier,
               ALET, CRED,
               ALET, Name_flag
              )
```

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 words 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.

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.

Requested_access_code
The name of a 1-byte field containing the requested access. The defined codes are:

X'00' no access

X'01' Execute access

X'02' Write access

X'03' Write and execute access

X'04' Read access

X'05' Read and execute access

X'06' Read and write access

X'07' Read, write, and execute access

X'81' Search access (against a directory)

X'87' Any access

FSP
The name of the IFSP for the file being accessed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See the *z/OS Security Server RACF Data Areas*. The CRED contains a pointer to the ACL, if one exists.

Name_flag
The name of a byte indicating which pathname and file name is being checked. The byte contains one of these values:

0 Use the CRED_name_flag to determine pathname being checked.

1 The old (or only) name is being checked.

2 The new name is being checked.

Return and reason codes

IRRSKA00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized to access the file.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. The access checks performed are POSIX file permission checks defined in POSIX 1003.1.
3. If the audit function code in the CRED is access or eaccess, no audit record is written. Access checking only tests whether a process would have access if it were running with its real UID. Eaccess checking only tests whether a process would have access with its effective UID. Neither gives access to the file.
4. If the calling syscall is not access (for real or effective UID), an audit record is optionally written, depending on the audit options in effect for the system.
5. The caller must pass in the address of the object's access ACL in the CredAclPtr field.

Related services

R_chmod, R_chown, R_setfacl

ck_file_owner (IRRSKF00): Check file owner

Function

The **ck_file_owner** service checks whether the calling process is a superuser or is the owner of the file represented by the input.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. The ck_file_owner service checks whether the calling process is a superuser.
2. The ck_file_owner service checks whether the calling process is the owner of the file represented by the input IFSP. A process is the owner of a file if the effective UID of the process is equal to the file's owner UID.
3. If the SECLABEL class is active and the file or directory has a security label, then the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the current security label of the process, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service is passed a system CRED.

Format

```
CALL IRRSKF00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, FSP,
               ALET, File_identifier,
               ALET, CRED
              )
```

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 words containing the ALETs must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

SAF_return_code
The name of a full word in which the SAF router returns the SAF return code.

RACF_return_code
The name of a full word in which the service routine stores the return code.

RACF_reason_code
The name of a full word in which the service routine stores the reason code.

FSP
The name of the IFSP for the file to be checked.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSKF00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized.
8	8	12	An internal error occurred during RACF processing.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

None

ck_IPC_access (IRRSKI00): Check IPC access

Function

The **ck_IPC_access** service determines whether the current process has the requested access to the interprocess communication (IPC) key or identifier whose IPC security packet (IISP) is passed.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user/any task if system user type is specified

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
None

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. The access checks performed are XPG4 IPC permission checks defined in XPG4 System Interfaces and Headers, as follows:
   - The effective z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) for the calling process is used for all access checks.
   - If the CREI user type is system, IRRSKI00 allows any access. No UIDs or GIDs are used in this case because no process exists.
   - If the user being checked is a superuser, IRRSKI00 allows any access. The user is considered a superuser if the selected UID is 0 or if the ACEE indicates trusted or privileged authority.
   - If the user is not system and is not a superuser, the permission bits for the IPC key are checked to see if the access requested is allowed. If the effective UID matches either the owner UID or creator's UID of the IPC key, the USER permission bits are checked. If the UIDs do not match, the owner GID and creator's GID of the IPC key are checked against the user's effective GID and the user's supplemental group list GIDs. If any one matches, the GROUP permission bits are checked. If the UIDs and GIDs don't match, the OTHER permission bits are checked.
   - If the SECLABEL class is active and the ISP contains a security label, the accessing process must have an equivalent security label. With MLIPCOBJ active, requests will be failed if either the accessing process or the ISP does not contain a security label.

Format

```
CALL IRRSKI00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Requested_access_code,
               ALET, ISP,
               ALET, CREDIPC
              )
```

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 words 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.

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.

Requested_access_code
The name of a one-byte field containing the requested access. The defined codes are:

X'00' No access

X'02' Write access (or alter access)

X'04' Read access

X'06' Read and write access

ISP
The name of the IISP for the key being accessed.

CREDIPC
The name of the CREI structure for the current IPC system callable service. Use the CREI to determine the IPC identifier and IPC key being used. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSKI00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized to access the IPC mechanism.
8	8	32	CREI user type is not supported.

Usage notes

1. This service is intended for use only by the MVS BCP.
2. An audit record is optionally written, depending on the audit options in effect for the system.

   If the audit function code in the CREDIPC is AFC_WGETIPC, no audit record is written.
3. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

makeISP, R_IPC_ctl

---

ck_owner_two_files (IRRSC200): Check owner of two files

Function

The **ck_owner_two_files** service checks whether the calling process is a superuser or is the owner of either of the file/directory, or directory/directory entry pair represented by input values FSP1 and FSP2.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. A process is the owner of the file if the process's effective OS/390® UNIX user identifier (UID) is equal to the file's owner UID.

2. If the caller is not superuser nor the owner, and the audit function code is listed in Table 3, an authorization check is performed on the corresponding resource name in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

Table 3. UNIXPRIV class resource names used in ck_owner_two_files

Audit function code	Resource name	Access required
RENAME, RMDIR, UNLINK	SUPERUSER.FILESYS	CONTROL

3. If the SECLABEL class is active and the file or directory has a security label, then the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the process's current security label, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service is passed a system CRED.

Format

```
CALL IRRSC200  (Work_area,
                ALET, SAF_return_code,
                ALET, RACF_return_code,
                ALET, RACF_reason_code,
                ALET, FSP1,
                ALET, FSP2,
                ALET, File_identifier_1,
                ALET, File_identifier_2,
                ALET, CRED
               )
```

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 words 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.

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.

FSP1
The name of the IFSP for the first file, directory, or directory entry to be checked. If FSP1 is a file, FSP2 must be a directory. If FSP1 is a directory entry, FSP2 must be a directory.

FSP2
The name of the IFSP for the second file, directory, or directory to be checked. If FSP2 is a file, FSP1 must be a directory. If FSP2 is a directory entry, FSP1 must be a directory.

File_identifier_1
The name of a 16-byte area containing a unique identifier of the first file to be checked.

File_identifier_2
The name of a 16-byte area containing a unique identifier of the second file to be checked.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSC200 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized.
8	8	12	An internal error occurred during RACF processing.
8	8	32	CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

None

ck_priv (IRRSKP00): Check privilege

Function

The **ck_priv** service checks whether the calling process is a superuser.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. A superuser is a user whose process has an effective UID of 0 or has RACF trusted or privileged authority.
2. If the caller is not superuser and the audit function code is listed in Table 4, an authorization check is performed on the corresponding resource name in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

Table 4. UNIXPRIV class resource names used in ck_priv

Audit function code	Resource name	Access required
MOUNT(nosetuid), MOUNT_NA (nosetuid), UNMOUNT(nosetuid), CHMOUNT(nosetuid)	SUPERUSER.FILESYS.MOUNT	READ
MOUNT_U, MOUNT_UNA UNMOUNT_U UNMOUNT_UNA	SUPERUSER.FILESYS.USERMOUNT	READ
MOUNTSETUID, UNMOUNTSETU, CHMOUNT(setuid)	SUPERUSER.FILESYS.MOUNT	UPDATE
QUIESCE(nosetuid), UNQUIESCE(nosetuid)	SUPERUSER.FILESYS.QUIESCE	READ
QUIESCESETU, UNQUIESCESU	SUPERUSER.FILESYS.QUIESCE	UPDATE
PFSCTL	SUPERUSER.FILESYS.PFSCTL	READ
SETPRIORITY, NICE	SUPERUSER.SETPRIORITY	READ
VREGISTER	SUPERUSER.FILESYS.VREGISTER	READ
SHMMCV	SUPERUSER.SHMMCV.LIMIT	READ

Format

```
CALL IRRSKP00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Audit_function_code
              )
```

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 words 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.

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.

Audit_function_code
The name of a fullword containing a function code identifying the system call function being processed. See *z/OS Security Server RACF Data Areas* for a list of the defined codes.

Return and reason codes

IRRSKP00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The user is privileged.
4	0	0	RACF is not installed.
8	8	4	The user is not privileged.
8	8	12	An internal error occurred during RACF processing.

Usage notes

1. This service is intended for use only by the MVS BCP, a z/OS UNIX file system, and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. An audit record is written.

Related services

None

ck_process_owner (IRRSKO00): Check process owner

Function

The **ck_process_owner** service checks whether the calling process is the owner of the target process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. For request types 2, 3, and 4, IRRSKO00 checks whether the caller has superuser authority or is the owner of the target process, and returns a return and reason code indicating the result. For request type 5, if the SECLABEL class is active, IRRSKO00 also checks if the caller's security label is equivalent to the security label of the target process, unless the ACEE indicates trusted or privileged authority.
2. The caller is an owner of a process if either the real or effective z/OS UNIX user identifier (UID) of the calling process is equal to either the real or saved UID passed in the *Target_process_UIDs* parameter area.
3. If the caller is not superuser nor the process owner, and the request type is listed in Table 5, an authorization check is performed on the corresponding resource name in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

Table 5. UNIXPRIV class resource names used in ck_process_owner

Request type	Resource name	Access required
2, 5	SUPERUSER.PROCESS.KILL	READ

Table 5. UNIXPRIV class resource names used in ck_process_owner (continued)

Request type	Resource name	Access required
3	SUPERUSER.PROCESS.GETPSENT	READ

Format

```
CALL IRRSKO00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Request_type,
               ALET, Target_process_UIDs,
               ALET, Target_PID,
               ALET, Signal_code
              )
```

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 words 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.

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.

Request_type
The address of a byte containing a request type. The defined types are:

- 1 - audit-only request from kill. It is used when a SIGCONT signal is being sent to a process in the same session as the signalling process.
- 2 - kill request
- 3 - getpsent request
- 4 - open_tty request
- 5 - sigqueue request

Target_process_UIDs
For request types 1 through 4, the address of a 3-word area containing the real, effective, and saved z/OS UNIX user identifiers (UIDs) (in that order) for the target process . For request type 5, the address of a 5-word area containing the real, effective, and saved z/OS UNIX user identifiers, and the 8-byte security label for the target process.

Target_PID
The name of a fullword containing the PID of the target process.

Signal_code
The address of a word containing a code that identifies the type of signal being

sent. This code is used only for auditing. The signal code values are defined in the z/OS UNIX macro BPXYSIGH. This parameter is ignored for request type 3.

Return and reason codes

IRRSKO00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The caller is not authorized.
8	8	8	The request type is not valid.
8	8	12	An internal error occurred during RACF processing.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. An audit record is optionally written, depending on the audit options in effect for the system.

   If the request type is 3 for getpsent, then PROCACT class auditing for SETR LOGOPTIONS(FAILURES) and LOGOPTIONS(SUCCESSES) will be ignored. Use LOGOPTIONS(ALWAYS), if auditing is required, however, this may result in an excessive amount of auditing.
3. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

None

clear_setid (IRRSCS00): Clear set ID

Function

The **clear_setid** service clears the S_ISUID, S_ISGID, and S_ISVTX bits for the file passed as input.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSCS00 (Work_area,
               ALET,SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, FSP,
               ALET, File_identifier,
               ALET, CRED
              )
```

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 words 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.

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.

FSP
The name of the IFSP in which the S_ISUID, S_ISGID, and S_ISVTX bits are to be cleared.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSCS00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by an z/OS UNIX System Services file system and by z/OS UNIX System Services servers. The service contains support for z/OS UNIX System Services servers, but cannot be directly invoked by an z/OS UNIX System Services server.
2. The caller is responsible for preserving the updated IFSP.
3. If either bit was on, an audit record is optionally written.

Related services

R_chmod, R_exec

---

deleteUSP (IRRSDU00): Delete USP

Function

The **deleteUSP** service deletes the security environment for the calling process. The caller can continue as an MVS user, but is no longer an z/OS UNIX process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address

space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSDU00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
              )
```

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 words 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.

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.

Return and reason codes

IRRSDU00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.

Usage notes

1. This service is intended only for use by the MVS BCP and by z/OS UNIX System Services servers. This service can be directly invoked by an z/OS UNIX System Services server.
2. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

initUSP

getGMAP (IRRSGM00): Get GID-to-Group-Name mapping

Function

The **getGMAP** service returns the z/OS UNIX group identifier (GID) or group name corresponding to the input group name or GID, based on the setting of an input lookup flag.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSGM00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Flag,
               ALET, GID,
               ALET, group_name
              )
```

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 words 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.

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.

Flag
The name of a word containing the lookup option:

X'00000000'
search by z/OS UNIX group identifier (GID), return group name

X'00000001'
search by group name, return GID

X'00000002'
search by group name, return z/OS UNIX group identifier (GID), but do not create a new GID even if BPX.UNIQUE.USER is defined.

GID
The name of a fullword for a z/OS UNIX group identifier (GID). The GID is either input or output in this word, depending on the flag parameter.

Group_name
The name of an 8-byte area for the group name. The group name is left-justified and padded with blanks and is either input or output in the area, depending on the flag parameter.

Return and reason codes

IRRSGM00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	If search by GID: GID is not defined. If search by group name: The current group's profile has no OMVS segment.
8	8	8	The group name is not defined.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.
8	8	20	OMVS segment of the current group's profile has no GID.
8	8	24	The maximum number of file descriptors (OPEN MAX) are currently open in the calling process. **Note:** RACF does not issue this return code, but other security products may.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	28	The maximum allowable number of files is currently open in the system. **Note:** RACF does not issue this return code, but other security products may.

Usage notes

- This service is intended only for use by the MVS BCP.
- If getGMAP is given a group name and flag X'00000001' as input, and the corresponding GROUP profile has no OMVS segment, getGMAP checks for the existence of the FACILITY class profile BPX.UNIQUE.USER and, if the corresponding FACILITY class profile BPX.NEXT.USER defines a valid GID value or range, the service generates and returns a unique GID. The new GID is saved in the group profile.

  getGmap generates an SMF type 80 record (event 88) when a new GID is assigned and SETROPTS AUDIT(GROUP) is in effect.
- Check if any logrec entry has been created to ensure getGMAP service was being run successfully. Refer to *z/OS Security Server RACF Diagnosis Guide* for detailed logrec information.

Related services

None

---

get_uid_gid_supgrps (IRRSGE00): Get UIDs, GIDs, and supplemental groups

Function

The **get_uid_gid_supgrps** service gets the real, effective, and saved z/OS UNIX user identifiers (UIDs) and z/OS UNIX group identifiers (GIDs), and the supplemental groups from the USP.

Because the size of the supplemental group list varies, IRRSGE00 checks the input group count before putting supplemental GIDs in the grouplist area. See **Group_count** under "Parameters" on page 34 for more information.

The GIDs are not explicitly added to or deleted from the supplemental group list. A GID is in this list if the user was a member of the group when the user's ACEE was created through a RACROUTE REQUEST=VERIFY request and if the GID was assigned to the group before the **initUSP** service was performed for the process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSGE00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, RACF_work_area,
               ALET, User_key,
               ALET, Group_count,
               ALET, Grouplist,
               ALET, Number_of_GIDs,
               ALET, Output_UIDs,
               ALET, Output_GIDs
              )
```

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 words 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.

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.

RACF_work_area
The name of a 1024-byte work area for RACF use.

User_key
The name of a byte containing the user's key. This key is used to store into the output grouplist area. The key is in the four high-order bits of the byte.

Group_count
The name of a word containing the number of z/OS UNIX group identifier (GID) entries that can be stored in the *Grouplist* area. If *Group_count* is:

1. 0, the *Grouplist* area is not used. IRRSGE00 returns the total supplemental GID count of the current process in the *Number_of_GIDs* parameter.
2. Less than the total supplemental GID count:
   a. An error code is returned.
   b. The GIDs of the supplemental groups for the current process are put into the *Grouplist* area, which can only accommodate the number of GIDs specified in the *Group_count* parameter.
   c. The count of the supplemental GIDs actually placed in the *Grouplist* area is returned in the *Number_of_GIDs* parameter.
   d. The *Group_count* field is set to the total supplemental GID count of the current process.

      The supplemental groups in the *Grouplist* area are listed in the same order as the group connections shown in the output of the LISTUSER command.
3. Greater than or equal to the total supplemental z/OS UNIX group identifier (GID) count:
   a. The GIDs of the supplemental groups for the current process are put into the *Grouplist* area.
   b. The supplemental GID count of the current process is put into the *Number_of_GIDs* parameter.

Grouplist
The name of an area in which the GIDs of the supplemental groups for a process are returned. The *Group_count* parameter indicates the number of entries this area can contain. The GIDs are returned as consecutive 4-byte entries.

Number_of_GIDs
The name of a word in which the number of GIDs put in the *Grouplist* area is returned.

Output_UIDs
The name of a 3-word area in which, respectively, the real, effective, and saved z/OS UNIX user identifiers (UIDs) are returned.

Output_GIDs
The name of a 3-word area in which, respectively, the real, effective, and saved z/OS UNIX group identifiers (GIDs) are returned.

Return and reason codes

IRRSGE00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	4	*Group_count* is less than the number of supplemental groups (see item 2 on page 35 under the **Group_count** parameter).
8	8	8	The grouplist address is not valid.
8	8	12	An internal error occurred during RACF processing.

Usage notes

- This service is intended only for use by z/OS UNIX. The service which executes in the primary address space contains support that accesses the home address space task control block and address space control block for the requested data.
- In order to support multiple processes in one address space, this function needs to return the requested data from either the task control area or the address space control area. The task control area is accessed before the address space control area.

Related services

None.

getUMAP (IRRSUM00): Get UID-to-User-ID mapping

Function

The **getUMAP** service returns the z/OS UNIX user identifier (UID) or user ID corresponding to the input user ID or UID, based on the setting of an input lookup flag.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSUM00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Flag,
               ALET, UID,
               ALET, Userid
              )
```

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 words 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.

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.

Flag
The name of a word containing the lookup option:

X'00000000'
search by z/OS UNIX user identifier (UID), return user ID

X'00000001'
search by user ID, return z/OS UNIX user identifier (UID)

X'00000002'
search by user ID, return z/OS UNIX user identifier (UID), but do not create a new UID even if BPX.UNIQUE.USER is defined.

UID
The name of a fullword for a z/OS UNIX user identifier (UID). The UID is either input or output in this word, depending on the flag parameter.

Userid
The name of an 8-byte area for the user ID. The user ID is left-justified and padded with blanks, and is either input or output in the area depending on the flag parameter.

Return and reason codes

IRRSUM00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	If search by UID: UID is not defined. If search by user ID: The user's profile has no OMVS segment.
8	8	8	User ID is not defined.
8	8	12	An internal error occurred during RACF processing
8	8	16	Recovery could not be established.
8	8	20	The OMVS segment of the user's profile has no UID.
8	8	24	The maximum number of file descriptors (OPEN MAX) are currently open in the calling process. **Note:** RACF does not issue this return code, but other security products may.
8	8	28	The maximum allowable number of files is currently open in the system. **Note:** RACF does not issue this return code, but other security products may.

Usage notes

- This service is intended only for use by the MVS BCP.
- If getUMAP is given a user ID and flag X'00000001' as input, and the corresponding USER profile has no OMVS segment, getUMAP checks for the existence of the FACILITY class profile BPX.UNIQUE.USER and, if the corresponding FACILITY class profile BPX.NEXT.USER defines a valid UID value or range, the service generates and returns a unique UID. The new UID is saved in the user profile, along with any OMVS field information copied from the profile of a user if specified in the application data field of the BPX.UNIQUE.USER profile.

  getUmap generates an SMF type 80 record (event 88) when a new UID is assigned and SETROPTS AUDIT(USER) is in effect.
- Check if any logrec entry has been created to ensure getUMAP service was being run successfully. Refer to *z/OS Security Server RACF Diagnosis Guide* for detailed logrec information.

Related services

None.

initACEE (IRRSIA00): Initialize ACEE

Function

The **initACEE** service provides an interface for creating and managing RACF security contexts through the z/OS UNIX System Services pthread_security_np

service, __login service, or by other MVS server address spaces that do not use z/OS UNIX services. This service also provides an interface for registering and deregistering certificates through the z/OS UNIX System Services __security service. It also provides an interface for querying a certificate to determine if it is associated with a user ID.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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

1. If the function_code indicates that a certificate is to be registered or deregistered, initACEE will perform the following authority checks:
   - To register a certificate with the current user ID, the caller must be RACF SPECIAL or have at least READ authority to the IRR.DIGTCERT.ADD resource in the FACILITY class.
   - To deregister a certificate with the current user ID, the caller must be RACF SPECIAL or have at least READ authority to the IRR.DIGTCERT.DELETE resource in the FACILITY class.
   - To register a certificate as a CERTAUTH certificate, the caller must be RACF SPECIAL or have at least CONTROL authority to the IRR.DIGTCERT.ADD resource in the FACILITY class.
2. If the function_code indicates that an ACEE is to be created or a certificate is to be queried and the service determines that the user ID to use is specified in the hostIdMappings extension of the input certificate, the caller's authority to the

IRR.HOST.*(host-name)* resource in the SERVAUTH class is checked. (The value for *host-name* is specified in the hostIdMappings extension.) The resource must exist and the caller must have READ authority to it, otherwise the extension is ignored.

Note: To determine the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.

Format

```
CALL IRRSIA00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
                     Function_code,
                     Attributes,
                     RACF_userid,
                     ACEE_ptr,
                     APPL_id,
                     Password,
                     Logstring,
                     Certificate,
                     ENVR_in,
                     ENVR_out,
                     Output_area,
                     X500 name,
                     Variable_list,
                     Security_label,
                     SERVAUTH_name,
                     Password_phrase,
                     IDID_area
                     )
```

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 ALET can be different. The words 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.

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.

Function_code
The name of a 1-byte area containing the function code.

X'01' Create an ACEE.

X'02' Delete an ACEE.

X'03' Purge all managed ACEEs.

X'04' Register a certificate

X'05' Deregister a certificate

X'06' Query a certificate

Attributes
The name of a 4-byte area containing information about the function to be performed. Zero or more attributes can be set. (See Table 7 on page 45 for the values allowed for the Attributes parameter.)

RACF_userid
The name of a 9-byte area that consists of a 1-byte length field followed by up to 8 characters. It must be specified in uppercase. If not specified, the length must equal 0.

ACEE_ptr
The name of a 4-byte area that contains the ACEE address.

APPL_id
The name of a 9-byte area that consists of a 1-byte length field followed by the name of the application to be used if verifying the user's authority to access the application. This saves the application from having to do a separate authorization check. When using certificate mapping profiles, the application name is also used as part of the additional criteria in determining a user ID when a certificate is passed to initACEE. It must be specified in uppercase. If not specified, the length must equal zero.

Password
The name of a 9-byte area that consists of a 1-byte length field followed by the password or PassTicket provided by the user. If not specified, the length must equal zero.

Logstring
The name of an area that consists of a 1-byte length field followed by character data to be written to the system-management-facilities (SMF) data set, together with any RACF audit information, if logged. If not specified, the length must equal zero.

Certificate
The name of an area that consists of a 4-byte length field followed by a digital certificate. If not specified, the length must equal 0; or the end of the parameter list must be indicated by the setting of the high order bit in the address of the previous parameter. The certificate must be a single DER encoded X.509 certificate. For the registration and deregistration functions, PKCS #7, PEM, or Base64 encoded certificates are also allowed.

ENVR_in
The name of the data structure that contains the information necessary to re-create a security environment. The data structure must have the format shown in Table 8 on page 45. See the ENVR_out parameter for additional information about this data structure and the ENVR object to which it points. The structure must reside on a doubleword boundary.

While the format of the data structure pointed to by ENVR_in is known to the initACEE invokers, the content of the object itself is determined by the external security product.

The input for this parameter can be the output from a previous initACEE with the ENVR_out parameter specified, or from RACROUTE REQUEST=VERIFY or REQUEST=EXTRACT, with the ENVROUT parameter specified.

If ENVR_in is not specified, the ENVR object length must equal 0, or the end of the parameter list must be indicated by the setting of the high order bit in the address of a previous parameter. ENVR_in should not be specified when requesting that an ENVR object be returned (INTA_ENVR_RET).

For more information about the ENVR data structure, see *z/OS Security Server RACROUTE Macro Reference*.

ENVR_out
The name of the data structure to contain the security environment that was just created. The data structure must have the format shown in Table 8 on page 45. This data structure describes the storage location for the ENVR object that is created as part of this initACEE create request.

While the format of the data structure pointed to by ENVR_out itself is known to the initACEE invokers, the content of the object itself is determined by the external security product.

The ENVR object storage area can be supplied by the caller or obtained by RACF. If supplied by the caller, it must be on a doubleword boundary and be associated with the job step task. If RACF obtains the storage area, it is on a doubleword boundary and is associated with the job step task. The storage is allocated based on the mode of the caller (LOC=ANY for 31-bit callers and LOC=24 for 24-bit callers).

Storage for the ENVR object is obtained and freed in the subpool and key specified by the caller in the ENVR_out data structure. For additional details on specifying the ENVR object storage area length and address, see Table 9 on page 46.

Since the ENVR object length is returned to the caller, the ENVR object can be moved from one storage area to another. It is intended for use on subsequent initACEEs with the ENVR_in parameter, or on RACROUTE REQUEST=VERIFY with the ENVRIN parameter, as input when rebuilding a user's security environment. It should not be saved for a long period or passed to another system that does not share the same RACF database.

If the Attributes parameter indicates that an ENVR object should be returned (INTA_ENVR_RET), then this parameter must be specified with at a minimum the values for the subpool and key fields.

For more information about the ENVR data structure, see *z/OS Security Server RACROUTE Macro Reference*.

Output_area
The name of a fullword in which the service routine stores the address of an area containing data about the user. The output area is obtained in the primary address space, in subpool 229, and must be freed by the caller of initACEE. The following data is returned; the area returned is mapped by macro IRRPOUSP (OUSP):

- TSO user ID
- z/OS UNIX user identifier (UID) of user
- z/OS UNIX group identifier (GID) of current group
- Home directory path name
- Initial program path name
- User limits (when OUSP version is greater than 0)

If the Attributes parameter indicates that an OUSP should be returned (INTA_USP and INTA_OUSP_RET), then this parameter must be specified. If

the Attributes do not indicate that an OUSP should be returned, then the fullword must equal 0, or the end of the parameter list must be indicated by the setting of the high order bit in the address of a previous parameter.

X500name
The name of a fullword in which the service routine stores the address of the X500 name pair data structure if the function code indicates a certificate is being queried, and the attributes indicate that an X500 name pair should be returned. The X500 name pair data structure is obtained in the primary address space, in subpool 229, and must be freed by the caller of initACEE.

If the function code indicates that an ACEE is to be created, and the RACF_userid parameter is specified, X500name can supply the name of a fullword containing the address of the X500 name pair to be associated with the ACEE. The X500 name pair should previously have been obtained, along with the RACF user ID, by querying a certificate using initACEE. Both the issuer's name and subject's name must be supplied, and the length of each must be in the range 1 to 255 to prevent a parameter list error. If a valid X500 name pair is supplied, the ACEE created will point to a copy of the name pair, and it will subsequently be used in auditing.

Variable_list
The name of the data structure that contains the additional criteria to be used to determine the user ID associated with the certificate supplied to initACEE. The variable list data structure is a 4-byte number of value entries, followed by that number of entries. Each value entry consists of an 8-byte value name, a 4-byte value length, and the value. The value name must be padded on the right with blanks if it is less than 8-bytes. The value length must be in the range of 1 to 255. If it is outside of this range, a parameter list error will result. A maximum of 10 values may be specified. If the number of values is greater than 10, a parameter list error will result.

Variable names should be meaningful to the caller of initACEE. Making the 3 character prefix associated with the product calling initACEE part of the variable name will ensure that it is unique. For example, assume RACF implemented a server that calls initACEE for its clients. It will pass a variable, IRRSLVL, which has 2 values. The values are LOW and HIGH. LOW is if the user is accessing the server from the internet and HIGH is if the user is accessing the server from the intranet. The variable_list containing the variable name and its value, LOW or HIGH, is passed to initACEE, along with the certificate supplied by the user. The value of the variable will be used as additional criteria in selecting which user ID the certificate maps to. All callers of initACEE should document their variable names, and the values they pass for each name, in their product documentation.

All value names and values should be uppercase. Do not specify the APPLID or SYSID criteria values in the variable_list. These are determined from the APPL_id parameter and MVS control; blocks, respectively. If they are specified in the variable_list, that specification will be ignored.

This parameter is ignored unless the certificate parameter is specified, and the function code indicates that an ACEE is to be created, or that the certificate is to be queried to find a user ID. If the certificate is defined to RACF in the DIGTCERT class, additional criteria will not be used, and the variable_list values will be ignored. If the certificate is not defined in the DIGTCERT class, the values in the variable list will be used along with APPL_id and SYSID to look for an associated user ID using the DIGTNMAP and DIGTCRIT classes if these classes are active and have been RACLISTed with SETROPTS.

Security_label
The name of a 9-byte area that consists of a 1-byte length field followed by the name of the security label that defines the security classification of the environment to be created. It must be specified in uppercase. If not specified, the length must be zero.

SERVAUTH_name
The name of an area that consists of a 1-byte length field followed by the name of a resource in the SERVAUTH class to be used if verifying the user's authority to access this server. This resource is the network access security zone name that contains the IP address of the user. It must be specified in uppercase, and cannot exceed 64 bytes in length. If not specified, the length must be zero.

Password_phrase
The name of an area that consists of a 1-byte length field followed by the password phrase provided by the user. If the length is not specified, it must equal zero. If the length is specified, the length field must be 9-100 characters to prevent a parameter list error.

IDID_area
The name of a fullword containing the address of a distributed identity data structure (IDID). If the function code parameter indicates that an ACEE is to be created, no user ID parameter is specified, and the IDIDMAP class is active and RACLISTed, information in the IDID is used to determine a RACF user ID. If the IDIDMAP class is inactive or the information in the IDID does not map to a RACF user ID, the initACEE service will fail. If a RACF_userid parameter is specified on the initACEE call, the IDID_area parameter can supply the name of an IDID to be associated with the ACEE; the IDIDMAP class is not used. When the IDID_area parameter is specified, the distributed identity information in the IDID should have been previously authenticated. If an IDID is supplied and an ACEE is successfully created, the ACEE will point to a copy of the IDID, and it will subsequently be used in auditing.

Table 6. Parameter usage

Parameter	Create ACEE	Delete ACEE	Purge ACEE	Reg/Dereg certificate	Query certificate
SAF_return_code	Output	Output	Output	Output	Output
RACF_return_code	Output	Output	Output	Output	Output
RACF_reason_code	Output	Output	Output	Output	Output
Function_code	Input	Input	Input	Input	Input
Attributes	Input	Input	n/a	n/a	Input
RACF_userid	Input	n/a	n/a	n/a	Output
ACEE_ptr	Output	Input	n/a	n/a	n/a
APPL_id	Input	n/a	n/a	n/a	n/a
Password	Input	n/a	n/a	n/a	n/a
Logstring	Input	n/a	n/a	n/a	n/a
Certificate	Input	n/a	n/a	Input	Input
ENVR_in	Input	n/a	n/a	n/a	n/a
ENVR_out	Input/ Output See Table 8 on page 45	n/a	n/a	n/a	n/a
Output_area	Output	n/a	n/a	n/a	n/a

Table 6. Parameter usage (continued)

Parameter	Create ACEE	Delete ACEE	Purge ACEE	Reg/Dereg certificate	Query certificate
X500name	Input	n/a	n/a	n/a	Output
Variable_list	Input	n/a	n/a	n/a	Input
Security_label	Input	n/a	n/a	n/a	n/a
SERVAUTH_name	Input	n/a	n/a	n/a	n/a
Password_phrase	Input	n/a	n/a	n/a	n/a
IDID_area	Input	n/a	n/a	n/a	n/a

Table 7. Values allowed for attributes parameter

Parameter	Value	Action to be taken
INTA_MANAGED	X'80000000'	Create an ACEE for the user ID that is cached by RACF.
INTA_USP	X'40000000'	Create a USP for the user ID
INTA_TASK_LVL	X'20000000'	For create function code, create an ACEE and attach to the current TCB. For delete function code, delete the ACEE attached to the current TCB.
INTA_UNAUTH_CLNT	X'10000000'	Create an ACEE for an unauthenticated client.
INTA_AUTH_CLNT	X'08000000'	Create an ACEE for an authenticated client.
INTA_MSG_SUPP	X'04000000'	Suppress RACF messages produced as a result of creating a user's security context.
INTA_ENVR_RET	X'02000000'	Return an ENVR object for the ACEE created by this request.
INTA_NO_TIMEOUT	X'01000000'	Create a managed ACEE that does not time out. When this bit and INTA_MANAGED are set for the creation of a new managed ACEE, the ACEE is cached and does not expire after 5 minutes.
INTA_OUSP_RET	X'00800000'	Return an OUSP in the output area
INTA_X500_RET	X'00400000'	Return an X500 name pair

Table 8. ENVR data structure

Description	Length (Bytes)	ENVR_out Usage	ENVR_in Usage
ENVR object length	4	Output	Input
ENVR object storage area length	4	Input/Output (see Table 9 on page 46)	Input
ENVR object storage area address	4	Input/Output (see Table 9 on page 46)	Input
ENVR object storage area subpool	1	Input	n/a

Table 8. ENVR data structure (continued)

Description	Length (Bytes)	ENVR_out Usage	ENVR_in Usage
ENVR object storage area key	1	Input	n/a

Table 9. ENVR_out storage area processing

ENVR object storage area length	ENVR object storage area address	Result
Zero	Any value	RACF obtains storage size needed to contain ENVR object and sets ENVR object storage area length and address fields.
Nonzero	Zero	RACF obtains storage size specified or minimum needed to contain ENVR object and sets ENVR object storage area length and address fields.
Nonzero	Nonzero	RACF uses the area provided if large enough to contain ENVR object. If too small, RACF freemains the area, obtains a larger area, and sets ENVR object storage area length and address fields.

Table 10. X500 name pair data structure

Offset	Length (Bytes)	Description
0	4	Length of name pair data structure
4	2	Length of issuer's name (1 to 255)
6	2	Length of subject's name (1 to 255)
8	1 to 255	Issuer's distinguished name
*	1 to 255	Subject's distinguished name

Table 11. Variable list data structure

Offset	Length (Bytes)	Description
0	4	Number of value entries
4	8	Value name
12(C)	4	Value length (1 to 255)
16(10)	1 to 255	Value

Return and reason codes

IRRSIA00 returns the following values in the reason and return code parameters:

Table 12. initACEE create return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing.
8	8	12	Recovery environment could not be established.
8	8	16	User ID is not defined to RACF.
8	8	20	Password, Password Phrase or Pass Ticket is not valid.
8	8	24	Password or Password Phrase is expired.
8	8	28	User ID is revoked or user access to group is revoked.
8	8	32	The user does not have appropriate RACF access to either the SECLABEL, SERVAUTH profile, or APPL specified in the parmlist.
8	8	36	Certificate is not valid.
8	8	40	Either no user ID or userid mapping is defined for this certificate or the status of the certificate or mapping is NOTRUST, or there are no mapping profiles associated with the certificate. Mapping profiles can either be defined as a Certificate Name Filtering profile or as SERVAUTH profiles that are used for HostID Mappings. See Usage Note number 37.
8	8	44	The client security label is not equivalent to the server's security label.
8	8	48	A managed ACEE is requested with a nested RACO in the Envir_In parameter.
8	12	InitUSP reason code	initUSP failed. See initUSP reason codes in “Return and reason codes” on page 57.

Table 13. initACEE delete return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing.
8	8	12	Recovery environment could not be established.

Table 13. initACEE delete return codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	16	An attempt was made to delete the server address space ACEE before invoking initACEE to purge all managed ACEEs.

Table 14. initACEE purge return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing.
8	8	12	Recovery environment could not be established.
8	8	16	There are managed ACEEs that are still in use.

Table 15. initACEE register and deregister return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing
8	8	12	Recovery environment could not be established.
8	8	16	The user is not authorized.
8	8	20	The certificate does not meet RACF requirements.
8	8	24	The certificate is defined for another user.
8	8	28	The certificate can not be deregistered because it has been used to generate a request through RACDCERT GENREQ.
8	8	32	RESERVED
8	8	36	The certificate is not valid.

Table 16. initACEE query return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.

Table 16. initACEE query return codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing.
8	8	12	Recovery environment could not be established.
8	8	16	RESERVED
8	8	20	RESERVED
8	8	24	RESERVED
8	8	28	RESERVED
8	8	32	RESERVED
8	8	36	This certificate is not valid.
8	8	40	No user ID is defined for this certificate.

Usage notes

1. This service is only intended for use by the z/OS UNIX kernel or by other MVS servers that do not use z/OS UNIX.
2. This service can only be used by supervisor state callers.
3. An ALET must be specified for the SAF_return_code, RACF_return_code, and RACF_reason_code parameters.
4. When ACEEs are created by initACEE, the following information is used on the RACROUTE REQUEST=VERIFY:
   - Password, if verifying a password
   - Appl_id, if verifying authority to an application
   - Logstring, if any audit records are created as a result of authenticating the user ID
   - LOC=ANY. If the caller is running in 31-bit address mode, the ACEE may be allocated above the 16MB line.
   - Subpool of the address space ACEE is used for the SUBPOOL keyword
   - ENVROUT, if an ENVR object data structure address was supplied by the ENVR_out parameter.
   - X500name, if the RACF_userid and X500_name parameters were specified, or if a certificate was provided as input and an associated user ID was found using the DIGTNMAP class profiles.
   - SECLABEL, if the security environment should be set up with a certain security classification.
   - SERVAUTH, if verifying authority to a resource in the SERVAUTH class.
   - PHRASE, if verifying a password phrase.
   - IDID, if the RACF_userid and the IDID_area parameters were specified, or if an IDID_area was provided as input and an associated user ID was found using the IDIDMAP class profiles.

5. When creating an ACEE, statistics are updated on the first request per day for each user ID.
6. Audit records are written only in the following situations:
   a. An ACEE is to be created and a password has been specified that is not the user's current password, or a password phrase has been specified that is not the user's current password phrase.
   b. An ACEE is to be created and a PassTicket has been specified that does not evaluate.
   c. An ACEE is to be created and the user ID has been revoked.
   d. A certificate is to be registered, and the user is not authorized to the FACILITY class resource IRR.DIGTCERT.ADD.
   e. A certificate is to be deregistered and the user is not authorized to the FACILITY class resource IRR.DIGTCERT.DELETE.
   f. A certificate is successfully registered or deregistered, and SETROPTS AUDIT(USER) is in effect, or UAUDIT is in effect for the user, or the user has SPECIAL authority and SETROPTS SAUDIT is in effect.
   g. An ACEE is to be created and a certificate has been specified that does not correspond to a RACF user ID.
   h. An ACEE is to be created and a certificate has been specified that is not trusted.
7. If an ACEE is to be anchored off the current TCB, then the INTA_TASK_LVL attribute must be set. Any value passed in ACEE_ptr is ignored, and the ACEE address is not returned. If an ACEE address is to be returned, the INTA_TASK_LVL attribute must be off. This results in the ACEE address being returned in the ACEE_ptr parameter area.
8. If an ACEE is to be deleted from the current TCB, then the INTA_TASK_LVL attribute must be set. If this is not done, the ACEE_ptr parameter must point to the address of the ACEE to be deleted.
9. If the function_code and attributes indicate that an ACEE is to be created and anchored off the TCB and there is an ACEE already anchored off the TCB, the caller receives a parameter list error.
10. If the function_code and attributes indicate that an ACEE is to be deleted from the TCB and there is no ACEE anchored to the TCB, the caller receives a parameter list error.
11. 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. The first parameter that can have the high-order bit on, ending the parameter list, is the logstring parameter
12. When the application is terminating and there are no tasks outstanding, initACEE should be called to purge all the managed ACEEs. Then the ACEE for the application server address space can be deleted.
13. The RACROUTE service should not be used to delete the managed ACEEs.
14. You can find parameter usages in Table 6 on page 44.
15. The service serializes resources at the address space level with a STEP ENQ on QNAME "SYSZRACF".
16. If the function_code indicates that an ACEE is to be created and the length of the certificate parameter is not zero, the length of the RACF_user ID, IDID_area, password phrase and password must all be 0. If a RACF_user ID, IDID_area, password phrase or password is supplied with the certificate, the caller receives a parameter list error.

17. If the function_code indicates that an ACEE is to be deleted or that managed ACEEs should be purged, and the length of the certificate parameter is not zero, then the caller receives a parameter list error.
18. If the function_code indicates that a certificate is to be registered, deregistered, or queried, and the length of the certificate parameter is zero, then the caller receives a parameter list error.
19. The certificate supplied by the certificate parameter is used only to identify a RACF user ID. It is expected that the certificate was previously verified. Note the following additional details regarding initACEE's certificate processing:
    a. All fields as defined for X.509 version 1 certificates must be present and non-null.
    b. X.509 certificates with version numbers greater than 3 are not supported.
    c. Version 3 certificates with critical extensions are not supported. Noncritical extensions are ignored.
    d. Subject and issuer names can contain only the following string types:
       - T61STRING- TAG 20
       - PRINTABLESTRING- TAG 19
       - IA5STRING- TAG 22
       - VISIBLESTRING- TAG 26
       - GENERALSTRING- TAG 27
       - BMPString-TAG 30
       - UTF8-TAG 12
    e. The length of the serial number plus the length of the issuer's name cannot exceed 245.
    f. No date validity check is performed on the certificate.
    g. No signature check is performed on the certificate.

    If the function_code indicates that an ACEE is to be created, or that a certificate is to be queried, the certificate must be a single DER encoded X.509 certificate.

    If the function_code indicates that a certificate is to be registered or deregistered, it must be in one of the following formats:
    a. A single DER encoded X.509 certificate.
    b. A Privacy Enhanced Mail (PEM) encoded X.509 certificate. If the input is in this format, only the Originator Certificate is used.
    c. One or more X.509 certificates contained within a PKCS #7 DER encoding. If the input is in this format, only the first certificate in the PKCS #7 encoding will be used.
    d. A Base64 encoded X.509 certificate as returned from a PKCS #10 certificate request. The data must include the string

       ```
       '-----BEGIN CERTIFICATE-----'
       ```

       immediately prior to the Base64 encoding, and the string

       ```
       '-----END CERTIFICATE-----'
       ```

       immediately following.

    If transmitted from an ASCII system, PEM and Base64 encoded certificates must be translated from ASCII to EBCDIC before being passed to initACEE.

20. If the function_code indicates that a certificate is to be queried, the caller is expected to supply a 9-byte area for the RACF_userid parameter. If a user ID is associated with the certificate, initACEE updates this area with the length and value of the user ID.
21. If the function_code indicates that an ACEE is to be created or that a certificate is to be queried, and the certificate supplied by the caller is defined to RACF with a status of NOTRUST, initACEE will return a RACF return code 8 and a RACF reason code 40, indicating that no user ID is defined to use this certificate.
22. If the function_code and attributes indicate that an ACEE is to be created and an ENVR object is to be returned, then the ENVR_out parameter must point to a data structure for the ENVR object. The caller receives a parameter list error if the high order bit of a previous parameter indicates the end of the parameter list.
23. If the attributes indicate that an ENVR object is to be returned, it is the caller's responsibility to free the ENVR object storage. The caller should check the storage area length and address to determine if storage needs to be freed, not the initACEE return code. In some cases, an error may be encountered after creation of the ENVR object, resulting in a non-zero return code. The caller is still responsible for freeing the ENVR object in these cases.
24. If the function_code indicates that an ACEE is to be created, and the ENVR_in parameter points to an ENVR object data structure, the length of the RACF_userid, IDID_area, password phrase, password, and certificate parameters must all be 0. The caller receives a parameter list error if a RACF_userid, IDID_area, password phrase, password, or certificate is supplied with the ENVR_in parameter.
25. If the function_code indicates that an ACEE is to be deleted or that managed ACEEs should be purged, and the ENVR_in or ENVR_out parameter is specified, then the caller receives a parameter list error.
26. When an ENVR object is supplied with the ENVR_in parameter and an ACEE creation is requested, the attribute bits that affect the ACEE (INTA_UNAUTH_CLIENT, INTA_AUTH_CLIENT, INTA_NO_TIMEOUT) and the application name (APPL_id) are ignored.
27. An OUSP can only be returned if the Attributes parameter also indicates that a USP should be created (INTA_OUSP_RET should only be on if INTA_USP is on). If an OUSP is requested without a USP, then the caller receives a parameter list error.
28. If the Attributes parameter indicates that an OUSP should be returned (INTA_OUSP_RET), then the Output_area parameter must be specified. If it is not, the caller receives a parameter list error.
29. When the INTA_NO_TIMEOUT bit and the INTA_MANAGED bit are set for the creation of a new managed ACEE, the ACEE is cached and does not expire after five minutes.
30. If the Attributes parameter indicates that a no timeout ACEE is requested (INTA_NO_TIMEOUT) and a managed ACEE with an expiration time is found in the cache that satisfies the request, the address of the managed ACEE is returned. The expiration time of the ACEE in the cache remains the same. After receiving a subsequent delete request, the ACEE may expire.
31. If the function indicates that a certificate is being queried, and the Attributes parameter indicates that an X500 name pair should be returned (INTA_X500_RET), then both the DIGTCERT and DIGTNMAP profiles will be checked for a user ID associated with the certificate. If the function indicates that a certificate is being queried, and the X500 name pair has not been

requested, then only the DIGTCERT profiles will be checked to determine if the certificate has been defined to RACF, and associated with a user ID.

32. If the Attributes parameter indicates that an X500 name pair should be returned (INTA_X500_RET), then the X500name parameter must be specified. If it is not, the caller receives a parameter list error.
33. If a certificate is being queried, and an X500 name pair has been requested, DIGTNMAP profiles may be used to determine the RACF user ID. If there are additional criteria associated with the DIGTNMAP profile, the APPL_id and Variable_list parameters, as well as the system-identifier of the system initACEE is running on, are used in determining the RACF user ID. If the certificate supplied for the query will later be used to create an ACEE, and the same user ID is expected to result, then the additional criteria must be the same for the query and create functions. In other words, the APPL_id and Variable_list parameter specifications must be the same, and the query and create must be run on the same system.
34. When the ENVR_in parameter is specified, the INTA_USP attribute is ignored. If the ENVR_in contains a USP, the resulting ACEE will also have a USP associated with it. The X500name parameter will also be ignored. If the ENVR_in contains an X500 name, the resulting ACEE will also have an X500 name associated with it. The information needed to create an OUSP is not available in an ENVR_object. If the INTA_RET_OUSP attribute is set indicating that an OUSP should be returned, and an ENVR object is supplied with the ENVR_in parameter, the caller receives a parameter list error.
35. If the function_code indicates that an ACEE is to be created or that a certificate is to be queried and processing determines that the user ID to be used is to be extracted from the hostIdMappings certificate extension, InitACEE will ignore the extension under the following conditions:
    - The caller does not have READ authority (or greater) to any SERVAUTH class resource identified by the hostName(s) in the extension.
    - The user ID extracted has a length less than 1 or greater than 8.
    - For create only, the user ID is not a RACF defined user.
    - The definition of the hostIdMappings extension in ASN.1 syntax is:

```
id-ce-hostIdMappings OBJECT IDENTIFIER::= {1 3 18 0 2 18 1}

HostIdMappings::= SET OF HostIdMapping

HostIdMapping::= SEQUENCE{
   hostName        IMPLICIT[1] IA5String,
   subjectId         IMPLICIT[2] IA5String,
   proofOfIdPossession IdProof OPTIONAL
 }
 IdProof::= SEQUENCE{
   secret        OCTET STRING,
   encryptionAlgorithm OBJECT IDENTIFIER
 }
```

36. If the function_code indicates that a certificate is to be registered, and one of the CERTAUTH certificates meets the following three conditions, the input certificate is treated as a certificate authority certificate to be registered as CERTAUTH:
    - The CERTAUTH certificate's public key matches that of the input certificate.
    - The CERTAUTH certificate's subject distinguished name matches that of the input certificate and
    - The CERTAUTH certificate has a private key.

Otherwise, the input certificate is treated as an end-user certificate to be registered with the current user ID.

37. When a return code of 8 8 40 is received from an initACEE Create, other occurrences could be if the DIGTCERT, DIGTNMAP, or DIGTCRIT class has not been processed using SETROPTS RACLIST, or if SETROPTS RACLIST was used, but the class was not RACLIST REFRESHed after the certificate or mapping was added or altered. See *z/OS Security Server RACF Command Language Reference* for information about the RACDCERT and SETROPTS commands.
38. If the Function_code indicates an ACEE creation, and the seclabel class is active, the security_label parameter should be implicitily specified. If it's not specified, and the address space ACEE contains a label other than SYSMULTI, then that label will be used for the new ACEE.

    If the value of the existing address space ACEE is SYSMULTI or no security label is available, a label from the profile protecting the SERVAUTH resource will be used and will override the user's default security label.
39. If the Function_code indicates that an ACEE is to be created and anchored in the TCB, and the ASXB also contains the address of an ACEE, the security labels associated with the ACEEs must be equivalent if the MLACTIVE option is in effect.
40. InitACEE will accept a nested ENVR object in the ENVR_in parameter, unless a managed ACEE is requested. A parameter list error occurs when a nested ENVR object is passed in.
41. If the function_code indicates that an ACEE is to be created, and both the password and password phrase parameters are specified, only the password phrase will be used for authentication.
42. If the Function_code indicates that an ACEE is to be created, and an IDID_area is provided without a RACF_userid, a parameter list error will occur if a password phrase or password is specified.
43. If the Function_code indicates that an ACEE is to be created, and an IDID_area is provided, a parameter list error will occur if the IDID has a length less than its header size.
44. If the function code parameter indicates that an ACEE is to be created, an IDID_area is provided, no user ID parameter is specified, and the IDIDMAP class is active and RACLISTed, InitACEE attempts to map to a RACF user ID using the distributed identity information in the IDID and mapping filters previously defined using the RACMAP command.

    During the mapping process the following operations are performed on a copy of the data (the original data is not modified):

    - All leading and trailing blanks (x'20'), nulls (x'00'), or combination of blanks and null characters are removed from the distributed identity information strings in the IDID, and the lengths are appropriately adjusted.
    - If the distributed-identity-user-name (user name) is in X.500 format the name is normalized before it is used to find the matching RACF user ID that is associated with the distributed identity filter.

    The normalization rules are described in detail under RACMAP MAP.

Related services

None

initUSP (IRRSIU00): Initialize USP

Function

The **initUSP** service verifies that the user is authorized to use z/OS UNIX and, if so, establishes security attributes for the calling process. The **initUSP** service also returns any z/OS UNIX limits that have been set on a user basis.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSIU00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Output_area
              )
```

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 and start on a word boundary.

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 words 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.

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.

Output_area
The name of a fullword in which the service routine stores the address of an area containing data about the user. IRRSIU00 uses the high-order bit of this fullword to determine if z/OS UNIX requested default processing for failures that occur under certain circumstances (as described in Note 1). If the high-order bit is on, an initUSP that would normally fail because of missing information completes successfully and builds a default USP. With current default processing (described for the getUMAP and getGMAP callable services and in usage notes for this service) even without the high-order bit turned on under the circumstances described in Notes 4 and 5, an initUSP that would previously fail is now completed successfully.

For all successful initUSP requests, the output area is obtained in the primary address space and must be freed by the caller of initUSP. The following data is returned:

- TSO/E user ID
- z/OS UNIX user identifier (UID) of user
- z/OS UNIX group identifier (GID) of current group
- Home directory path name
- Initial program path name
- User limits (when OUSP version is greater than 0)

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

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

Constant name	Value
OUSP_CPUTimeMax	1
OUSP_ASSizeMax	2
OUSP_FileProcMax	3
OUSP_ProcUserMax	4
OUSP_ThreadsMax	5
OUSP_MMapAreaMax	6
OUSP_Memlimit	7
OUSP_ShmemMax	8

The length of a limit entry in the limits array will continue to be 5 bytes. However, when the limit type is 7 (OUSP_Memlimit) or 8 (OUSP_ShmemMax) the format of that entry will be different (see below). In this case, the returned limit value will be a 3-byte value, rather than 4 bytes, and a new 1-byte units identifier will be provided. The units specification will be set to:

M megabytes

G gigabytes

T terabytes

P petabyte

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0		5	OUSP_LIMIT_ENTRY	array entry
0	0	UNSIGNED	1	OUSP_LIMITKEY	Key defining type of limit
1	1	UNSIGNED	3	OUSP_LIMITVALUE	Value of limit
4	4	CHAR	1	OUSP_UNITS	units -- M, G, T, P (megabytes, gigabytes, terabytes, or petabytes)

Return and reason codes

IRRSIU00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Current user is not defined to RACF.
8	8	8	Current group has no OMVS segment.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.
8	8	20	The user's profile has no OMVS segment.
8	8	24	The OMVS segment in the user's profile has no UID.
8	8	28	The OMVS segment in the current group's profile has no GID.

Usage notes

1. This service is intended only for use by z/OS UNIX servers and the MVS BCP. It can be directly invoked by a z/OS UNIX server. The high-order bit of the output_area is set on by the caller when initUSP is called to establish the security attributes for critical z/OS UNIX address spaces such as the kernel. When the bit is on, initUSP builds a default z/OS UNIX security environment in certain cases when it would normally fail. InitUSP sets a SAF return code of 0, RACF return code of 0, RACF reason code of 0, and builds a default USP for the following cases:
   - The user ID is not defined to RACF
   - There is no OMVS segment in the user's profile
   - There is no UID in the OMVS segment of the user's profile
   - There is no GID in the OMVS segment of the current group's profile

   The default USP returned to the caller (mapped by IRRPOUSP) contains a z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) of 0.

The lengths for initial program and home directory path names is 0. If the user ID is defined to RACF, the user ID is returned as an TSO/E user ID. If the user ID is not defined to RACF, the TSO/E user ID is set to an asterisk in the returned USP.

2. The address space or task must have an ACEE when this service is called.
3. A RACF user can be connected to more than NGROUPS_MAX groups, but only up to the first NGROUPS_MAX groups will be associated with the process when the process is created.

   The first NGROUPS_MAX z/OS UNIX groups to which a user is connected (as shown by a LISTUSER command) get associated with the process.
4. If no OMVS segment is found in the user's profile, the initUSP service checks for the existence of the FACILITY class profile BPX.UNIQUE.USER and, if the corresponding FACILITY class profile BPX.NEXT.USER defines a valid UID value or range, the service generates and stores a unique UID in the USP. If the application data field of the BPX.UNIQUE.USER profile contains the name of a user, initUSP also copies any other OMVS field information from that user profile into the USP. This information along with the new UID is saved in the new OMVS segment for the current user.
5. If no OMVS segment is found in the group profile of the user's current connect group, the initUSP service checks for the existence of the FACILITY class profile BPX.UNIQUE.USER and, if the corresponding FACILITY class profile BPX.NEXT.USER defines a valid GID value or range, the service generates and stores a unique GID in the USP. The new GID is saved in the new OMVS segment for the current group.

   See *z/OS Security Server RACF Security Administrator's Guide* for information about APPLDATA in the FACILITY class.

Related services

deleteUSP

---

makeFSP (IRRSMF00): Make IFSP

Function

The **makeFSP** service builds an IFSP in the area provided by the caller.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
PASN = HASN or PASN not = HASN

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSMF00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Mode,
               ALET, Output_FSP,
               ALET, Owning_directory_FSP,
               ALET, File_Identifier,
               ALET, CRED
              )
```

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 words 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.

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.

Mode
The name of a word containing the mode values (the filetype, the permission bits, and the S_ISUID, S_ISGID, and S_ISVTX bits) to be set for the file.

See “File type and file mode values” on page 4 for a definition of the security bits in the mode parameter.

Output_FSP
The name of a 64-byte area in which the new IFSP is built.

Owning_directory_FSP
The name of an area containing the IFSP for the owning directory.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSMF00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	12	An internal error occurred during RACF processing.
8	8	32	CRED user type is not supported.
8	12	4	A model Access Control List (ACL) exists for the parent, but no buffer address was provided for the new object's access ACL in the CredAccAcl field.
8	12	8	The buffer provided (in the CredAccAclLen field) for the new object's access ACL is not large enough. It must be at least as large as the size of the parent's directory model ACL or file model ACL (in the FACL_Len field), as appropriate to the type of object being created.
8	12	12	A directory model ACL exists for the parent, but no buffer address was provided for the new directory's directory model ACL in the CredDirModelAcl field
8	12	16	The buffer provided (in the CredDirModelAclLen field) for the new directory's directory model ACL is not large enough. It must be at least as large as the size of the parent's directory model ACL (in the FACL_Len field)
8	12	20	A file model ACL exists for the parent, but no buffer address was provided for the new directory's file model ACL in the CredFileModelAcl field
8	12	24	The buffer provided (in the CredFileModelAclLen field) for the new directory's directory model ACL is not large enough. It must be at least as large as the size of the parent's directory model ACL (in the FACL_Len field)

Usage notes

1. This service is only intended for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.

2. If the CRED user type is system, IRRSMF00 allows the operation, and sets the owning z/OS UNIX user identifier (UID) to zero.
3. IRRSMF00 builds the IFSP in the output_FSP area provided by the caller. The caller must save the IFSP as part of the attributes for the object.
4. IRRSMF00 builds the IFSP with the S_ISUID bit set to zero and the S_ISVTX bit set to the value in the mode byte. If the new object is a directory, and the FILE.GROUPOWNER.SETGID profile exists in the UNIXPRIV class, the S_ISGID bit is inherited from the parent directory. Otherwise, the S_ISGID bit is set to zero.
5. The new object's owning UID is set to the effective UID of the process. By default, the owning GID is set to that of the parent directory. However, if the FILE.GROUPOWNER.SETGID profile exists in the UNIXPRIV class, then the owning GID is determined by the set-gid bit of the parent directory as follows:
   - If the parent's set-gid bit is on, then the owning GID is set to that of the parent directory.
   - If the parent's set-gid bit is off, then the owning GID is set to the effective GID of the process.
6. If the parent directory has a directory model ACL, and the new object is a directory, then the parent's directory model ACL is copied as the new directory's access ACL and directory model ACL. The caller must pass in the address of the parent's directory model ACL in the CredPDirModelAcl field. The caller must pass in the length and address of buffers to contain both the new directory's access ACL and directory model ACL. The buffers must be large enough to contain the copied ACL. The address of the new directory's directory model ACL buffer must be passed in using the CredDirModelAcl field, and its length must be passed in using the CredDirModelAclLen field. The address of the new directory's access ACL buffer must be passed in using the CredAccAcl field, and its length must be passed in using the CredAccAclLen field.
7. If the parent directory has a file model ACL, and the new object is a directory, then the parent's file model ACL is copied as the new directory's file model ACL. The caller must pass in the address of the parent's file model ACL in the CredPFileModelAcl field. The caller must pass in the length and address of a buffer to contain the new directory's file model ACL. The buffer must be large enough to contain the copied ACL. The address of the new directory's file model ACL buffer must be passed in using the CredFileModelAcl field, and its length must be passed in using the CredFileModelAclLen field.
8. If the parent directory has a file model ACL, and the new object is a file, then the parent's file model ACL is copied as the new file's access ACL. The caller must pass in the address of the parent's file model ACL in the CredPFileModelAcl field. The caller must pass in the length and address of a buffer to contain the new file's access ACL. The buffer must be large enough to contain the copied ACL. The address of the new file's access ACL buffer must be passed in using the CredAccAcl field, and its length must be passed in using the CredAccAclLen field.
9. If the SECLABEL class is active, the security label from the owning directory will be propagated to the output FSP unless the security label is SYSMULTI. If the owning directory's security label is SYSMULTI, the security label of the output FSP will be set to that of the requesting address space, unless a system CRED is passed containing a security label. If a system CRED containing a security label is passed when the owning directory's security label is SYSMULTI, the security label from the CRED will be used in the output FSP instead of the address space security label.

Related services

ck_access, R_umask

---

makeISP (IRRSMI00): Make IISP

Function

The **makeISP** service builds an IISP in the area provided by the caller.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
PASN = HASN or PASN not = HASN

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSMI00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Mode_Permissions,
               ALET, Output_ISP,
               ALET, Output_IPCP,
               ALET, CREDIPC
              )
```

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 words 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.

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.

Mode_Permissions
The name of a word containing the mode permission flags to be set for this IPC key. The following is a list of defined permission bits mapped by BPXYMODE:

S_IRUSR
Permits the process that owns the IPC member to read it.

S_IWUSR
Permits the process that owns the IPC member to alter it.

S_IRGRP
Permits the group associated with the IPC member to read it.

S_IWGRP
Permits the group associated with the IPC member to alter it.

S_IROTH
Permits others to read the IPC member.

S_IWOTH
Permits others to alter the IPC member.

Alter and write have the same meaning for access checks. Alter applies to semaphores and write applies to message queueing and shared memory segments.

Output_ISP
The name of a 64-byte area in which the new IISP is built. The name is set by the kernel. See *z/OS Security Server RACF Data Areas*.

Output_IPCP
The name of a 20-byte area in which the new IPCP is built. The name is set by the kernel.

CREDIPC
The name of the CREI structure for the current IPC system callable service. The CREI contains the IPC identifier and IPC key. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSMI00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized.
8	8	12	An internal error occurred during RACF processing.
8	8	32	The CREI user type is not supported.

Usage notes

1. This service is only intended for use by the MVS BCP.
2. The CREI user type must be local (that is, 1).
3. IRRSMI00 builds the IISP in the output_ISP area and the output_IPCP areas provided by the caller. The caller must save the IISP as part of the attributes for the key.
4. The IPCP ALET and address are retrieved from the parameters and set into the output_ISP by RACF.
5. The effective z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) are retrieved from the USP and set into the owner and creator fields of the output_IPCP by RACF.
6. The mode is retrieved from the parameters and set into the output_IPCP by RACF.
7. The IPC Key and IPC ID are retrieved from the CREI and set into the output_ISP by RACF.
8. An audit record is optionally written, depending on the audit options in effect for the system.
9. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.
10. If the SECLABEL class is active, the security label from the creating process will be propagated to the ISP. If the creating process has no security label and MLIPCOBJ is active, the request will fail with an authorization failure.

Related services

ck_IPC_access, R_IPC_ctl

make_root_FSP (IRRSMR00): Make root IFSP

Function

The **make_root_FSP** service initializes an IFSP for the root directory of a new file system being initialized in a shared file system data set.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSMR00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Mode,
               ALET, Output_FSP,
               ALET, File_Identifier,
               ALET, Data_set_name
              )
```

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 words 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.

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.

Mode
The name of a word containing the mode value (the file type, the permission bits, and the S_ISUID, S_ISGID, and S_ISVTX bits) to be set for the file.

See “File type and file mode values” on page 4 for a definition of the security bits in the mode parameter.

Output_FSP
The name of a 64-byte area in which the new IFSP is built.

File_Identifier
The name of a 16-byte area containing a unique identifier of the root directory.

Data_set_name
The name of an area containing the data set name of the shared file system data set being created. This is a 44-byte area padded with blanks.

Return and reason codes

IRRSMR00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
0	0	4	The service was successful. The security label was set from the requesting address space.
4	0	0	RACF is not installed.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

Usage notes

1. This service is only intended for use by DFSMS/MVS, during allocation of an shared file system data set, and by z/OS UNIX System Services servers. The service contains support for z/OS UNIX System Services servers, but cannot be directly invoked by an z/OS UNIX System Services server.
2. IRRSMR00 may be called from a non-z/OS UNIX address space.
3. These are the default attributes set for the root directory:
   - The file's owner z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) are initialized as follows:
     – If the caller is a z/OS UNIX process:
       - owner UID = effective UID of the process
       - owner GID = effective GID of the process

       Note: This differs from **makeFSP** because there is no owning directory to propagate the GID from.
     – If the caller is not a z/OS UNIX process but is defined to RACF as a z/OS UNIX user:
       - owner UID = UID from the user's profile
       - owner GID = GID from the group profile for the user's current group

       If the group has no GID, the owner GID is set to 0.
     – If the caller is not a z/OS UNIX process and is not defined to RACF as a z/OS UNIX user:
       - owning UID = 0
       - owning GID = 0

If the UID or GID is set to 0, a superuser should change the fields to valid values using **chown** after the file system is mounted.

- The permission bits are set from the input mode parameter.
- The S_ISUID and S_ISGID are set to 0 and the S_ISVTX bit is set to the value in the input mode parameter.
- The user audit options are set to audit access failures for all types of access.
- The auditor audit options are set to no auditing.

4. IRRSMR00 builds the IFSP in the output_FSP area provided by the caller. The caller must save the IFSP as part of the attributes for the object.
5. If the SECLABEL class is active, and the containing data set has a security label, the security label will be propagated into the root FSP. If the data set has no security label and MLFSOBJ is active, the security label of the output FSP will be set to that of the requesting address space. Note that only the data set name is available to make_root_FSP, and not the setting of the RACF indicator bit or the volume serial number. Make_root_FSP will look for a discrete profile with the same name as the containing data set. If one is found, the security label from that profile will be used. If no discrete is found, make_root_FSP will look for a covering generic profile. If more than one discrete profile is found with the same name, RACF will return an internal error since it does not know the volume serial number of the data set.

Related services

makeFSP

---

query_file_security_options (IRRSQF00): Query file security options

Function

The **query_file_security_options** service returns the value of the requested file system option.

For a list of supported options, see the Option_code parameter.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
None

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSQF00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Option_code,
               ALET, Output_value
              )
```

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 words 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.

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.

Option_code
The name of a word containing a code identifying the requested option. The code value 1 identifies the _POSIX_CHOWN_RESTRICTED option. Option code value 2 is sent as input to IRRSQF00 to determine if the system supports Access Control Lists (ACLs) or not.

Output_value
The name of a word in which the value of the requested option is returned.

- Option_code value 1

  For option code value 1, the following may be returned:

  0 _POSIX_CHOWN_RESTRICTED is in effect

  -1 _POSIX_CHOWN_RESTRICTED is not in effect

  Note:

If all of the following are true, then _POSIX_CHOWN_RESTRICTED is not in effect:

- The UNIXPRIV class is active
- The UNIXPRIV class has been processed using SETROPTS RACLIST
- The CHOWN.UNRESTRICTED discrete profile exists in the UNIXPRIV class, regardless of its universal access (UACC) value, or the contents of its access list.

- Option_code value 2

  For option code value 2, the following may be returned:

  0 ACLs are supported

  -1 ACLs are not supported

RACF will always return 0, indicating that ACLs are supported.

Return and reason codes

IRRSQF00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The option code is not supported.

Usage note

1. This service is intended only for use by a z/OS UNIX file system.

Related services

None

query_system_security_options (IRRSQS00): Query system security options

Function

The **query_system_security_options** service returns the value of the requested system options. The only supported options are NGROUPS_MAX and _POSIX_SAVED_IDS.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
None

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. RACF returns the following values for the requested system options:
   - NGROUPS_MAX: 300
   - _POSIX_SAVED_IDS: 0

Format

```
CALL IRRSQS00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Option_code,
               ALET, Output_value
              )
```

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 words 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.

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.

Option_code
The name of a word containing a code identifying the requested option. The supported values are:

1 NGROUPS_MAX

2 _POSIX_SAVED_IDS

All other values are reserved.

Output_value
The name of a word in which the value of the requested option is returned.

- The values for _POSIX_SAVED_IDS are:

  0 _POSIX_SAVED_IDS is in effect.

  -1 _POSIX_SAVED_IDS is not in effect.

- The value for NGROUPS_MAX is the maximum number of supplemental groups supported.

Return and reason codes

IRRSQS00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The option code is not supported.

Usage note

- This service is intended only for use by the MVS BCP.

Related services

None

R_admin (IRRSEQ00): RACF administration API

Function

The **R_admin** callable service provides an interface with which to manage and retrieve RACF profile and SETROPTS data. Several function codes are available for use, depending on what profile type you want to manage, and which operation you want to perform. The available functions are:

- Input functions
  - **Run-command**: Accepts a command image and executes this command in the RACF subsystem address space.
  - **Update** functions: Accepts tokenized input from which a RACF command image is constructed, and executed in the RACF subsystem address space. These functions shield the programmer from the details of RACF command syntax. The following RACF information can be managed using the **update** functions:
    - USER profiles
    - GROUP profiles
    - User-to-group connections
    - General resource profiles
    - Data set profiles
    - General resource and data set profile access lists
    - SETROPTS options
- Output functions

- **Profile extract** functions: Return tokenized, formatted data for RACF profiles in all classes except the DATASET class.
- SETROPTS retrieval - returns SETROPTS data in either of two formats:
  - SMF Unload
  - The same tokenized structure used as input to the SETROPTS update function
- Password and password phrase envelope retrieval - Retrieves an encrypted password or password phrase envelope for a specified user.

Most, but not all, of these function codes require the RACF subsystem address space to be up and running. Some function codes require that the caller be in supervisor state, but some are also available for problem state callers. Usually, problem state callers require additional RACF profile authorization and certain options are not available to them (for example, the ability to run the request under a different identity).

The IRRPCOMP mapping macro contains the definitions for the function codes and structure mappings used by **R_admin**. The relevant fields start with prefix ADMN_.

A REXX interface to the profile extract functions is available. This program, named IRRXUTIL, is designed to be invoked by REXX in problem state, and converts the output of an **R_admin** extract request to a set of REXX stem variables. For more information, refer to *z/OS Security Server RACF Macros and Interfaces*.

Requirements

Authorization:
Any PSW key in supervisor or problem state, depending on the function code

Dispatchable unit mode:
Any task

Cross memory mode:
PASN = HASN = SASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary only

Recovery mode:
Recovery must be provided by caller

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space.

RACF authorization

The authorization requirements differ among the **R_admin** functions. For the functions which send requests to the RACF subsystem address space (all of the input functions, and password/password phrase envelope retrieval), a user ID is passed to the RACF address space where an ACEE is created under which to execute the RACF command. For the profile extract functions, which run in the caller's address space, the actual ACEE, if present, will be used for authority checking. If a user ID is provided, an ACEE will be created for this purpose. Only a supervisor state caller can directly specify the user ID or ACEE as a parameter to **R_admin**. The following list shows the possible sources of the user ID or ACEE, in the order in which they are searched:

- The RACF_userID parameter (supervisor state callers only)
- The ACEE_ptr parameter (supervisor state callers only)
- The user ID associated with the current task control block (TCB)
- The user ID associated with the current address space (ASXB)

Where appropriate, see the Authorization Required section for the relevant RACF command, in the *z/OS Security Server RACF Command Language Reference*.

The following table summarizes the authorization required for the function codes:

Function code	**Problem state allowed?**	**Command authority enforced?**	**FACILITY class authorization**
ADMN_RUN_COMD	Yes	Yes	For problem state callers only, READ access to IRR.RADMIN.*command-name*. The resource must be defined using the full command name even if the abbreviated version of the command name is used with **R_Admin**. (For example, 'LU JOEUSER' would require READ authority to IRR.RADMIN.LISTUSER.)
Update function codes (X'01'-X'04' and X'06'- X'15')	No	Yes	N/A
ADMN_XTR_SETR	Yes	Yes (optionally for a supervisor state caller). The authorization rules of the SETROPTS LIST command are applied.	For problem state callers, and for supervisor state callers who request it, READ access to IRR.RADMIN.SETROPTS.LIST
ADMN_UNL_SETR	No	No	N/A
ADMN_XTR_PPENV	No	N/A	READ access to IRR.RADMIN.EXTRACT.PPENV (If this access check is being audited, the logstring will identify the user ID whose password phrase envelope was extracted.)
ADMN_XTR_PWENV	No	N/A	READ access to IRR.RADMIN.EXTRACT.PWENV (If this access check is being audited, the logstring will identify the user ID whose password envelope was extracted.)

Function code	Problem state allowed?	Command authority enforced?	FACILITY class authorization
Profile extract functions (X'19' - X'1D', X'1F', X'20')	Yes	Yes. Each function code maps to a RACF listing command (For example, the authorization rules of the LISTGRP command). This authority can be skipped for a supervisor state caller if requested in the input parameter list.	For problem state callers and for supervisor state callers who request it: • Extract user, extract next user, and extract connect - READ access to IRR.RADMIN.LISTUSER • Extract group and extract next group - READ access to IRR.RADMIN.LISTGRP • Extract resource and extract next resource - READ access to IRR.RADMIN.RLIST

Note:

1. Generic FACILITY class profiles can be used.
2. For the profile extract functions, it is possible that the caller is only authorized to see a subset of the profile information. In this case, only this subset of information is returned. If any information is returned, the caller receives a 0 return code, and no indication that information has been suppressed. Following are some situations in which information is suppressed (for problem state callers, and for supervisor state callers who have not requested that the command authority checks be bypassed). These situations are identical to the corresponding RACF command (e.g. LISTUSER, LISTGRP, and RLIST).
   - Only a user with the AUDITOR attribute (at either the group or system level) can view the UAUDIT setting of a USER profile, or the AUDITOR-related audit settings in a general resource profile.
   - Users can be authorized to view the BASE segment, but no additional segments.
   - With the use of field-level access checking (using the FIELD class), users can be authorized to view certain non-BASE segment information without having authority to view the BASE segment.
   - When certain SECLABEL-related SETROPTS options are in effect, the installation data field of the USER profile is suppressed for all but system SPECIAL users.
   - When a non-privileged user has at least READ, but not ALTER access to a general resource profile, the access list is not returned.
3. For the SETROPTS extract function, it is possible that the caller is authorized to see only a subset of the SETROPTS information. In this case, only this subset of information is returned. If any information is returned, the caller receives a 0 return code, and no indication that information has been suppressed. For any caller who does not have the AUDITOR attribute, or the group-AUDITOR attribute in any of their groups, the auditing related fields are suppressed.

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 words 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 19 on page 78.

RACF_return_code
The name of a fullword in which the service routine stores the return code. These return codes are found in Table 19 on page 78.

RACF_reason_code
The name of a fullword in which the service routine stores the reason code. These reason codes are found in Table 19 on page 78.

Function_code
The name of a 1-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 17 on page 76.

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 17 on page 76.

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 uppercase. If specified, the RACF command executed through **R_admin** will run under the authority of this userID. Ignored for problem state callers.

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, except for the extract user/group/connect function codes. 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. Ignored for problem state callers.

Out_message_subpool
The name of a 1-byte field that specifies the subpool used to obtain storage for output that is 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. See the appropriate mappings of the output data for each function code in "Reference documentation by function code" on page 77.

Function code values

Table 17 shows the function code values in the mapping macro IRRPCOMP.

Table 17. 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
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
ADMN_XTR_USER	X'19'	Extract a user profile
ADMN_XTR_NEXT_USER	X'1A'	Extract the next user profile
ADMN_XTR_GROUP	X'1B'	Extract a group profile
ADMN_XTR_NEXT_GROUP	X'1C'	Extract the next group profile

Table 17. Function code values in mapping macro IRRPCOMP (continued)

Function code	Value	Description
ADMN_XTR_CONNECT	X'1D'	Extract connection information for a specified user and group
ADMN_XTR_PPENV	X'1E'	Extract PKCS #7 encrypted password phrase envelope
ADMN_XTR_RESOURCE	X'1F'	Extracts a general resource profile
ADMN_XTR_NEXT_RESOURCE	X'20'	Extracts the next general resource profile

Reference documentation by function code

Table 18 lists reference documentation for each of the function codes. The documentation provides a brief overview of the function, and detailed information about the input parameters and the output format. Note that for the update functions (ADMN_ADD_xxx, ADMN_DEL_xxx, ADMN_ALT_xxx, ADMN_LST_xxx, ADMN_CONNECT, ADMN_PERMIT , and ADMN_REMOVE) there is a common overview section at "R_admin update functions" on page 83.

Table 18. Reference documentation for Function_code values

Function_code(s)	Reference documentation
ADMN_ADD_USER, ADMN_DEL_USER, ADMN_ALT_USER, ADMN_LST_USER	"User administration" on page 84
ADMN_XTR_PWENV, ADMN_XTR_PPENV	"Password and password phrase envelope retrieval" on page 110
ADMN_XTR_USER, ADMN_XTR_NEXT_USER, ADMN_XTR_GROUP, ADMN_XTR_NEXT_GROUP, ADMN_XTR_CONNECT, ADMN_XTR_RESOURCE, ADMN_XTR_NEXT_RESOURCE	"Profile extract functions" on page 99
ADMN_RUN_COMD	"Running RACF commands" on page 82
ADMN_ADD_GROUP, ADMN_DEL_GROUP, ADMN_ALT_GROUP, ADMN_LST_GROUP	"Group administration" on page 87
ADMN_CONNECT, ADMN_REMOVE	"Group connection administration" on page 89
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	"General resource profile administration" on page 90
ADMN_ALT_SETR	"SETROPTS administration" on page 97
ADMN_XTR_SETR, ADMN_UNL_SETR	"SETROPTS reporting functions" on page 108

Return and reason codes

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

Table 19. 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. For ADMIN_XTR_PPENV, the user does not have a password phrase envelope.
4	4	4	The target profile does not exist. For ADMN_XTR_PWENV and ADMN_XTR_PPENV, the target user does not exist. For extract functions, the requested profile does not exist. For extract-next requests, there are no more profiles that the caller is authorized to extract.
4	4	8	For ADMN_XTR_PWENV, a problem was encountered while retrieving the password envelope. For ADMN_XTR_PPENV, a problem was encountered while retrieving the password phrase envelope.
4	4	12	The specified class could not be found in the class descriptor table. Note that the class cannot be USER, GROUP, CONNECT, or DATASET.
4	4	16	A discrete profile was not found, and a matching generic profile could not be located because the class is inactive or SETROPTS NOGENERIC is in effect for the class. A matching generic profile may or may not exist for the resource name specified.
4	4	20	On an extract-next request, the next profile encountered is a discrete profile which contains at least one generic character. See usage note 8 on page 81 for more information.
8	8	0	Incorrect function code
8	8	4	Input parameter list error. See usage note 7 on page 81 for possible causes.

Table 19. Return and reason codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	8	Invalid data in the input parameter list may cause the program to abend. Some fields in the parameter list must be coded with the defined length and data format. The profile segment name, for example must not be longer than 8 bytes. If more than 8 bytes are passed, unpredictable results may occur and the user gets a return code of 8 8 8 with a possible program abend. If the program ABENDs before GTF trace records are created, then you do not have GTF trace records to use for debugging. Check the input parameter list for errors.
8	8	12	Recovery environment could not be established.
8	8	16	Invalid request specified. For ADMN_RUN_COMD, the input command image represented an incorrect or unsupported command. For all other functions, the input parameter list contained an incorrect segment name, field name, or flag byte, or incorrectly specified field data. The function-specific parameter list header contains an offset to the segment or field entry in error, relative to the start of the parameter list.
8	8	20	Function not supported for problem state caller.
8	8	24	Caller not authorized
8	12	*IEFSSREQ return code*	Unable to invoke the RACF subsystem. Reason code field contains a return code from the IEFSSREQ macro invocation. See *z/OS MVS Using the Subsystem Interface* for information about possible return codes from IEFSSREQ. **Note:** A LOGREC record is cut by RACF with the function_code, cmd, and cmd_length.
8	16	*RACF command return code*	RACF request failed. Reason code field contains the return code from the RACF request. See *z/OS Security Server RACF Command Language Reference* for information about possible return codes from the RACF commands. In addition, diagnostic output resulting from the RACF command may be present. The Out_message_strings parameter should be interrogated by the caller.

Table 19. Return and reason codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	20	ICHEINTY return/reason code	An unexpected ICHEINTY macro error was encountered during an extract or extract-next request. The high order halfword of the reason code contains the ICHEINTY return code and the low order halfword contains the ICHEINTY reason code. See *z/OS Security Server RACF Macros and Interfaces* for ICHEINTY return and reason code documentation.
8	24	*xx*	Unexpected internal error occurred. The value of xx is an internal code to be used by the IBM support center.

Note: Return and reason codes are shown in decimal.

Usage notes

1. You must link edit the IRRSEQ00 callable service stub into your application code to resolve the entry point address at run time.
2. For the Out_message_subpool parameter, select a subpool carefully. z/OS makes certain assumptions about subpool usage and characteristics. Using subpool 0 or 250 or any subpool documented in *z/OS MVS Programming: Authorized Assembler Services Guide* as having a storage key of USER (for example, 227-231 and 241) may give unpredictable results.
3. All requests are processed synchronously. Control is not returned to the caller until RACF has processed the administration request and output, if any has been returned to the caller.
4. For the ADMN_RUN_COMD function code, the following RACF commands are not supported through this interface:
   - BLKUPD
   - RACLINK
   - RVARY
   - RACF operator commands (DISPLAY, RESTART, SET, SIGNOFF, STOP, and TARGET)

   RACF TSO administrative commands may not be directed to other RACF remote sharing facility (RRSF) nodes. The command image passed by the caller cannot contain the keywords AT or ONLYAT. These keywords cause the command to fail with SAF return code 8, RACF return code 16, RACF reason code 8.

   These messages are returned as command output:

   IRRV013I *subsystem-name* SUBSYSTEM *racf-command* COMMAND FROM THE IRRSEQ00 CALLABLE SERVICE WAS NOT PROCESSED.

   IRRV014I subsystem-name SUBSYSTEM AT() OR ONLYAT() KEYWORDS MAY NOT BE SPECIFIED WITH COMMANDS FROM THE IRRSEQ00 CALLABLE SERVICE.

   Any update to the RACF database caused by this service is subject to automatic direction and password synchronization as implemented by the installation.

5. The parameter list passed to this service is a variable-length (VL) parameter list. The high-order bit of the last field (address of Out_message_strings) must be set to mark the end of the parameter list.
6. All field data must be supplied in character format. For information about the contents of the field data, refer to *z/OS Security Server RACF Command Language Reference* for the appropriate command keyword as indicated in the following tables. For example, looking at Table 148 on page 367 to find details on the content of the HLDCLASS field, see the ADDUSER/ALTUSER documentation for the HOLDCLASS keyword of the TSO segment.

   Additionally, RACF has a restriction of no more than 255 operands affecting a single nonbase segment (such as the TSO segment in a user profile, or the TME segment in a general resource profile) on a single command. Since the **R_admin** callable service generates a RACF command, this restriction applies to the number of field operands affecting nonbase segments. For the CSDATA segment in a user or group profile, this RACF restriction is further limited to no more than 85 operands on a single command. See the "RACF command restriction for nonbase segments in RACF profiles" section in *z/OS Security Server RACF Command Language Reference* for specifics on this restriction.
7. The following errors result in an "input parameter list error" being returned to the caller:
   - VL bit not set
   - An incorrectly specified ADMN_USRADM_USER_LEN, ADMN_GRPADM_LEN, or ADMN_RESADM_CLAS_LEN (must be from 1-8, inclusively)
   - An incorrectly specified length for the RACF user ID parameter (must be from 0 to 8, inclusively)
   - Invalid profile name length specified in input parameter list for profile extract functions (must be greater than 0 and less than or equal to 8 for USER or GROUP, less than or equal to 17 for CONNECT, and less than or equal to 246 for a general resource class). Note that the actual maximum profile length is determined by the Class Descriptor Table entry for a given resource class, and the length can be less than 246 characters. If a request specifies a profile name which is longer than the allowed maximum for the specified class, a "profile not found" (4/4/4) return code combination will result.
   - Setting ADMN_USRADM_SEG_NUM=0 on any of the list functions
   - Omitting the PROFILE field on any of the general resource, data set (except list), permit, or profile extract function codes
   - Specifying a subpool outside the range of 1 to 127 when the caller is in problem state
   - For ADMN_XTR_NEXT_RESOURCE, specifying a resource name containing generic characters without turning on the ADMN_PROF_GENERIC flag.
8. When a return code combination of 4/4/20 is returned on an extract-next request, this means that RACF has encountered a profile name which appears to be generic, but is in fact discrete. This is almost always an error condition in the RACF database, and continuing to extract profiles will have unexpected results. The offending profile name is returned in the output buffer, but no profile data is returned.

   The normal cause of this problem is that the profile was defined prior to activating generics in the class. To fix the problem, delete the profile and define the profile again. For example:

   ```
   RDELETE class profile-name
   RDEFINE class profile-name operands ...
   ```

Issue PERMIT commands as appropriate to recreate the access list.

Some profiles are not used to protect resources, but instead contain data (such as profiles in the DIGTCERT class, which contain digital certificates). It may be valid for such profiles to contain asterisks or other generic characters, without generics being active for the class. These profiles will cause problems when R_admin is used to extract all profiles from the class. In the case of digital certifcates, **R_admin** does not return any actual information related to the certificate. To obtain certificate information, **R_datalib** (IRRSDL00) or Database Unload are the appropriate services.

Note that the IRRICE member of SYS1.SAMPLIB contains a sample query (named BGGR) which searches Database Unload output for discrete profiles containing generic characters.

9. Inconsistencies can occur when extracting profiles in RACLISTed classes if the in-storage profiles are different from those in the RACF database. In other words, when you are extracting a profile which has been changed since the last time the class was refreshed, the authorization information could be inconsistent between the in-storage copy, which was used to determine your authorization, and the database copy which is actually returned. This could result in certain profiles being unexpectedly returned or not.

Related services

None

Reference documentation

The following information describes how to use the various functions of **R_admin**:

Running RACF commands

The **R_admin run-command** function accepts a command image that your application has constructed, and executes this command in the RACF subsystem address space. It can be invoked by problem state callers.

Output, if any, which resulted from RACF's processing of the command is returned to the caller in virtual storage. There is a maximum of 4096 lines (not bytes) of output which will be returned by **R_admin**.

Run-command does not support all RACF commands. For a list of the commands that are not allowed by this service, see “Usage notes” on page 80.

The exact format (spacing and order) of the data in the command output or messages does not constitute a programming interface. No programs should depend on the exact format of this data. The **R_admin** extract functions, described below, provide a programming interface with which to retrieve RACF profile data for some profile types. The extract functions are not subject to the 4096 limit on output lines.

For the ADMN_RUN_COMD (**run-command**) function code, the structure associated with the function-specific parameter list is mapped as follows:

Table 20. 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.

Table 20. Parameter list format for running a command (continued)

Offset	Length	Description
2	*	Syntactically correct RACF TSO administration command string.

The command string must be left-justified within the input buffer.

The output from **run-command** is either command output or error messages. The output format is shared with the **R_admin update** functions and is documented in "Command output message block mapping" on page 98.

R_admin update functions

This set of **R_admin** function codes allow the caller to add, alter, delete, and list RACF profile and SETROPTS information without requiring knowledge of RACF command syntax. These functions accept an input parameter list containing the names of the segments and fields you wish to manipulate, along with the data for the fields. These functions will construct a RACF command from the input parameter list, send the command to the RACF address space, and return the command output to the caller. These functions are most useful for adding, altering, and deleting RACF profiles.

Note: When listing profiles, you are getting RACF command output, which is not a programming interface. The extract functions are the preferred method for retrieving RACF profile information for those profile types which are supported. For more information about extract functions, see "Profile extract functions" on page 99.

These functions can only be invoked by supervisor state callers. Tables referred to below describe the supported segments and fields for the various profile types.

The following is a schematic representation of the input parameter list:

Table 21. Input parameter list for update functions

Request header
Segment entry 1
Field Entry 1
... ...
Field Entry *n*
Segment entry 2
Field Entries ...
Segment entry *y*
Field Entries ...

The header identifies the name of the profile being managed (for user and group operations), and the number of segment entries contained in the parameter list. Each segment entry contains the segment name, and the number of fields provided for that segment. Each field entry identifies a field in a RACF profile, and provides the new value to be set for the field. A field entry can also be used to delete the field. For general resource, data set, and group connection operations, a field entry in the BASE segment is used to identify the profile being manipulated.

Parameter list construction is simpler for the list and delete functions because the listing functions only accept optional segment entries, and the delete functions do not require any segment entries at all. For listing and deleting, it may be simpler to construct your own command image and use the **run-command** function. See “Segment and field entry mappings” on page 357 for detailed information about segment and field entries.

The request header mapping varies slightly depending on what profile type you are managing. Individual sections for each profile type will document the request header mapping for that function, along with a list of the supported fields, and their characteristics.

The concepts of profiles, segments and fields will be familiar to anyone with some knowledge of the RACF commands, classes and templates. In order to provide a common view of the data, **R_admin** stretches this concept slightly in a few cases.

- SETROPTS information can be manipulated using **R_admin**. By convention, all SETROPTS fields reside within a BASE segment entry, even though SETROPTS information does not actually reside within a RACF profile.
- Group connections are managed not by updating fields for a user or group profile, but rather by managing fields within a BASE segment entry for a logical "CONNECT" profile. Only one connection can be managed per request.
- Profile access lists are managed by updating fields within a logical BASE segment entry for a general resource or data set profile. Only one access list entry can be managed per request, and this is a separate request type than the standard add and alter functions for general resources and data sets.

User administration: For the ADMN_ADD_USER, ADMN_DEL_USER, ADMN_ALT_USER, ADMN_LST_USER, ADMN_XTR_PPENV, and ADMN_XTR_PWENV function codes, the mapping associated with the function-specific parameter list is mapped as in Table 22.

Table 22. 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

For the ADMN_DEL_USER, ADM_XTR_PPENV, and ADMN_XTR_PWENV function codes, no segment data is expected. The number of segments should be zero. If non-zero, any segment data present is ignored.

See “Segment and field entry mappings” on page 357 for segment and field entry information.

The output from these functions, except for password and password phrase envelope retrieval, is either command output or error messages. See “Command output message block mapping” on page 98 for the output format.

For tables defining the field names and their usage, see “User administration” on page 359.

Examples: The following examples are not coding samples, rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Add user BRUCE with some BASE segment fields and some OMVS segment fields.

Function code = ADMN_ADD_USER

Note: For fields which take quoted data (for example, user name), the quotes must be included as part of the data.

```
* First, define the request header
HEADER  DS 0H
        DC AL1(5)         Length of user
        DC CL8'BRUCE'     User name
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(2)         Number of segments (BASE+OMVS)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(3)         Field count - 3
* First BASE segment field entry
BFLD1   DC CL8'NAME'      Name field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(13)        Length of field data
        DC CL13'''BRUCE WELLS'''  Field data
* Second BASE segment field entry
BFLD2   DC CL8'OWNER'     Owner field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(7)         Length of field data
        DC CL7'RACFDEV'   Field data
* Third BASE segment field entry
BFLD3   DC CL8'SPECIAL'   Special field
        DC CL1'Y'         Flag byte - Y - boolean value
        DC AL2(0)         Length 0 - no data for booleans
* Second segment entry - OMVS
OSEG    DC CL8'OMVS'      OMVS segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(3)         Field count - 3
* First OMVS segment field entry
OFLD1   DC CL8'UID'       UID field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'3500'      Field data
* Second OMVS segment field entry
OFLD2   DC CL8'HOME'      Home field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(10)        Length of field data
        DC CL10'/u/brwells' Field data
* Third OMVS segment field entry
OFLD3   DC CL8'PROGRAM'   Program field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(7)         Length of field data
        DC CL7'/bin/sh'   Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_ADD_USER

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(3)
BFLD1   DC CL8'NAME',CL1'Y',AL2(13),CL13'''BRUCE WELLS'''
BFLD2   DC CL8'OWNER',CL1'Y',AL2(7),CL7'RACFDEV'
BFLD3   DC CL8'SPECIAL',CL1'Y',AL2(0)
OSEG    DC CL8'OMVS',CL1'Y',AL2(3)
OFLD1   DC CL8'UID',CL1'Y',AL2(4),CL4'3500'
OFLD2   DC CL8'HOME',CL1'Y',AL2(10),CL10'/u/brwells'
OFLD3   DC CL8'PROGRAM',CL1'Y',AL2(7),CL7'/bin/sh'
```

Example 3: Alter user BRUCE to add some categories and delete a couple of OMVS segment fields.

Function code = ADMN_ALT_USER

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(1)
BFLD1   DC CL8'CATEGORY',CL1'A',AL2(14),CL14'CAT1 CAT2 CAT3'
OSEG    DC CL8'OMVS',CL1'Y',AL2(2)
OFLD1   DC CL8'UID',CL1'N',AL2(0)
OFLD2   DC CL8'HOME',CL1'N',AL2(0)
```

Example 4: Alter user BRUCE to delete some categories, delete the entire OMVS segment, and add a TSO segment.

Function code = ADMN_ALT_USER

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(3)
BSEG    DC CL8'BASE',CL1'Y',AL2(1)
BFLD1   DC CL8'CATEGORY',CL1'D',AL2(9),CL9'CAT2 CAT3'
OSEG    DC CL8'OMVS',CL1'N',AL2(0)
TSEG    DC CL8'TSO',CL1'Y',AL2(2)
TFLD1   DC CL8'ACCTNUM',CL1'Y',AL2(7),CL7'ACCT123'
TFLD2   DC CL8'JOBCLASS',CL1'Y',AL2(1),CL1'J'
```

Example 5: List user BRUCE, displaying the OMVS and TSO segments (and not the BASE segment).

Function code = ADMN_LST_USER

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(2)
OSEG    DC CL8'OMVS',CL1'Y',AL2(0)
TSEG    DC CL8'TSO',CL1'Y',AL2(0)
```

Example 6: Delete user BRUCE.

Function code = ADMN_DEL_USER

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(0)
```

Example 7: Retrieve the password envelope for user BRUCE.

Function code = ADMN_XTR_PWENV

```
HEADER  DC AL1(5),CL8'BRUCE',AL1(0),AL2(0),AL2(0)
```

Note: This example also applies to the extraction of a password phrase envelope using the function code ADMN_XTR_PPENV.

Group administration: For the ADMN_ADD_GROUP, ADMN_DEL_GROUP, ADMN_ALT_GROUP, and ADMN_LST_GROUP function codes, the mapping associated with the function-specific parameter list is mapped as follows:

Offset	Length	Description
0	1	Length of the Group Name
1	8	Uppercase RACF Group Name
9	1	Reserved
10	2	Output offset to the segment or field entry in error in relation to the start of the Parm_list. Only applied to ADMN_ADD_GROUP and ADMN_ALT_GROUP requests when an "invalid 'request" error is returned to the caller.
12	2	Number of RACF profile segments
14	*	Start of first segment entry

For ADMN_DEL_GROUP, no segment data is expected. The number of segments should be zero. If non-zero, any segment data present is ignored.

See “Segment and field entry mappings” on page 357 for segment and field entry information.

The output from these functions is either command output or error messages. See “Command output message block mapping” on page 98 for the output format.

For tables defining the field names and their usage, see “Group administration” on page 368.

Examples: The following examples are not coding samples, rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Add group FINANCE with some BASE segment fields and an OMVS segment.

Function code = ADMN_ADD_GROUP

Note: For fields which take quoted data (for example, installation data), the quotes must be included as part of the data.

```
* First, define the request header
HEADER  DS 0H
        DC AL1(7)         Length of group
        DC CL8'FINANCE'   Group name
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(2)         Number of segments (BASE+OMVS)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(4)         Field count - 4
* First BASE segment field entry
```

```
BFLD1   DC CL8'DATA'      Data field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(20)        Length of field data
        DC CL20'''FINANCE DEPARTMENT'''  Field data
* Second BASE segment field entry
BFLD2   DC CL8'OWNER'     Owner field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'CORP'      Field data
* Third BASE segment field entry
BFLD3   DC CL8'SUPGROUP'  Superior group field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'CORP'      Field data
* Fourth BASE segment field entry
BFLD4   DC CL8'UNIVERSL'  Universal field
        DC CL1'Y'         Flag byte - Y - boolean value
        DC AL2(0)         Length 0 - no data for booleans
* Second segment entry - OMVS
OSEG    DC CL8'OMVS'      OMVS segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(1)         Field count - 1
* First OMVS segment field entry
OFLD1   DC CL8'GID'       GID field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(2)         Length of field data
        DC CL2'46'        Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_ADD_GROUP

```
HEADER  DC AL1(7),CL8'FINANCE',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'DATA',CL1'Y',AL2(20),CL20'''FINANCE DEPARTMENT'''
BFLD2   DC CL8'OWNER',CL1'Y',AL2(4),CL4'CORP'
BFLD3   DC CL8'SUPGROUP',CL1'Y',AL2(4),CL4'CORP'
BFLD4   DC CL8'UNIVERSL',CL1'Y',AL2(0)
OSEG    DC CL8'OMVS',CL1'Y',AL2(1)
OFLD1   DC CL8'GID',CL1'Y',AL2(2),CL2'46'
```

Example 3: Alter group FINANCE to remove the OMVS segment and add a DFP segment.

Function code = ADMN_ALT_GROUP

```
HEADER  DC AL1(7),CL8'FINANCE',AL1(0),AL2(0),AL2(2)
OSEG    DC CL8'OMVS',CL1'N',AL2(0)
DSEG    DC CL8'DFP',CL1'Y',AL2(2)
DFLD1   DC CL8'DATAAPPL',CL1'Y',AL2(5),CL5'APPL1'
DFLD2   DC CL8'DATACLAS',CL1'Y',AL2(6),CL6'CLASS1'
```

Example 4: List group FINANCE, showing the BASE, OMVS, and DFP segments.

Function code = ADMN_LST_GROUP

```
HEADER  DC AL1(7),CL8'FINANCE',AL1(0),AL2(0),AL2(3)
BSEG    DC CL8'BASE',CL1'Y',AL2(0)
OSEG    DC CL8'OMVS',CL1'Y',AL2(0)
DSEG    DC CL8'DFP',CL1'Y',AL2(0)
```

Example 5: Delete the FINANCE group.

Function code = ADMN_DEL_GROUP

```
HEADER  DC AL1(7),CL8'FINANCE',AL1(0),AL2(0),AL2(0)
```

Group connection administration: For the ADMN_CONNECT and ADMN_REMOVE function codes the mapping associated with the function-specific parameter list is mapped as follows:

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 field entry in error in relationship to the start of the Parm_list. Only applies to ADMN_CONNECT and ADMN_REMOVE requests when an "Invalid Request" error is returned to the caller.
12	2	Number of RACF segments. The value is 1, for base segment only.
14	*	Start of first segment entry

By convention, use the BASE segment to specify field information for ADMN_CONNECT and ADMN_REMOVE. The flag byte in the segment entry is ignored.

Note: The request header identifies the user, but not the group, in which the connection applies. Set up a field entry for the GROUP field in the BASE segment in order to specify the group name to **R_admin**.

See “Segment and field entry mappings” on page 357 for segment and field entry information.

See “Group connection administration” on page 370 for tables defining the field names and their usage.

The output from these functions is either command output or error messages. See “Command output message block mapping” on page 98 for the output format.

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Connect user JOSEPH to the FINANCE group with various authorities.

Function code = ADMN_CONNECT

```
* First, define the request header
HEADER  DS 0H
        DC AL1(6)         Length of user
        DC CL8'JOSEPH'    User name
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(1)         Number of segments (BASE only)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(4)         Field count - 4
* First BASE segment field entry
BFLD1   DC CL8'GROUP'     Group field
```

```
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(7)         Length of field data
        DC CL7'FINANCE'   Field data
* Second BASE segment field entry
BFLD2   DC CL8'AUTH'      Authority field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(7)         Length of field data
        DC CL7'CONNECT'   Field data
* Third BASE segment field entry
BFLD3   DC CL8'AUDITOR'   Auditor field
        DC CL1'Y'         Flag byte - Y - boolean value
        DC AL2(0)         No field data for booleans
* Fourth BASE segment field entry
BFLD4   DC CL8'UACC'      Universal access field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(6)         Length of field data
        DC CL6'UPDATE'    Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_CONNECT

```
HEADER  DC AL1(6),CL8'JOSEPH',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'GROUP',CL1'Y',AL2(7),CL7'FINANCE'
BFLD2   DC CL8'AUTH',CL1'Y',AL2(7),CL7'CONNECT'
BFLD3   DC CL8'AUDITOR',CL1'Y',AL2(0)
BFLD4   DC CL8'UACC',CL1'Y',AL2(6),CL6'UPDATE'
```

Example 3: Alter JOSEPH's connection to FINANCE to remove the AUDITOR attribute and define a revoke and resume date.

Function code = ADMN_CONNECT

```
HEADER  DC AL1(6),CL8'JOSEPH',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'GROUP',CL1'Y',AL2(7),CL7'FINANCE'
BFLD2   DC CL8'AUDITOR',CL1'N',AL2(0)
BFLD3   DC CL8'REVOKE',CL1'Y',AL2(7),CL7'8/29/05'
BFLD4   DC CL8'RESUME',CL1'Y',AL2(6),CL6'9/8/05'
```

Example 4: Remove JOSEPH's connection to FINANCE.

Function code = ADMN_REMOVE

```
HEADER  DC AL1(6),CL8'JOSEPH',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(1)
BFLD1   DC CL8'GROUP',CL1'Y',AL2(7),CL7'FINANCE'
```

General resource profile administration: For all of the resource-related function codes (ADMN_ADD_GENRES, ADMN_ALT_GENRES, ADMN_DEL_GENRES, ADMN_LST_GENRES, ADMN_ADD_DS, ADMN_ALT_DS, ADMN_DEL_DS, ADMN_LST_DS, and ADMN_PERMIT), the mapping associated with the function-specific parameter list is mapped as follows:

Offset	Length	Description
0	1	Length of the class name
1	8	Uppercase RACF class name
9	1	Reserved

Offset	Length	Description
10	2	Output offset to the segment or field entry in error in relation to the start of the Parm_list. Only applies to add, alter, and list requests when an "Invalid Request" error is returned to the caller.
12	2	Number of RACF profile segments
14	*	Start of the first segment entry

Note:

1. The class name is not required for the data set functions.
2. The request header identifies the class name, but not the resource name itself. You should set up a field entry for the PROFILE field in the BASE segment in order to specify the resource name to **R_admin**.

The BASE segment is used to specify field information for ADMN_PERMIT. You should also use the BASE segment to specify certain keywords for the delete and list functions. For example, you may need to specify the GENERIC keyword when deleting a data set. Also, for example, when using a list function, you can specify the NORACF field if you do not want the BASE segment listed.

See “Segment and field entry mappings” on page 357 for segment and field entry information.

The output from these functions is either command output or error messages. See “Command output message block mapping” on page 98 for the output format.

The general resource, data set, and permit functions each utilize a separate set of field definitions. Table 23 shows where to find the field definitions for the resource related function codes.

Table 23. Resource related field definitions

For Field Definitions	See
ADMN_ADD_GESRES ADMN_ALT_GENRES ADMN_LST_GENRES	“General resource administration” on page 372
ADMN_ADD_DS, ADMN_ALT_DS, ADMN_LST_DS, ADMN_DEL_DS	“Data set administration” on page 380
ADMN_PERMIT	“Access list administration” on page 382

Note: For ADMN_DEL_GENRES, specify only the PROFILE field in the BASE segment.

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Define the IRR.RADMIN.EXTRACT.PWENV profile in the FACILITY class with universal access of NONE, notify user who is also the owner, and some installation data.

Note:

- The profile name is identified as a (required) field in the BASE segment.

- For fields which take quoted data (for example, installation data), the quotes must be included as part of the data.
- When altering fields in RACLISTed classes, the class must be refreshed after making the update. See "SETROPTS administration" on page 97 for examples.

Function code = ADMN_ADD_GENRES

```
* First, define the request header
HEADER  DS 0H
        DC AL1(8)         Length of class
        DC CL8'FACILITY'  Class name
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(1)         Number of segments (BASE only)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(5)         Field count - 5
* First BASE segment field entry
BFLD1   DC CL8'PROFILE'   Profile name - required!
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(24)        Length of field data
        DC CL24'IRR.RADMIN.EXTRACT.PWENV' Field data
* Second BASE segment field entry
BFLD2   DC CL8'OWNER'     Owner field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(6)         Length of field data
        DC CL6'SHERID'    Field data
* Third BASE segment field entry
BFLD3   DC CL8'NOTIFY'    Notify field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(6)         Length of field data
        DC CL6'SHERID'    Field data
* Fourth BASE segment field entry
BFLD4   DC CL8'UACC'      Universal access field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'NONE'      Field data
* Fifth BASE segment field entry
BFLD5   DC CL8'DATA'      Installation data field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(43)         Length of field data
        DC CL43'''Protects extraction of password envelopes'''
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_ADD_GENRES

```
HEADER  DC AL1(8),CL8'FACILITY',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(5)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(24),CL24'IRR.RADMIN.EXTRACT.PWENV'
BFLD2   DC CL8'OWNER',CL1'Y',AL2(6),CL6'SHERID'
BFLD3   DC CL8'NOTIFY',CL1'Y',AL2(6),CL6'SHERID'
BFLD4   DC CL8'UACC',CL1'Y',AL2(4),CL4'NONE'
BFLD5   DC CL8'DATA',CL1'Y',AL2(43),CL43'''Protects extraction of passw+
              ord envelopes'''
```

Example 3: For the profile in the previous example, specify that all access attempts are to be logged, using the auditing options for the AUDITOR. In addition, assuming the profile already contains three categories, remove two of them.

Function code = ADMN_ALT_GENRES

```
HEADER  DC AL1(8),CL8'FACILITY',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(3)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(24),CL24'IRR.RADMIN.EXTRACT.PWENV'
BFLD2   DC CL8'GAUDREAD',CL1'Y',AL2(3),CL3'ALL'
BFLD3   DC CL8'CATEGORY',CL1'D',AL2(9),CL9'CAT2 CAT3'
```

Example 4: Alter a TERMINAL class profile to specify the days and times it can be used to log on to the system. Remove any installation data which may be present.

Function code = ADMN_ALT_GENRES

```
HEADER  DC AL1(8),CL8'TERMINAL',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(8),CL8'TERMID01'
BFLD2   DC CL8'DATA',CL1'N',AL2(0)
BFLD3   DC CL8'WHENDAYS',CL1'Y',AL2(8),CL8'WEEKDAYS'
BFLD4   DC CL8'WHENTIME',CL1'Y',AL2(9),CL9'0800:1800'
```

Example 5: Add a CDTINFO segment to an existing profile in the CDT class. Delete the KERB segment.

Function code = ADMN_ALT_GENRES

```
HEADER  DC AL1(3),CL8'CDT',AL1(0),AL2(0),AL2(3)
BSEG    DC CL8'BASE',CL1'Y',AL2(1)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(7),CL7'MYCLASS'
KSEG    DC CL8'KERB',CL1'N',AL2(0)
CSEG    DC CL8'CDTINFO',CL1'Y',AL2(7)
CFLD1   DC CL8'CDTUACC',CL1'Y',AL2(4),CL4'NONE'
CFLD2   DC CL8'CDTFIRST',CL1'Y',AL2(5),CL5'ALPHA'
CFLD3   DC CL8'CDTOTHER',CL1'Y',AL2(21),CL21'ALPHA NUMERIC SPECIAL'
CFLD4   DC CL8'CDTMAXLN',CL1'Y',AL2(2),CL2'42'
CFLD5   DC CL8'CDTPOSIT',CL1'Y',AL2(3),CL3'303'
CFLD6   DC CL8'CDTRACL',CL1'Y',AL2(8),CL8'REQUIRED'
CFLD7   DC CL8'CDTSLREQ',CL1'Y',AL2(3),CL3'YES'
```

Example 6: List the profile in the previous example, showing all of the BASE segment information, and the information in the CDTINFO segment.

Function code = ADMN_LST_GENRES

```
HEADER  DC AL1(3),CL8'CDT',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(2)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(7),CL7'MYCLASS'
BFLD2   DC CL8'ALL',CL1'Y',AL2(0)
CSEG    DC CL8'CDTINFO',CL1'Y',AL2(0)
```

Example 7: List the profile in the previous example, but suppress the BASE segment information.

Function code = ADMN_LST_GENRES

```
HEADER  DC AL1(3),CL8'CDT',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(2)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(7),CL7'MYCLASS'
BFLD2   DC CL8'NORACF',CL1'Y',AL2(0)
CSEG    DC CL8'CDTINFO',CL1'Y',AL2(0)
```

Example 8: Delete the profile in the previous example.

Function code = ADMN_DEL_GENRES

```
HEADER  DC AL1(3),CL8'CDT',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(1)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(7),CL7'MYCLASS'
```

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Define a discrete data set profile with a security label of SECRET and a DFP segment.

Note:

- The profile name is identified as a (required) field in the BASE segment.
- For fields which take quoted data (for example, installation data), the quotes must be included as part of the data.
- Since the data set name is not quoted, it will be prefixed with the creator's user ID.

Function code = ADMN_ADD_DS

```
* First, define the request header
HEADER  DS 0H
        DC AL1(7)         Length of class
        DC CL8'DATASET'   Class name - optional for DATASET
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(2)         Number of segments (BASE+DFP)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(4)         Field count - 4
* First BASE segment field entry
BFLD1   DC CL8'PROFILE'   Profile name - required!
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(9)         Length of field data
        DC CL9'TEST.DATA' Field data
* Second BASE segment field entry
BFLD2   DC CL8'VOLUME'    Volume field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(6)         Length of field data
        DC CL6'D79PK5'    Field data
* Third BASE segment field entry
BFLD3   DC CL8'UNIT'      Unit field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'3390'      Field data
* Fourth BASE segment field entry
BFLD4   DC CL8'SECLABEL'  Security label field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(6)         Length of field data
        DC CL6'SECRET'    Field data
* Second segment entry - DFP
DSEG    DC CL8'DFP'       DFP segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(1)         Field count - 1
* First DFP segment field entry
DFLD1   DC CL8'RESOWNER'  Resource owner field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(5)         Length of field data
        DC CL5'MAURA'     Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_ADD_DS

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(9),CL9'TEST.DATA'
BFLD2   DC CL8'VOLUME',CL1'Y',AL2(6),CL6'D79PK5'
BFLD3   DC CL8'UNIT',CL1'Y',AL2(4),CL4'3390'
BFLD4   DC CL8'SECLABEL',CL1'Y',AL2(6),CL6'SECRET'
DSEG    DC CL8'DFP',CL1'Y',AL2(1)
DFLD1   DC CL8'RESOWNER',CL1'Y',AL2(5),CL5'MAURA'
```

Example 3: Define a fully qualified generic data set profile, explicitly specifying a high level qualifier by enclosing the profile name in quotes. Model the profile based on the profile defined in the previous example.

Function code = ADMN_ADD_DS

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(18),CL18'''DEPT06.TEST.DATA'''
BFLD2   DC CL8'GENERIC',CL1'Y',AL2(0)
BFLD3   DC CL8'FROM',CL1'Y',AL2(9),CL9'TEST.DATA'
BFLD4   DC CL8'FVOLUME',CL1'Y',AL2(6),CL6'D79PK5'
```

Example 4: Modify the owner of the profile defined in the previous example. Also, place the profile into warning mode, and remove the security label. Delete the DFP segment, in case it exists.

Function code = ADMN_ALT_DS

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(5)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(18),CL18'''DEPT06.TEST.DATA'''
BFLD2   DC CL8'GENERIC',CL1'Y',AL2(0)
BFLD3   DC CL8'OWNER',CL1'Y',AL2(5),CL5'MIKEO'
BFLD4   DC CL8'SECLABEL',CL1'N',AL2(0)
BFLD5   DC CL8'WARNING',CL1'Y',AL2(0)
DSEG    DC CL8'DFP',CL1'N',AL2(0)
```

Example 5: Display the contents of a generic data set profile and list the catalogued data set names for which it offers protection. Also, display the DFP segment.

Function code = ADMN_LST_DS

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(2)
BSEG    DC CL8'BASE',CL1'Y',AL2(2)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(11),CL11'''IBMUSER.*'''
BFLD2   DC CL8'DSNS',CL1'Y',AL2(0)
DSEG    DC CL8'DFP',CL1'Y',AL2(0)
```

Example 6: Delete the data set profile from example 4.

Function code = ADMN_DEL_DS

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(2)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(18),CL18'''DEPT06.TEST.DATA'''
BFLD2   DC CL8'GENERIC',CL1'Y',AL2(0)
```

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Permit group FINANCE to the DATASET profile named 'CORP.SALES.*' with READ access.

```
* First, define the request header
HEADER  DS 0H
        DC AL1(7)         Length of class
        DC CL8'DATASET'   Class name
        DC AL1(0)         Reserved byte
        DC AL2(0)         Not used on input
        DC AL2(1)         Number of segments (BASE only)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC CL1'Y'         Flag byte - Y - create segment
        DC AL2(3)         Field count - 3
* First BASE segment field entry
BFLD1   DC CL8'PROFILE'   Profile name - required!
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(14)        Length of field data
        DC CL14'''CORP.SALES.*'''   Field data
* Second BASE segment field entry
BFLD2   DC CL8'ID'        Id field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(7)         Length of field data
        DC CL7'FINANCE'   Field data
* Third BASE segment field entry
BFLD3   DC CL8'ACCESS'    Access field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(4)         Length of field data
        DC CL4'READ'      Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_PERMIT

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(3)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(14),CL14'''CORP.SALES.*'''
BFLD2   DC CL8'ID',CL1'Y',AL2(7),CL7'FINANCE'
BFLD3   DC CL8'ACCESS',CL1'Y',AL2(4),CL4'READ'
```

Example 3: Permit the SALES group with UPDATE access to the DATASET in the previous example, but only when logged on to a specific terminal.

Function code = ADMN_PERMIT

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(4)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(14),CL14'''CORP.SALES.*'''
BFLD2   DC CL8'ID',CL1'Y',AL2(5),CL5'SALES'
BFLD3   DC CL8'ACCESS',CL1'Y',AL2(6),CL6'UPDATE'
BFLD4   DC CL8'WHENTERM',CL1'Y',AL2(8),CL8'TERMID01'
```

Example 4: Remove the access list entry for the FINANCE group.

Function code = ADMN_PERMIT

```
HEADER  DC AL1(7),CL8'DATASET',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(3)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(14),CL14'''CORP.SALES.*'''
BFLD2   DC CL8'ID',CL1'Y',AL2(7),CL7'FINANCE'
BFLD3   DC CL8'DELETE',CL1'Y',AL2(0)
```

Example 5: Reset the access list for the BPX.SUPERUSER profile in the FACILITY class.

Function code = ADMN_PERMIT

```
HEADER  DC AL1(7),CL8'FACILITY',AL1(0),AL2(0),AL2(1)
BSEG    DC CL8'BASE',CL1'Y',AL2(2)
BFLD1   DC CL8'PROFILE',CL1'Y',AL2(13),CL13'BPX.SUPERUSER'
BFLD2   DC CL8'RESET',CL1'Y',AL2(0)
```

SETROPTS administration: For the ADMN_ALT_SETR function code, the function-specific parameter list is mapped as follows:

Table 24. Parameter list mapping for SETROPTS administration

Offset	Length	Description
0	10	Reserved
10	2	Output offset to the segment or field entry in error, relative to the start of the Parm_list. Only applies when an "Invalid Request" error is returned to the caller.
12	2	Number of RACF profile segments. The value is 1, for base segment only.
14	*	Start of first segment entry

By convention, the BASE segment is used to specify field information for SETROPTS.

See “Segment and field entry mappings” on page 357 for segment and field entry information.

The output from this function is either command output or error messages. See “Command output message block mapping” on page 98 for the output format.

For tables defining the field names and their usage, see “SETROPTS administration” on page 383.

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Set various password policy controls.

Function code = ADMN_ALT_SETR

```
* First, define the request header
HEADER  DS 0H
        DC CL10''         Unused
        DC AL2(0)         Not used on input
        DC AL2(1)         Number of segments (BASE only)
* First segment entry - BASE
BSEG    DC CL8'BASE'      BASE segment entry
        DC AL1(0)         Flag byte - ignored
        DC AL2(4)         Field count - 4
* First BASE segment field entry
BFLD1   DC CL8'HISTORY'   Password history field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(2)         Length of field data
        DC CL2'14'        Field data
* Second BASE segment field entry
BFLD2   DC CL8'REVOKE'    Password revoke field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(1)         Length of field data
        DC CL1'4'         Field data
* Third BASE segment field entry
BFLD3   DC CL8'WARNING'   Password warning field
```

```
        DC CL1'Y'         Flag byte - Y - boolean value
        DC AL2(2)         Length of field data
        DC CL2'10'        Field data
* Fourth BASE segment field entry
BFLD4   DC CL8'RULE1'     Password rule field
        DC CL1'Y'         Flag byte - Y - create field
        DC AL2(10)        Length of field data
        DC CL10'3:6 A*NV*A' Field data
```

Example 2: This is the same as example 1, but is shown in "rows", where a single line represents the request header, and individual segment and field entries. This convention will be used from this point on.

Function code = ADMN_ALT_SETR

```
HEADER  DC CL10'',AL2(0),AL2(1)
BSEG    DC CL8'BASE',AL1(0),AL2(4)
BFLD1   DC CL8'HISTORY',CL1'Y',AL2(2),CL2'14'
BFLD2   DC CL8'REVOKE',CL1'Y',AL2(1),CL1'4'
BFLD3   DC CL8'WARNING',CL1'Y',AL2(2),CL2'10'
BFLD4   DC CL8'RULE1',CL1'Y',AL2(10),CL10'3:6 A*NV*A'
```

Example 3: This example refreshes in-storage profiles for a RACLISTed class, in this case, the FACILITY class.

Function code = ADMN_ALT_SETR

```
HEADER  DC CL10'',AL2(0),AL2(1)
BSEG    DC CL8'BASE',AL1(0),AL2(2)
BFLD1   DC CL8'RACLIST',CL1'A',AL2(8),CL8'FACILITY'
BFLD2   DC CL8'REFRESH',CL1'Y',AL2(0)
```

Example 4: Add two classes to the list of active classes, remove two classes from the list of GENLISTed classes, and activate the setting which prevents the command issuer's user ID from being added to a new profile's access list (NOADDCREATOR).

Function code = ADMN_ALT_SETR

```
HEADER  DC CL10'',AL2(0),AL2(1)
BSEG    DC CL8'BASE',AL1(0),AL2(3)
BFLD1   DC CL8'CLASSACT',CL1'A',AL2(14),CL14'UNIXPRIV FSSEC'
BFLD2   DC CL8'GENLIST',CL1'D',AL2(13),CL13'DASDVOL FIELD'
BFLD3   DC CL8'ADDCREAT',CL1'N',AL2(0)
```

Command output message block mapping

The following mapping applies to all function codes which execute a RACF TSO command in the RACF subsystem address space. The output could represent command output (e.g. LISTUSER), or error and informational messages from a command.

If a RACF command runs and does not return any output, the Out_message_strings parameter will be zero. Otherwise, **R_admin** will set Out_message_strings with a pointer to a command output structure. This structure consists of a linked list of blocks, each of which consists of a block header followed by one or more message entries. A single message entry corresponds to a line of RACF command output.

The following mappings can be found in the IRRPCOMP mapping macro. Also, see the ADMN DSECT within the COMP data area in *z/OS Security Server RACF Data Areas*.

Table 25. Mapping of output message block

Offset	Length	Description
0	4	Next output messages block, or zero if no additional blocks follow
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 relative to the first message.
16	1	Start of the first message

Table 26. Format of each message entry

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

Profile extract functions

Note:

1. This section does not include ADMN_XTR_PWENV (extract a password envelope), ADMN_XTR_PPENV (extract a password phrase envelope), and ADMN_XTR_SETR (SETROPTS extract).
2. A REXX interface to the profile extract functions is available. This program, named IRRXUTIL, is designed to be invoked by REXX in problem state, and converts the output of an **R_admin** extract request to a set of REXX stem variables. For more information, refer to *z/OS Security Server RACF Macros and Interfaces*.

This set of function codes allows you to extract RACF profile information from USER, GROUP, CONNECT, and general resource profiles. Note that CONNECT is a logical class that returns the details of a specific user-to-group connection. The extract functions return the entire profile; you cannot specify a list of segments, or fields, to be returned. You can, however, request that only the BASE segment be returned.

The output (returned in the Out_message_strings parameter) consists of a header followed by a series of segment descriptors, which are followed in turn by a series of field descriptors, such as:

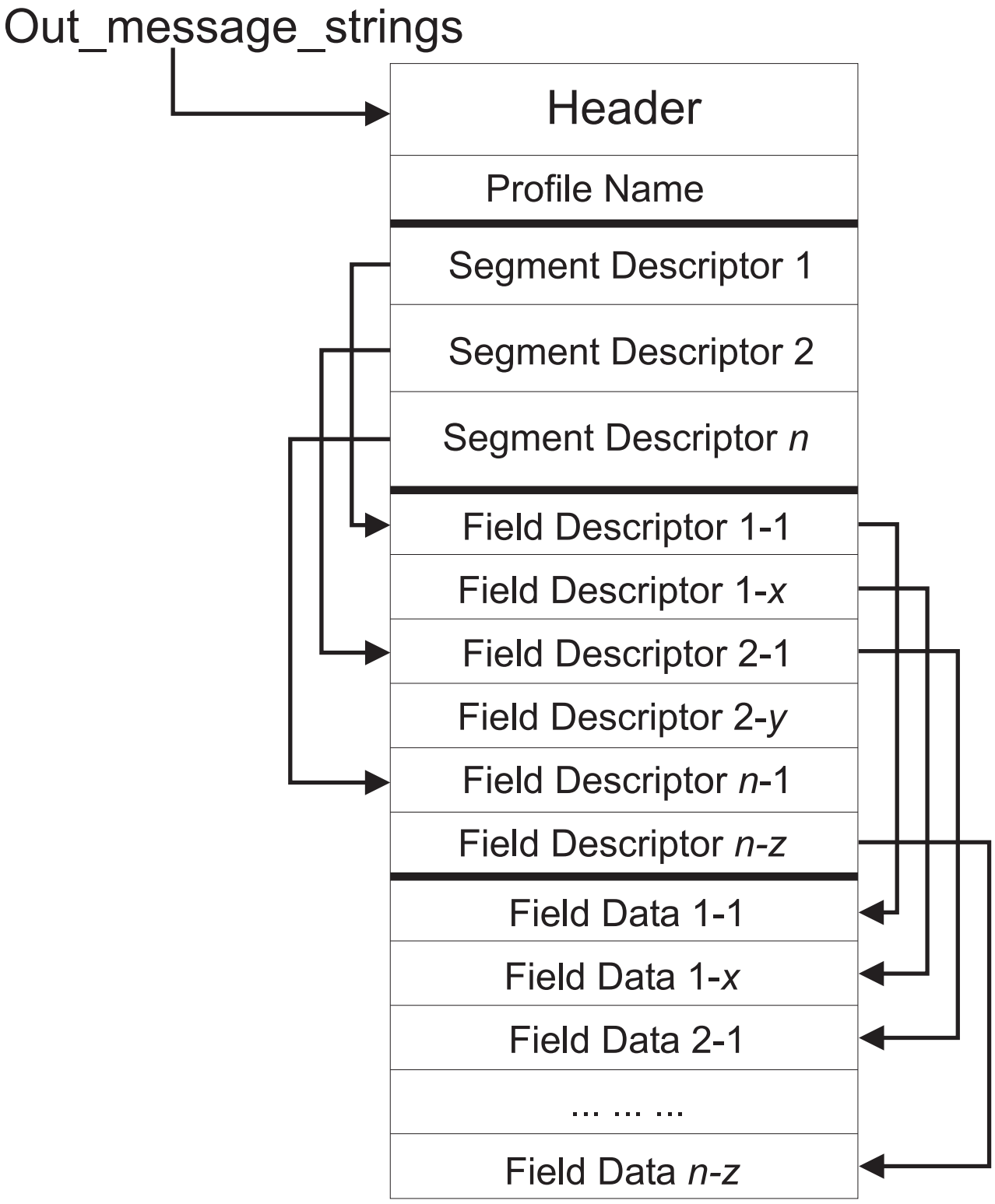

Figure 1. Extract functions output

Note: These segment and field descriptors are not the same format as the segment and field entries described above for the update functions and SETROPTS extract. The term descriptor is used in order to highlight this distinction.

The header identifies the total length and subpool of the output block, the profile and class name of the data being returned, and the number of segments (and hence segment descriptors) which exist for the profile. Each segment descriptor contains the number of field descriptors being returned for that segment, and an offset to the first field descriptor for that segment. Each field descriptor contains the length of, and offset to, the data for the field. All of the field descriptors for a given segment will be contiguous. The mappings for this structure can be found in the IRRPCOMP macro.

The field descriptors return character data, not the raw data as it exists within the RACF database. The data is returned exactly as it is expected on input to a RACF command, or to the **R_admin update** functions. Also, data is returned for some profile fields even though they cannot be directly altered by a RACF command. For most of these fields, the data is returned in the same format which is displayed by the associated RACF listing command. The exception is for date fields. All dates are returned in *mm/dd/yy* format, to maintain consistency with the way dates are expected as input to commands.

See Table 32 on page 107 for the fields associated with each profile type. For details on RACF profiles, segments, and fields, see *z/OS Security Server RACF Macros and Interfaces*. Only a subset of the defined fields are returned, as is detailed within the field tables referenced below. Mostly, the RACF command keywords map to database fields. See the *z/OS Security Server RACF Command Language Reference* for syntax rules regarding the format of the data expected by the commands, and the format of the data returned by the **R_admin** extract functions.

For extract user, extract group, and extract resource, there is a corresponding **extract-next** function code. These allow you to iteratively retrieve all RACF profiles of a given type for which you are authorized (this can be viewed as a hybrid between the LISTUSER * command, for example, and RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN). For each request, RACF extracts and returns the next profile that the caller is authorized to see (using the rules applied by the LISTUSER, LISTGRP, or RLIST command) after the profile that was specified in the parameter list header.

Note:

1. When using extract-next to return all general resource profiles for a given class, all the discrete profiles are returned, followed by the generic profiles. An output flag indicates if the returned profile is generic. A flag can be specified in the parameter list to request only the generic profiles in a given class. If only the discrete profiles are desired, check the output flag indicating whether the returned profile is generic. If it is, ignore the entry and terminate your extract-next processing.
2. For a caller with little RACF authority, a large amount of I/O may be performed against the RACF database until such a profile is located. This is no different from the LISTUSER *, LISTGRP *, or RLIST *class-name* * command. The IRR.RADMIN.LISTUSER, IRR.RADMIN.LISTGRP, and IRR.RADMIN.RLIST resources in the FACILITY class can be used to limit which users are allowed to use the extract and extract-next interfaces.
3. The following are characteristics of the extract functions:
   - Are not subject to the 4096 limit on output lines like the list functions described above.
   - Provide similar functions as ICHEINTY LOCATE and RACROUTE REQUEST=EXTRACT, but do not provide all of the functions provided by those interfaces.

- Run in the caller's address space, not in the RACF address space. Therefore, are faster than the **R_admin** list functions and do not contend with other functions, such as the RACF Remote Sharing Facility, for RACF subsystem address space resources.
- Are available to problem state callers, and require the same authority as do the corresponding RACF commands (e.g. LISTUSER), plus the FACILITY class check enforced by **R_admin**. Supervisor state callers can choose to bypass command authorization checking. The FACILITY check is not enforced for supervisor state callers by default. However, the caller can request that the check be performed.

The header mapping described below is used both as input, for the caller to identify the requested profile, and as output, for RACF to return the profile data.

Note: These are two separate pieces of storage and RACF will not modify the caller's input storage.

Note: All offsets below are relative to the start of the extract output block.

Table 27. Profile extract parameter list (input and output)

Offset	Length	Description
0	4	Eye catcher to aid in virtual storage dumps: 'PXTR'
4	4	Total length of the output buffer
8	1	Subpool of output buffer (specified by caller in Out_message_subpool parameter)
9	1	Parameter list version
10	2	Reserved
12	8	Class name (uppercase and padded with blanks). Class names cannot be abbreviated.
20	4	Length of profile name (maximum of 8 for users and groups; maximum of 17 for connects; maximum of 246 for general resources)
24	8	Reserved
32	4	Reserved
36	4	Flag word **Note:** RACF only propagates the "generic" flag (X'10000000') into the output version of the parameter list. When using extract-next, the caller must set the other flags as appropriate in the parameter list before calling R_admin for the next iteration.
	x'80000000'	Bypass command processor (e.g. LISTUSER) authorization checking. Available for supervisor state callers only.
	x'40000000'	Extract BASE segment only. By default, RACF will extract all profile segments, resulting in an I/O operation for each segment.
	x'20000000'	For supervisor state callers, enforce the FACILITY class authorization check. By default, the FACILITY check is bypassed for supervisor state callers.

Table 27. Profile extract parameter list (input and output) (continued)

Offset	Length	Description
	X'10000000'	For general resource requests only. **On input**: • For extract requests: return the profile which covers the input profile name if there is no exact match. By default, if there is no exact match on profile name, a "not found" return code combination (4/4/4) will be returned. Note that the class must be active, and SETROPTS GENERIC must be in effect for the class, in order for a generic match to be found. • For extract-next requests: return the next alphabetic generic profile in the class with respect to the input name (regardless of whether the profile is generic or discrete). To retrieve the first generic profile in a class, the caller can provide a resource name consisting of a single blank (X'40'). **On output:** indicates that the profile returned by RACF is generic. When using extract-next to cycle through profiles, the caller should not alter this bit.
	X'08000000'	Upper case the input name as appropriate. When the CLASS is USER, GROUP, CONNECT, or a general resource class which has been defined as CASE(UPPER), **R_admin** will upper-case the input name before attempting to extract it. This flag is ignored for extract-next requests.
	X'04000000'	Return only the profile name, without any of the profile data On return, the field at offset 40 containing the number of returned segments will be zero. If the "base-only" flag (x'40000000') is also specified, "name-only" will take precedence.
40	4	Number of segments (including base)
44	16	Reserved
60	0	The start of the profile name. For extract connect, the profile name is <userID>.<group>. On output, the first segment descriptor starts immediately after the profile name.

Table 28. Segment descriptor mapping

Offset	Length	Description
0	8	Segment name (uppercase and padded with blanks)
8	4	Flag word
12	4	Number of fields for this segment (a list field header together with all its subfields are considered one logical field).
16	4	Reserved
20	4	Offset to first field descriptor for this segment
24	16	Reserved
40	0	Start of next segment descriptor, or, if last segment descriptor, start of field descriptors

Table 29. Field descriptor mapping

Offset	Length	Description
0	8	Field name (uppercase and padded with blanks)
8	2	Field type
	'8000'x	Member of a repeat group
	'4000'x	Reserved
	'2000'x	Flag (boolean) field
	'1000'x	Repeat field header. This descriptor contains the number of occurrences of the repeat field, and the dimension of each repeat field (that is, the number of field descriptors comprising a single occurrence of the repeat field).
10	2	Reserved
12	4	Flag word
	'80000000'x	For boolean fields, this bit indicates the value of the field.
	'40000000'x	This is an output-only field.
16	4	For repeat field headers (type='1000'x): Number of occurrences of repeat group (e.g. 50 group connections) Otherwise: Length of field data
20	4	Reserved
24	4	For repeat field headers (type='1000'x): number of elements (subfields) within a repeat group occurrence (For example, each group connection in the user profile consists of 15 subfields) Otherwise: offset to field data
28	16	Reserved
44	0	Start of next field descriptor, or if last field descriptor, start of field data

On input: The caller needs to construct a header containing:

- A value of 0 in the parameter list version field
- The length of the profile in the profile name length field

The caller then places the profile name after the header.

For the **extract-next** functions, the caller can invoke RACF iteratively, using as input the block which RACF previously returned as output. For example:

1. Construct an input parameter list, as described above, where the profile name consists of a single blank character (X'40').
2. Set the Parm_list parameter to the address of the header just created.
3. Call IRRSEQ00.
4. Process the output returned by RACF in the Out_message_strings parameter. The output header, using the profile name offset, length, and value fields, will identify the profile which was extracted.
5. Set the Parm_list parameter to the address returned by RACF in the Out_message_strings parameter. Note that you might need to make some adjustments to the output returned by RACF to use it as an input parameter list. For example, if a supervisor state caller has turned on the “bypass authorization checking” or the “BASE only” flag in the previous call, he will need to turn them on again in the subsequent call.

6. Call IRRSEQ00 again.
7. On return, the storage now pointed to by Parm_list (previously returned by RACF) can be freed. You should be careful of virtual storage consumption issues if you do not immediately free this storage. That is, depending on the size of the output, and the number of times you are calling IRRSEQ00, your virtual storage consumption could add up to a significant amount.
8. Iterate (see step 4) until profiles are exhausted (SAF return code 4, RACF return code 4, RACF reason code 4), or until your purpose has been accomplished.

On output: The output storage is obtained in the subpool specified by the caller in the Out_message_subpool parameter. It is the responsibility of the caller to free this storage.

Note: Even though the profile name and the subpool are specified by the caller on input, that information, along with the class name, is also contained in the output block so that it is as self-descriptive as possible.

Segment and field descriptors will be returned as shown in the diagram above, following the header. Segment descriptors will only be returned for segments which exist within the profile, and which the requester is authorized to see.

For each segment descriptor, a set of field descriptors will be returned. There is no defined order in which fields are returned. The types of fields are:

- Boolean - A flag in the descriptor indicates whether the value is TRUE or FALSE. There is no data returned and the offset value will be 0.
- Character - A simple character string.
- Repeat - An array of character strings and/or booleans. These are described in more detail below.

Field data will only be returned if the requester is authorized to read the field. Also, if a field does not have a value in a given profile, usually no field descriptor will be returned. The exception to this rule is for multidimensional repeat fields as described below. The number of field descriptors must be the same for each occurrence of the repeat field, so that the entire field may be quickly skipped if you want, as described below. Therefore, if a given subfield within a repeat field occurrence has no value, for example, the revoke or resume date of a group connection in a USER profile, then a field descriptor will be present but the length of the data will be zero. A robust application should be written to expect this condition for any field descriptor, however, in case this behavior changes in the future.

Repeat fields: A repeat field, also known as a **list** field, is a multi-valued field. For repeat fields, a special field descriptor exists to act as a header to the subsequent repeating data. Such a field descriptor can be recognized by its field type value of '1000'x. This descriptor does not 'point' to field data, but, it indicates how many occurrences of the repeat group exist, and how many 'subfields' comprise a single repeat group occurrence. The information in this descriptor can be used to completely bypass the entire repeat field if the field is not of interest to the caller. Following this header are individual field descriptors for each subfield of each occurrence. These descriptors contain offsets to the actual data. For multidimensional repeat fields, the order of the subfields will be constant for every occurrence of the repeat field. See Table 31 on page 106 below.

Below, Table 30, is an example of a 1-dimensional repeat group, the user's class authority (CLAUTH). The user has CLAUTH for the FACILITY, UNIXPRIV, and OPERCMDS classes. The following is a schematic representation of how such a field would be represented:

Table 30. Repeat example 1

Field name	Value	Total number of occurrences	Dimension of repeat group
CLCNT	N/A	3	1
CLAUTH	FACILITY	N/A	N/A
CLAUTH	UNIXPRIV	N/A	N/A
CLAUTH	OPERCMDS	N/A	N/A

Group connection information in the GROUP profile is a slightly more complex example. The repeat group is a 2-dimensional array. The group has four connected users: JOE, LARRY, JIM, and MIKE.

Table 31. Repeat example 2

Field name	Value	Total number of occurrences	Dimension of repeat group
CONNECTS	N/A	4	2
GUSERID	JOE	N/A	N/A
GAUTH	JOIN	...	...
GUSERID	LARRY		
GAUTH	CONNECT		
GUSERID	JIM		
GAUTH	CREATE		
GUSERID	MIKE		
GAUTH	USE		

Consider an example where an application extracts a group profile and is interested only in the OWNER field, and the OWNER field is returned by **R_admin** after the group connections. The application steps through field descriptors, maintaining a pointer, or offset, to the current field descriptor. Upon encountering a repeat field, the application would add to its current pointer, or offset the length of a field descriptor, for the repeat field header, plus the length of a field descriptor, for each subfield of a repeat group occurrence, times the number of subfields in an occurrence, 2 in this example, times the number of occurrences, 4 in this example.

A repeat field is treated as a single logical field. That is, the field count within a segment descriptor is only taking the list header field into account, regardless of the dimension of the field, or the number of occurrences within it. If you are iterating through the fields associated with a given segment, and are decrementing the count field in order to determine when that segment's fields have been exhausted, decrement the count by only 1 for any list field which you encounter.

Table 32. Field tables by profile type

Profile type	Field tables
USER	User field tables in "User administration" on page 359
GROUP	Group field tables in "Group administration" on page 368
CONNECT	Connect field tables in "Group connection administration" on page 370
General resource	General resource tables in "General resource administration" on page 372

Examples*:* The following examples are not coding samples. Rather, they demonstrate how to construct the input parameter list for a number of requests.

Example 1: Extract the contents of the IBMUSER profile. Request only the base segment, and request that authorization be bypassed (caller is in supervisor state).

Function code = ADMN_XTR_USER

```
HEADER  DS 0H
        DC CL20''         Unused (note class name optional)
        DC AL4(7)         Length of user ID
        DC CL12''         Unused
        DC XL4'C0000000'  Flag word - BASE only+skip auth
        DC CL20''         More unused
NAME    DC CL7'IBMUSER'   User ID name
```

Example 2: Start an iterative extract by using extract-next starting with a profile name of blank.

Function code = ADMN_XTR_NEXT_USER

```
HEADER  DS 0H
        DC CL20''         Unused (note class name optional)
        DC AL4(1)         Length of "user ID"
        DC CL12''         Unused
        DC XL4'00000000'  Flag word - none specified
        DC CL20''         More unused
NAME    DC CL1' '         "User ID" name of 1 blank
```

Example 3: Extract group profile FINANCE. Take the defaults of returning all existing segments and performing LISTGRP authorization.

Function code = ADMN_XTR_GROUP

```
HEADER  DS 0H
        DC CL20''         Unused (note class name optional)
        DC AL4(7)         Length of group name
        DC CL12''         Unused
        DC XL4'00000000'  Flag word - none specified
        DC CL20''         More unused
NAME    DC CL7'FINANCE'   Group name
```

Example 4: Extract the connection information for user BRUSCHI in group PATRIOTS.

Function code = ADMN_XTR_CONNECT

```
HEADER  DS 0H
        DC CL20''         Unused (note class name optional)
        DC AL4(16)        Length of connect name
        DC CL12''         Unused
        DC XL4'00000000'  Flag word - none specified
        DC CL20''         More unused
NAME    DC CL16'BRUSCHI.PATRIOTS'  Connect name
```

Example 5: Extract the information for the IRR.RADMIN.RLIST resource in the FACILITY class. Request that the covering generic profile be returned if a discrete does not exist.

Function code = ADMN_XTR_RESOURCE

```
HEADER DS 0H
       DC CL12''                Unused
       DC CL8'FACILITY'         Class name - required
       DC AL4(16)               Length of resource name
       DC CL12''                Unused
       DC XL4'10000000'         Flags - return matching profile
       DC CL20''                More unused
NAME   DC CL16'IRR.RADMIN.RLIST' Resource name
```

SETROPTS reporting functions

This pair of function codes allows you to retrieve RACF SETROPTS settings in either of two formats:

1. In **unload** format, you get the data in the same format in which it is reported by the SMF Unload Utility (IRRADU00).
2. In SETROPTS **extract** format, the output is formatted in the same manner as the input to the SETROPTS alter function.

   Note:

   a. Although conceptually similar, this is a different format from the output produced by the profile extract functions.
   b. The IRRXUTIL program provides a REXX interface to the SETROPTS **extract** format. The IRRXUTIL program is designed to be invoked by REXX in problem state, and converts the output of an **R_admin** extract request to a set of REXX stem variables. For more information, refer to *z/OS Security Server RACF Macros and Interfaces*.

The unload format is available only to supervisor state callers. The extract format is available to both supervisor and problem state callers.

No input parameter list is required for these functions. However, additional options may be specified by providing a parameter list.

Table 33. ADMN_XTR_SETR parameter list

Offset	Length	Description
0	4	Request flags
	x'80000000'	Apply FACILITY class check to caller. Caller will require READ access to IRR.RADMIN.SETROPTS.LIST
	x'40000000'	Apply SETROPTS LIST command authority checks to caller.
4	10	Reserved

The mapping of the output message block returned by **R_admin** for the ADMN_XTR_SETR and ADMN_UNL_SETR function codes is as follows. The output storage is obtained in the subpool specified by the caller in the Out_message_subpool parameter of IRRSEQ00, and is returned in the Out_message_strings parameter. It is the responsibility of the caller to free this storage:

Table 34. Output message block

Offset	Length	Description
0	4	Eye catcher to aid in virtual storage dumps: 'RXTR' or 'RUNL'
4	4	Total length of the output buffer
8	4	Reserved
12	2	Number of segment entries for ADMN_XTR_SETR (always 1 for the BASE segment), or number of record types returned for ADMN_UNL_SETR
14	0	Start of the BASE segment entry (for extract) or the first record entry (for unload)

For ADMN_XTR_SETR, the output consists of a single segment entry for the base segment, followed by field entries for each of the supported input fields documented in “SETROPTS administration” on page 383.

Note: Not all of the fields are returned, and fields that are not returned are noted in the field table.

There is no defined order in which fields are returned. The segment and field entry for ADMN_XTR_SETR uses the standard ADMN_USRADM_SEGENTRY and ADMN_USRADM_FLDENTRY mappings used by other **R_admin** functions. See “Segment and field entry mappings” on page 357 for details.

For ADMN_UNL_SETR, the output data is mapped using the following mapping for each unloaded record type, the number of which is contained in ADMN_EXTRACT_NUM. There is a single record type of "RACFINIT" to describe the basic RACF options. Following this record is a series of records of type "CLASNAME". There are as many "CLASNAME" records as there are classes defined in the class descriptor table (CDT). These include classes supplied by IBM (ICHRRCDX), installation-defined classes defined in ICHRRCDE, and classes in the dynamic class descriptor table. Columns 44 through 51 of each record identify the name of the class that the record describes. See *z/OS Security Server RACF Macros and Interfaces* for detailed mappings of these record types.

Note: For ADMN_UNL_SETR, R_admin does not fill in the time-written, date-written, and SMF system ID fields.

Table 35. Output data

Offset	Length	Description
0	8	The SMF data unload record type (see *z/OS Security Server RACF Macros and Interfaces*)
8	4	The length of an individual record of this type
12	4	The number of records of this type
16	8	Reserved

Table 35. Output data (continued)

Offset	Length	Description
24	*	The start of the first record of this type

Password and password phrase envelope retrieval

RACF can be configured to create password or password phrase envelopes for eligible users. An envelope resides within a user's profile, and contains an encrypted version of the user's current password or password phrase. The password or password phrase can be recovered in clear text by authorized processes (for example, a password synchronization application).

The ADMN_XTR_PWENV function code of R_admin retrieves an encrypted password envelope. The ADMN_XTR_PPENV function code of R_admin retrieves an encrypted password phrase envelope. The ADMN_XTR_PWENV function code and the ADMN_XTR_PPENV function code of R_admin provide the only interfaces by which you can retrieve an encrypted envelope. An encrypted envelope is not returned as part of an R_admin extract function against a user profile, although there is a boolean field that indicates the existence of an envelope for that user. Neither RACROUTE REQUEST=EXTRACT nor ICHEINTY LOCATE returns an envelope field in a usable form. Password and password phrase envelope retrieval requires the RACF subsystem address space to be running. It is limited to supervisor state callers, and, in addition, requires access to a FACILITY class profile.

For a description of the RACF password and password phrase enveloping function, see *z/OS Security Server RACF Security Administrator's Guide*. The input parameter list format is the same as for the user-related update functions. See Table 22 on page 84 for more information. For envelope retrieval, the input parameter list is simply used to identify the target user ID for the retrieval, therefore, no segment/field information is required.

The following table is the mapping of the output message block returned by **R_admin** for the ADMN_XTR_PWENV and ADMN_XTR_PPENV function codes. The output storage is obtained in the subpool specified by the caller in the Out_message_subpool parameter.

Table 36. Output message block

Offset	Length	Description
0	4	Eye catcher to aid in virtual storage dumps: "RXPW" for password envelope and "RXPP" for password phrase envelope.
4	1	Subpool of this block
5	3	Total length of the output buffer
8	0	Start of the encrypted envelope

Contents of the encrypted password or password phrase 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 or password phrase 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.

Note that the RACF certificate is required to verify the signature of the envelope. See the description of the RACF password and password phrase enveloping function in *z/OS Security Server RACF Security Administrator's Guide* on how the recipient can obtain the RACF certificate.

For the structure of the envelope, refer to the PKCS #7 standard, Version 1.5, which can be obtained at http://www.rsasecurity.com/rsalabs/node.asp?id=2124.

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 arbitray 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.

The following is the ASN.1 notation describing the password payload as constructed by RACF:

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

The following is the ASN.1 notation describing the password phrase payload as constructed by RACF:

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

Version is the version number of the payload. For the password payload, it is set to 1 if the password has been changed to lowercase, or 2 if the password appears as entered. For the password phrase payload, it is set to 1.

Expired will be true if the new password or password phrase is marked as expired at the time of the change (for example, an ALTUSER command is used to change

the password or password phrase without specifying the NOEXPIRED operand). If Expired is true, the password or password phrase must be changed the next time the user logs on.

Password is the value of the new password. If the mixed case password support is not active, the password is in lowercase. If it is active, the password case is as entered.

Passwordphrase is the value of the new password phrase, with case preserved.

Changetime is a character string of decimal numbers in the format *yyyymmddhhiiss.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 or password phrase 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 and password phrases 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 log on to that system using the same keystrokes he used to change his password in RACF, the password will be rejected.

R_audit (IRRSAU00): Provide an audit interface

Function

The **R_audit** service provides an audit interface for functions that need to write an audit record for a condition where an audit by a security check service is not sufficient.

This service fills in the base part of the record and some standard relocate sections based on the function code in the CRED and the defined input parameters.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSAU00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, CRED,
               ALET, File_Identifier_1,
               ALET, FSP1,
               ALET, File_deleted_flag,
               ALET, File_Identifier_2,
               ALET, FSP2
              )
```

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 words 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.

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.

CRED
The name of the CRED structure for the current file system syscall.

File_Identifier_1
The name of a 16-byte area containing a unique identifier of the file identified by the old (or only) pathname specified on the syscall.

FSP1
The name of the IFSP for the old (or only) file.

File_deleted_flag
For system calls that can cause a file to be deleted, the address of a byte containing a flag:

- 0 - the last link was not removed.
- 1 - the last link was removed for a file. The file is deleted.

File_Identifier_2
For system calls that create a new file name. If the "new" file existed, this is the name of a 16-byte area containing a unique identifier of the "new" file.

FSP2
The name of the IFSP for the new file.

Return and reason codes

IRRSAU00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	24	The audit function code is not valid.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service can be called by the MVS BCP, by a z/OS UNIX file system, or by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by an z/OS UNIX server.
2. IRRSAU00 tests whether auditing is required and, if so, builds and writes an audit record. The record built contains data from the calling process's security attributes (USP) and from the input CRED, the input IFSPs, and the input parameters. The content depends on the function being audited, as determined from the CRED_audit_function_code.
3. See *z/OS Security Server RACF Macros and Interfaces* for tables describing the data included in audit records, the data included in each event record, and syscalls that cause the event records to be written.

Related services

None

R_auditx (IRRSAX00 or IRRSAX64): Audit a security-related event

Function

The **R_auditx** callable service generates an SMF type 83 audit record to record a security-related event and optionally issues a message indicating an authorization failure to the console. This service is intended for use only by components of z/OS that require the ability to log security-related events. The subtype must be defined to RACF or an equivalent security product. Contact IBM to determine the appropriate SMF type 83 record subtype to use.

Requirements

Authorization:
Any PSW key in supervisor state or problem state

Dispatchable unit mode:
Task

Cross memory mode:
PASN = HASN

AMODE:
31 or 64

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
FRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. The words containing ALETs must be in the primary address space. The Num_parms parameter 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 Num_parms parameter must indicate the number of parameters in the parameter list.

RACF authorization

For callers not running in system key or supervisor state, the use of **R_auditx** is authorized by the resource IRR.RAUDITX in the FACILITY class. The caller must be running with a RACF user or group that has at least READ authority to this resource. If the class is inactive, or the resource is not defined, only callers running with a system key or in supervisor state may use the **R_auditx** service.

Format

For 31-bit callers, every parameter address is 4 bytes in length:

```
CALL IRRSAX00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Num_parms,
               ACEE_ALET, ACEE
               Parm_ALET, Option_word,
                          Link_value,
                          Attributes,
                          Component,
                          FMID,
                          Subtype,
                          Event,
                          Qualifier,
                          Class,
                          Resource,
                          Log_string,
                          Relocate_count,
                          Relocate_ptr,
                          Message_count,
                          Message_ptr                    )
```

For 64-bit callers, every parameter address is 8 bytes in length. The module name is different but has the same number and order of parameters:

```
CALL IRRSAX64 (Work_area,...)
```

For C callers (31 or 64 bit), include sample irrc.h, which defines the rauditx prototype with the same number and order of parameters:

```
   #include <irrc.h>
int rauditx (Work_area,...)
```

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 words containing the ALETs must be in the primary address space.

SAF_return_code
The name of a 4–byte area in which the SAF router returns the SAF return code.

RACF_return_code
The name of a 4–byte area in which the service routine stores the return code.

RACF_reason_code
The name of a 4–byte area in which the service routine stores the reason code.

Num_parms
The name of a 4-byte area containing the number of parameters in the parameter list, including the Num_parms parameter. This parameter must be in the primary address space. It must be initialized to 26.

ACEE_ALET
The name of a 4-byte area containing the ALET for the ACEE pointed to named by the ACEE_ptr parameter. The 4-byte area must be in the primary address space.

ACEE
The name of an area containing the ACEE belonging to the RACF user that should appear in the log record. An ACEE may only be specified by a caller in supervisor state or system key. The ACEE must begin with eyecatcher "ACEE". Otherwise, the area must contain binary zeros in the first 4 bytes. When the area contains binary zeros, RACF uses the task-level ACEE if found, or the address space ACEE.

Parm_ALET
The name of a 4-byte area containing the ALET for the remaining parameters in the parameter list and any data areas referenced by parameter list pointers. The word containing the ALET must be in the primary address space.

Option_word
The name of a 4-byte area containing binary zeros. This area is reserved for future use.

Link_value
The name of an 8-byte area containing a value used to mark related SMF records. Since a single event may result in multiple calls to R_auditx for logging, you can logically link the records by specifying a common value such as a time stamp. Otherwise, fill the area with binary zeros.

Attributes
The name of a 4-byte area containing flags set by the caller. Possible attribute values are the following:

x'80000000'
Event Result. Used to indicate if the event was a success or a failure. Success if flag is set. Failure if flag is not set.

x'40000000'
Authentication Event. Use logging defaults for authentication events described in the usage notes.

x'20000000'
Authorization Event. Use logging defaults for authorization events described in the usage notes.

x'10000000'
Always log successes.

x'08000000'
Always log failures.

x'04000000'
Never log successes.

x'02000000'
Never log failures.

x'01000000'
Check warning mode.

Set any combination of attributes flags except the following pairs that directly conflict with each other:

```
'Authentication Event' & 'Authorization Event'
'Always log successes' & 'Never log successes'
'Always log failures' & 'Never log failures'
'Never log success' & 'Never log failures'
```

Refer to the usage notes for additional information about the priority of these flags for logging determination.

Component
The name of an area that consists of a 4-byte length field followed by character data. The character data is the name of the product or component calling the **R_auditx** service. The length represents the length of the character data. The maximum length of the data is 255. The component is required, therefore, the length must be greater than zero.

FMID
The name of a 7-byte area containing the FMID of the product or component calling the **R_auditx** service.

Subtype
The name of a 4-byte integer with the SMF type 83 record subtype assigned to the component. The value may range from 2 to 32767, but should match the subtype assigned to the component. Assigned subtypes are:

Subtype	z/OS component
2	Enterprise identity mapping
5	WebSphere® Application Server
6	Tivoli Key Lifecycle Manager (TKLM)

Event
The name of a 4-byte integer which the caller initializes with the event code. The value may range from 1 through 255.

Qualifier
The name of a 4-byte integer which the caller initializes with the event code qualifier. The value may range from 0 through 255.

Class
The name of an 8-byte area containing a RACF class name. If not specified, the area must contain all blanks. Otherwise, the class name is assumed to have the following characteristics:

- Left justified
- Padded to the right with blanks
- Specified in uppercase
- A static IBM class name, a statically defined installation class name, or a dynamically defined installation class name
- A general resource class

The class cannot be USER, GROUP, or DATASET. It must also be active and RACLISTed.

Resource

The name of an area that consists of a 4-byte length field followed by a resource name covered by a profile defined in the RACF class specified above. The length represents the length of the resource name. The maximum length is 246. The resource name is ignored if the length is zero. Ensure the letter case of the resource matches that defined for profiles in the class. For the RAUDITX class, profiles must be uppercase so the resource name must be folded to uppercase before being passed to this service. Some classes preserve case sensitivity for profiles and corresponding resource names should not be folded. Refer to *z/OS Security Server RACF Macros and Interfaces* for more information about class definitions.

Log_string
The name of an area that consists of a 4-byte length field followed by character

data to be written with the audit information. The length represents the length of the character data. The maximum length of the log string is 255. If character data is not specified, the length must equal zero.

Relocate_count
The name of a 4-byte area containing the number of relocate sections. The maximum number of relocate sections is 512. The minimum is 0.

Relocate_ptr
The name of an area containing the address of an array of relocate sections. For 31-bit callers, this area is 4-bytes long. For 64-bit callers, this area is 8-bytes long. The area is 4-bytes long. This parameter is ignored when the Relocate_count parameter is zero. The number of entries in the array must equal the value in the Relocate_count. The relocate is not added to the log record when the length is zero. When the length is greater than zero, the relocate data pointer must not be zero. An array entry for a 31 bit caller is:

Dec Offset	Hex Offset	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	16	RAUX_RELOCATE	A row in the array of relocate sections. It describes a single field value.
0	(0)	FIXED	2	RAUX_RELO_TYPE	The relocate section type. The value must not be less than 100 or greater than 65535.
2	(2)	FIXED	2	*	Reserved
4	(4)	FIXED	4	RAUX_RELO_LEN	The length of the data portion of the relocate section. The sum of the relocate lengths plus an additional four bytes for each relocate field must not exceed 20480 bytes.
8	(8)	FIXED	4	*	Reserved
C	(C)	ADDRESS	4	RAUX_RELO_DATA_PTR	The address of the data for the relocate section

An array entry for a 64 bit caller is:

Dec Offset	Hex Offset	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	16	RAUX64_RELOCATE	A row in the array of relocate sections. It describes a single field value.
0	(0)	FIXED	2	RAUX64_RELO_TYPE	The relocate section type. The value must not be less than 100 or greater than 65535.
2	(2)	FIXED	2	*	Reserved

Dec Offset	Hex Offset	Type	Len	Name(Dim)	Description
4	(4)	FIXED	4	RAUX64_RELO_LEN	The length of the data portion of the relocate section. The sum of the relocate lengths plus an additional four bytes for each relocate field must not exceed 20480 bytes.
8	(8)	ADDRESS	8	RAUX_RELO_DATA_PTR	The address of the data for the relocate section

Fields marked 'Reserved' must be filled with binary zeros.

Message_count
The name of a 4-byte integer containing the number of message segments that form the message. The maximum number of segments is 16. The service issues no message if Message_count is 0.

Message_ptr

The name of an area containing zero or the address of an array. For 31-bit callers, this area is 4-bytes long. For 64-bit callers, this area is 8-bytes long. This parameter is ignored when the Message_count parameter is zero. Otherwise, the number of entries in the array must equal the Message_count value.

The array contains the length and addresses of message segments are combined to form a message that is directed to the security console and the job log. Each message segment text, referenced by the segment pointer in each array entry, should be composed of valid uppercase characters that will display properly at the console. The length for each message segment must not exceed 70 characters. The first message segment should begin with a component message identifier of 15 characters or less.

For each array entry, the message segment is not included when its segment length is zero. When the segment length is greater than zero, the segment pointer must not be zero.

An array entry for a 31-bit caller is:

Dec Offset	Hex Offset	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	16	RAUX_SEGMENT	One phrase in the message
0	(0)	FIXED	4	*	Reserved
4	(4)	FIXED	4	RAUX_SEG_LEN	The length of the message segment. The value must not exceed 70.
8	(8)	FIXED	4	*	Reserved
12	(12)	ADDRESS	4	RAUX_SEG_PTR	The address of message segment

Fields marked 'Reserved' must be filled with binary zeros.

An array entry for a 64-bit caller is:

Dec Offset	Hex Offset	Type	Len	Name(Dim)	Description
0	(0)	STRUCTURE	16	RAUX64_SEGMENT	One phrase in the message
0	(0)	FIXED	4	*	Reserved
4	(4)	FIXED	4	RAUX64_SEG_LEN	The length of the message segment. The value must not exceed 70.
8	(8)	ADDRESS	8	RAUX64_SEG_PTR	The address of message segment

Fields marked 'Reserved' must be filled with binary zeros.

Return and reason codes

IRRSAX00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful and an audit log record written.
0	0	4	The service completed normally, but no audit log record was written.
0	0	8	The service was successful, an audit log record was written, and the authentication or authorization check should pass because the resource is in WARNING mode.
4	0	0	RACF is not installed.
8	8	4	Caller is not system authorized and does not have READ authority to the IRR.RAUDITX profile in the FACILITY class.
8	8	8	The class is not defined, not active, or not RACLISTed.
8	8	12	A covering profile for the resource was not found.
8	12	8	The Num_parms parameter is in error.
8	12	10	ACEE parameter must be zero for callers running in problem state with problem key.
8	12	12	The Option_word parameter is not zero.
8	12	14	The Attributes parameter is in error. One or more of the specified attributes is not supported or conflicts.
8	12	15	The Component parameter length is zero or exceeds the maximum allowed.

SAF return code	RACF return code	RACF reason code	Explanation
8	12	17	The value specified for the Subtype parameter is not supported.
8	12	18	The value specified for the Event parameter is not supported.
8	12	19	The value specified for the Qualifier parameter is not supported.
8	12	20	The value, USER, GROUP, or DATASET, specified for the Class parameter is not supported.
8	12	21	The Resource parameter length exceeds the maximum allowed.
8	12	22	The Log_string length exceeds the maximum allowed.
8	12	23	The Relocate_count parameter exceeds the maximum allowed.
8	12	24	The Relocate_ptr parameter or the data it references is in error. 1. The pointer cannot be zero when the Relocate_count parameter is greater than zero. 2. An unsupported value was specified for a structure field. Remember that reserved fields must be zero. 3. The sum of the relocate lengths plus an additional four bytes for each relocate exceeded 20480 bytes.
8	12	25	The Message_count parameter exceeds the maximum.
8	12	26	The Message_ptr parameter or the data it references is in error. 1. The pointer cannot be zero when the Message_count parameter is greater than zero. 2. An unsupported value was specified for a structure field. Remember that reserved fields must be zero.
8	16	8	Internal error during RACF processing.

Usage notes

1. This service is intended for use by components of z/OS that require the ability to log security-related events.
2. The use of RACF audit controls is optional. A product or component that uses its own mechanism for determining when to audit can pass in the appropriate 'always log' or 'never log' attributes for successes and failures to prevent the service from continuing to the RACF checks.

3. The service determines when to log by comparing the event result attribute flag (x'80000000') with auditing controls settings passed through logging attribute flags or found in RACF definitions. The following table describes the sequence of checks that the service makes. At any step for which 'log' or 'no log' is determined, the service ends the checking process.

Settings	Action
Input Attribute Checks:	
a. The caller specified “authentication event”	log if failure, otherwise, skip to RACF checks
b. The caller specified '“authorization event”	skip to RACF checks
c. The caller specified “always log successes”	log if success
d. The caller specified “always log failures”	log if failure
e. The caller specified “never log successes”	no log if success
f. The caller specified “never log failures”	no log if failure
RACF Checks:	
g. UAUDIT was specified for the user. ALTUSER RACFU00 UAUDIT	log
RACF Checks if Class Specified:	
h. SETROPTS LOGOPTIONS(NEVER(class))	no log
i. SETROPTS LOGOPTIONS(ALWAYS(class))	log
j. SETROPTS LOGOPTIONS(SUCCESSES(class))	log if success
k. SETROPTS LOGOPTIONS(FAILURES(class))	log if failure
l. If a covering resource profile is found in the class, the AUDIT and GLOBALAUDIT settings are checked. RALTER class profile AUDIT(ALL | SUCCESSES | FAILURES | NONE)	log if success or failure with corresponding AUDIT or GLOBALAUDIT settings
Default if prior steps do not determine logging	
m. Always	no log

4. The SMF Type 83 subtype must be one defined in *z/OS Security Server RACF Macros and Interfaces*. Contact IBM if you need to define a subtype for your component.
5. The SMF Type 83 record contains the following types of information:
   a. The record subtype
   b. The name of the component and release that detected the event
   c. The event, the event code qualifier, RACF release, and an indicator if the event was a success or a failure
   d. The reasons this event was logged. Did the calling application require it? Was SETROPTS LOGOPTIONS used? Was there a profile in the specified class covering the resource name associated with the event that had AUDIT or GLOBALAUDIT settings? Was UAUDIT specified for the end user? Many of these conditions could be valid for auditing the event, but only the first check that passes in the list of checks above will be listed as the reason for auditing.
   e. Information about the user
   f. Additional data about the event

6. The class must be a general resource class defined in the class descriptor table (CDT). It must be active and RACLISTed. The length of the resource name cannot exceed the length defined for profiles in the class. Generics may be used if they are enabled for the class.
7. The service will issue two multiline messages if the event is logged and the caller specified event result 'Failure' and the Message_count greater than zero.

   The first message is generated directly from the message segment data referenced by the Message_ptr parameter. The message text should be uppercase and contain component-specific details about the event. The first line should begin with a component message ID of 15 characters or less, as in the following example:

   ```
   COMP123I THE USER JT@CAFE.COM ATTEMPTED TO UPDATE THE PRICE LIST WITHOUT PERMISSION.
   THE TRANSACTION RECORD IS TJV04123.
   ```

   The second message includes associated information collected by **R_auditx** as in the following example:

   ```
   IRRY000I A SECURITY-RELATED EVENT HAS BEEN LOGGED.
      COMPONENT(EXPRESSO SERVICES)
      EVENT(AUTHORIZATION) TYPE(FAILURE) MESSAGE(COMP123I)
      USER(JOE) GROUP(JAVACLUB) NAME(JOE T. COFFEE)
      CLASS(RAUDITX) PROFILE(COMP.RESOURCE.*)
   ```

   The messages are routed to the security and programmer consoles (route codes 9 & 11) with the job status descriptor (6).
8. This service does not make use of the access levels (ALTER, READ, CONTROL, UPDATE) that may be specified with the RDEFINE AUDIT, RALTER AUDIT GLOBALAUDIT, and SETROPTS LOGOPTIONS commands.
9. An invoker of this service may call it several times, each time requiring a slightly different set of relocate sections. To allow these types of applications to reuse a single definition of the arrays of relocate sections, the service will skip over any relocate section that has a length of zero.
10. **R_auditx** supports warning mode. Warning mode is a feature of RACF that allows installations to try out security policies. Installations can define a profile with the WARNING attribute. When RACF performs an authorization check using the profile, it will log the event (if there are audit settings), issue a message to the operator's console if the check failed, and allow the authorization check to proceed. The messages and log records can be monitored to ensure the new policy is operating as expected before putting the policy into production by turning off WARNING attribute.

    A component which wants to provide a similar capability must first perform its authentication or authorization check. When it calls **R_auditx** to perform the logging, it must indicate to **R_auditx** using an attribute bit on the call that it is able to bypass authentication or authorization failures (for example, undoing the failure). **R_auditx** will proceed through its list of checks for logging. If it reaches the point where it searches for a profile and detects that the WARNING mode indicator is on for the profile, then **R_auditx** will log the event as indicated by the AUDIT and GLOBALAUDIT options, issue any messages to the operator's console for failures, and then return to the caller with a 0/0/8 return code. The component can then allow the security event to continue as if it succeeded.

Related services

None

R_cacheserv (IRRSCH00): Cache services

Function

The **R_cacheserv** SAF callable service provides a mechanism for the storage and retrieval of security relevant information from a cache.

Function codes X'0001' through X'0005' only: In addition, the entire cache can be automatically hardened to the security product (RACF) database, and restored as needed.

Requirements

Authorization:
Any PSW key in supervisor state or problem state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have a FRR active

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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. Use the **NumParms** parameter to indicate the number of parameters specified.

RACF authorization

Function codes X'0001' through X'0005' only: For callers not running in system key or supervisor state, the use of R_cacheserv is authorized by the resource IRR.RCACHESERV.*cachename* in the FACILITY class. The application server must be running with a RACF user or group that has at least READ authority to this resource. READ allows the application server to utilize the **Fetch** function, x'0004', while UPDATE authority provides the capability to use all the functions.

Function code X'0006' only: For callers not running in system key or supervisor state, the use of R_cacheserv is authorized by the resource

IRR.RCACHESERV.ICTX 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. READ allows the application server to utilize the **Retrieve**, and **RetrieveAppl**, and **RemoveExpired** options (X'0003', X'0004', and X'0006'), while UPDATE authority provides the capability to use all of the options.

Function code X'0007' only: For callers not running in system key or supervisor state, the use of R_cacheserv is authorized by the resource IRR.RCACHESERV.ICRX in the FACILITY class. The application server must be running with a RACF user or group ID at the address space level that has at least READ authority to this resource, and the FACILITY class must be active and RACLISTed. READ authority allows the application server to utilize the RetrieveAppl and Remove options (X'0002' and X'0003'), while UPDATE authority provides the capability to use all of the options.

All function codes: 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_cacheserv service. Changes to the RACF user or group ID's authorization to the facility class profile will not take effect until the job step task invoking the R_cacheserv service ends and a new one is started.

Format

```
CALL IRRSCH00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
                     ParmALET,
                     NumParms,
                     Function_code,
                     Option,
                     Version,
                     Version_length,
                     Cache_name,
                     Record_name_ptr,
                     Record_name_length,
                     Data_ptr,
                     Data_length,
                     Data_timeout,
                     Source_ptr,
                     Source_length,
                     Reference_timeout,
                     Reference_userID,
                     Reference,
                     Subpool,
                     ACEE_ALET,
                     ACEE,
                     ICRX_area,
                     ICRX_length
                     )
```

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 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.

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 that must be in the primary address space and contains the ALET for the remaining parameters, not including the ACEE_ALET and ACEE parameters.

NumParms
The name of a fullword containing the number of remaining parameters in the parameter list, including the NumParms parameter. This number must be 19 for function code X'0006' or 21 for function code X'0007'. However, for compatibility with prior releases, invokers who only use function codes X'0001' through X'0005' can continue to specify a NumParms value of 10.

Function_code
The name of a half-word (2-byte) area containing the function code.

Function codes X'0001' through X'0005', function code X'0006', and function code X'0007' are not compatible. In other words, function code X'0006' option X'0003' cannot be used to retrieve a record that was added to a cache using function code X'0002' and function code X'0004' cannot be used to fetch a record that was stored using function code X'0006' option X'0001'. The function code has one of the following values:

X'0001'-Start a new cache.
The cache is created and the caller is now ready to start adding records. No one but the caller has access to the cache. The cache is not made available to other callers at this time.

X'0002'-Add a record to the new cache.
The caller must provide a name for this record and the data associated with the record. The caller calls this function multiple times, once per record to complete the cache. Only the caller of function code X'0001' may call function code X'0002'. A caller of function code X'0002' who is not the same task as the one who called function code X'0001', will not be allowed to add a record to the cache.

X'0003'-End cache creation.
Only the caller of function code X'0001' is allowed to end the creation of the cache. A caller of function code X'0003' who is not the same task as the one who called function code X'0001', will not be allowed to end cache creation. This function is further defined by the Option parameter.

If the **Option** parameter is X'0001':

- The cache is made available, with system-wide name/token service, to callers of function x'0004' so they can retrieve records from the cache. Any previous cache of the same *Cache-name* is deleted.
- The cache is hardened to the RACF database if the CACHECLS class is active and if a profile exists in that class with a profile name identical to the Cache_name provided as input to R_cacheserv.

If the **Option** parameter is X'0002', discard the new cache and leave the existing cache intact. This is used if the calling application determines that something is wrong with the new cache or encounters an error while creating it.

X'0004'-Fetch information from cache.
Retrieve information from the cache. This function is further defined by the **Option** parameter.

If the **Option** parameter is X'0001' and the cache already exists, the requested (by name) record is retrieved for the caller (or is not found). If the cache does not exist yet (for example, right after IPL), a new cache can be automatically restored and populated with records that were hardened to the RACF database the last time someone called function code x'0003'. The cache is restored if the CACHECLS class is active and the cachename_ddd_nnnnn profiles containing the cache contents exist. **Data** and **Data_Length** are updated.

If the **Option** parameter is X'0002', then retrieve the version number of the existing in-storage cache. If no in-storage cache exists, retrieve the version from the database-hardened copy of the cache. Do not restore the cache from the database in this case. **Version** and **Version_Length** are updated.

X'0005'-Delete the cache.

This function is further defined by the **Option** parameter.

If the **Option** parameter is X'0001', then delete the in-storage cache only.

If the **Option** parameter is X'0002', then delete the hardened database copy of the cache only.

If the **Option** parameter is X'0003', then delete both the hardened database copy and in-storage cache.

X'0006'-Manage a read/write cache.

The function code X'0001-X'0005' cache is essentially a read-only cache. All of the data is added to the cache before the cache is made available for retrieval and if data needs to be changed a new cache must be created. This function, X'0006', provides support for a read/write cache. Multiple callers can store and retrieve cache data at the same time.

The read/write cache is not hardened to the RACF database.

This function is further defined by the **Option** parameter.

If the **Option** parameter is X'0001', then **Store** data in the read/write cache and return a reference to the data.

- The caller can store the following records:
  - A source record (**Source_ptr** parameter). The caller is responsible for building the source record. It will be stored as binary data.
  - An application data record (**Data_ptr** parameter). Like the source record, the caller is responsible for building the application data record and it will be stored as binary data. The caller must also specify the name of the record (**Record_name_ptr** parameter), so that the application data record can be found on a Locate request. The caller can also store a null application data record, by specifying a record name and a **Data_length** of zero.

  If the caller is in supervisor state or system key and specifies a valid ACEE on a **Store** request (option X'0001'), RACF will use the specified

ACEE to build and store an application data record name and an application data record. The **Record_name_ptr, Record_name_length, Data_ptr,** and **Data_length** parameters will be ignored. See the description of the ACEE parameter for more information. The ACEE parameter is ignored unless specified by a caller in supervisor state or system key.

If no ACEE, source record, application data record name, and application data record are specified on a **Store** request (**Source_length, Record_name_length,** and **Data_length** are all zeros), RACF will build and store the application data record name and the application data record using the task-level ACEE if found, or the address space ACEE.

See the descriptions of the **Record_name_ptr** and **Data_ptr** parameters for more information about the structure of the application data record name and application data record built by RACF.

- When the **Store** is successful, a cache reference is returned. A cache reference consists of both a reference value (**Reference** parameter) and an associated reference user ID (**Reference_userID** parameter). The cache reference can be used to retrieve the source record and the application data record (**Retrieve** option), retrieve just the application data record (**RetrieveAppl** option), or to remove the source record (**Remove** option).

  A cache reference can only be used to retrieve or remove data one time. When it has been specified, the cache reference is no longer valid.

  RACF recommends that customers put the Integrated Cryptographic Service Facility (ICSF) CSNBRNG module in the link pack area (LPA) or the modified link pack area (MLPA) so that it can be used for generating reference values (**Reference** parameter). If RACF cannot find CSNBRNG in LPA or MLPA, it will default to using a less efficient software pseudo random number generator (PRNG) for generating reference values.
- The caller can use the **Reference_timeout** parameter to control the lifespan of the cache reference, by specifying the time interval in which it can be used. When this interval has expired, the reference can only be used to **Remove** the source record. A retrieve request (**Retrieve, RetrieveAppl**) will fail when an expired cache reference is specified. The **RemoveExpired** option also uses the reference timeout interval to determine when source records can be removed from the cache. If not specified, a default timeout value will be assumed.
- The caller can use the **Data_timeout** parameter to control the lifespan of the application data record, by specifying the time interval in which it can be used. When this interval has expired, the record will not be found on a **Locate** request. The **RemoveExpired** option also uses the data timeout interval to determine when application data records can be removed from the cache. If not specified, a default timeout value will be assumed.
- If the cache does not exist yet, the first **Store** request will cause it to be created.

If the **Option** parameter is X'0002', then **Locate** an application data record in the read/write cache, store the specified source record, and return a reference to the data.

- The caller must specify the application data record name that was specified when the application data record was stored (**Record_name_ptr** parameter).

- **Locate** uses the timeout interval that was specified when the application data record was stored (**Data_timeout** parameter). When this interval has expired, the record will not be found.
- The caller must specify a source record (**Source_ptr** parameter) and, optionally, a reference timeout interval (**Reference_timeout** parameter). If not specified, a default timeout value will be assumed.
- When the **Locate** is successful, the source record is stored in the cache and a cache reference is returned. A cache reference consists of both a reference value (Reference parameter) and an associated reference user ID (**Reference_userID** parameter). The cache reference can be used to retrieve the source record and the application data record(**Retrieve** option), retrieve just the application data record (**RetrieveAppl** option), or to remove the source record (**Remove** option).

  A cache reference can only be used to retrieve or remove data one time. When it has been specified, the cache reference is no longer valid.

  RACF recommends that customers put the Integrated Cryptographic Service Facility (ICSF) CSNBRNG module in the link pack area (LPA) or the modified link pack area (MLPA) so that it can be used for generating reference values (**Reference** parameter). If RACF cannot find CSNBRNG in LPA or MLPA, it will default to using a less efficient software pseudo random number generator (PRNG) for generating reference values.

If the **Option** parameter is X'0003', then **Retrieve** data from the read/write cache.

- The caller must specify a cache reference. A cache reference consists of both a reference value (**Reference** parameter) and an associated reference user ID (**Reference_userID** parameter).

  A cache reference can only be used to retrieve data one time. When it has been specified, the cache reference is no longer valid.
- **Retrieve** uses the reference timeout interval that was specified on a **Store** or **Locate** request (**Reference_timeout** parameter). When this interval has expired, **Retrieve** will fail.
- When a **Retrieve** is successful, the following data can be returned:
  - A source record (**Source_ptr** parameter). The **Source_length** parameter contains the length of the record and the **Source_ptr** parameter contains the name of a fullword containing the address of the source record. Storage for the record is obtained in the subpool specified by the caller (**Subpool** parameter) and the caller is responsible for freeing it. A length of zero indicates that no source record was returned.
  - An application data record name (**Record_name_ptr** parameter). The **Record_name_length** parameter contains the name of a fullword containing the address of the application data record name. Storage for the application data record name is obtained in the subpool specified by the caller (**Subpool** parameter) and the caller is responsible for freeing it. A length of zero indicates that no application data record name was returned.
  - An application data record (**Data_ptr** parameter). The **Data_length** parameter contains the length of the record and the **Data_ptr** parameter contains the name of a fullword containing the address of the application data record. If the caller specifies an area and length that is large enough to contain the application data record, RACF will return the record in that area. If the caller specifies an area and length that is too small to contain the record, RACF will obtain storage in

the specified subpool (**Subpool** parameter) and will return the address of the application data record along with SAF return code 0, RACF return code 0, and RACF reason code 16. The caller can also just specify a length of zero to ask that RACF obtain storage without returning the 0/0/16 return code. **Data_length** will be set to the actual length of the application data record retrieved. The caller is responsible for freeing all application data record output areas, either provided by the caller or obtained by RACF. A length of zero indicates that no application data record was returned.

If the **Option** parameter is X'0004', then retrieve application data (**RetrieveAppl**) from the read/write cache.

- The caller must specify a cache reference. A cache reference consists of both a reference value (**Reference** parameter) and an associated reference user ID (**Reference_userID** parameter).

  A cache reference can only be used to retrieve data one time. When it has been specified, the cache reference is no longer valid.
- **RetrieveAppl** uses the reference timeout interval that was specified when the application data record was stored (**Reference_timeout** parameter). When this interval has expired, no data will be found.
- When a **RetrieveAppl** is successful, the following data can be returned:
  - An application data record name (**Record_name_ptr** parameter). The **Record_name_length** parameter contains the name of a fullword containing the address of the application data record name. Storage for the application data record name is obtained in the subpool specified by the caller (**Subpool** parameter) and the caller is responsible for freeing it. A length of zero indicates that no application data record name was returned.
  - An application data record (**Data_ptr** parameter). The **Data_length** parameter contains the length of the record and the **Data_ptr** parameter contains the name of a fullword containing the address of the application data record. If the caller specifies an area and length that is large enough to contain the application data record, RACF will return the record in that area. If the caller specifies an area and length that is too small to contain the record, RACF will obtain storage in the specified subpool (**Subpool** parameter) and will return the address of the application data record, along with SAF return code 0, RACF return code 0, and RACF reason code 16. The caller can also just specify a length of zero to ask that RACF obtain storage without returning the 0/0/16 return code. **Data_length** will be set to the actual length of the application data record retrieved. The caller is responsible for freeing all application data record output areas, either provided by the caller or obtained by RACF. A length of zero indicates that no application data record was returned.

If the **Option** parameter is X'0005', then **Remove** a source record from the read/write cache.

- The caller must specify a cache reference. A cache reference consists of both a reference value (**Reference** parameter) and an associated reference user ID (**Reference_userID** parameter).
- When a source record is retrieved (option X'0003'), RACF marks it as expired and eligible for explicit or internal **RemoveExpired** processing, so caller invocation of Remove processing is optional.

If the **Option** parameter is X'0006', then remove all expired records from the read/write cache (**RemoveExpired**).

- **RemoveExpired** uses the timeout values specified when the data was stored to determine which records are expired, so they are no longer valid. Source records are removed based on the reference timeout interval (Reference_timeout parameter) and application data records are removed based on the data timeout interval (**Data_timeout** parameter).
- RACF does cache cleanup on a regular basis by doing an internal **RemoveExpired**, so caller invocation of **RemoveExpired** processing is optional.

If the **Option** parameter is X'0007', then **Destroy** the read/write cache.

- The read/write cache is destroyed. All references to data in the cache are no longer valid.

  Under normal circumstances, **the caller should have no reason to destroy the read/write cache.** This option is provided only as a means for a customer, working with IBM support, to recover from a catastrophic cache error.
- A subsequent request to **Store** data will cause a new cache to be created.

  The value of the **Option** parameter also determines how other parameters will be used for **Function_code** X'0006'. For example, on a **Store** (**Option** X'0001') request, **Data_ptr** is an input parameter and on a **Retrieve** (**Option** X'0003') request, it is an output parameter. See the description of **Function_code** X'0006' for more information.

X'0007'-Manage an extended read/write cache.

The cache for function codes X'0001-X'0005' is essentially a read-only cache. All of the data is added to the cache before the cache is made available for retrieval, and if data needs to be changed a new cache must be created. Function code X'0006' provides support for a read/write cache of distributed user information. Function code X'0007' provides support for an extended read/write cache containing both RACF and distributed user information. Multiple callers can store and retrieve cache data at the same time. The read/write cache is not hardened to the RACF database.

This function is further defined by the **Option** parameter.

If the **Option** parameter is X'0001', then locate the record if present. If the record is not present, **Store** data in the read/write cache, and return an extended context reference to the data.

RACF recommends that customers put the Integrated Cryptographic Service Facility (ICSF) CSNBRNG module in the link pack area (LPA) or the modified link pack area (MLPA) so that it can be used for generating reference values (the ICRXREFR portion of the ICRX). If RACF cannot find CSNBRNG in LPA or MLPA, it will default to using a less efficient software pseudo random number generator (PRNG) for generating reference values.

- If the caller is in supervisor state or system key and specifies a valid ACEE on a **Store** request (option X'0001'), RACF will use the specified ACEE to locate, or to build and store, an application data record name and an application data record. The **Source_length**, **Record_name_length**, **Data_length** parameters, and their associated data parameters will be ignored. See the description of the **ACEE** parameter for more information. The **ACEE** parameter is ignored unless specified by a caller in supervisor state or system key.

- If no ACEE is specified on a **Store** request, or it is ignored because of the caller's state, RACF will build and store the application data record name and the application data record using the task-level ACEE (if found), or the address space ACEE.
- If the locate or the store is successful, an extended cache reference will be returned in the **ICRX_area** parameter, and the length of the ICRX in the **ICRX_length** parameter. The ICRX will contain a cache reference consisting of a reference value and an associated reference user ID, as well as the RACF user ID and IDID value. The ICRX can be used to retrieve the application data (**RetreiveAppl** option) or to remove the reference (**Remove** option). The structure of the ICRX data area is detailed in *z/OS Security Server RACF Data Areas*.
- A cache reference can only be used to retrieve or remove data one time. When it has been specified, the cache reference is no longer valid.
- The **Reference_timeout** and **Data_timeout** parameters are ignored for function code X'0007' since these values are set internally by RACF.

If the **Option** parameter is X'0002', then retrieve application data (**RetrieveAppl**) from the read/write cache. The caller must specify an extended cache reference (**ICRX_area** parameter) containing a cache reference.

- When a RetrieveAppl is successful, an application data record (**Data_ptr** parameter) is returned. The **Data_length** parameter contains the length of the record and the **Data_ptr** parameter contains the name of a fullword containing the address of the application data record. If the caller specifies an area and length that is large enough to contain the application data record, RACF will return the record in that area (ALET-qualified through the ParmALET parameter). If the caller specifies an area and length that is too small to contain the record, RACF will obtain storage in the specified subpool (**Subpool** parameter) and will return the address of the application data record, along with SAF return code 0, RACF return code 0, and RACF reason code 16. The caller can also just specify a length of zero to ask that RACF obtain storage without returning the 0/0/16 return code. If RACF obtains storage for the return of the data record, the storage will be obtained in the primary address space. **Data_length** will be set to the actual length of the application data record retrieved. The caller is responsible for freeing all application data record output areas, either provided by the caller or obtained by RACF. A length of zero indicates that no application data record was returned.

If the **Option** parameter is X'0003', then **Remove** a context reference record from the read/write cache.

- The caller must specify an extended cache reference (**ICRX_area** parameter) containing a cache reference.
- Remove processing is optional since records will be marked expired after a period of time and will become eligible for internal removal processing.
- **Option** parameter X'0003' can also be used to remove an application data record from the read/write cache by marking it no longer valid. The caller must specify an **ICRX_area** parameter containing the IDID information needed to identify the record, without a cache reference. Once marked not valid, attempts to retrieve the record using outstanding context references will no longer find the record, and it will not be used for subsequent store requests.

If the **Option** parameter is X'0004' (**store** and return **reusable** ICRX), then processing proceeds as with Option X'0001', but the returned ICRX will be marked as being reusable. This ICRX will be valid for multiple Option X'0002' **RetrieveAppl** calls until it times out after an hour of inactivity.

If the **Option** parameter is X'0005' (validate the input ICRX), we will validate the input ICRX and the IDID included in it. This function is intended to validate the user-built ICRX that is provided by the application to a subsystem like CICS®, not the completed ICRX that is returned by RACF to the caller of R_cacheserv as is done with Option X'0001', which contains the RACF user-id in ICRXUSER and the ICR. The caller must be in supervisor state or system key.

The following fields will be validated:

- **ICRX** fields:
  - **ICRXID** – The value must be the literal 'ICRX'
  - **ICRXVERS** – The value must be greater than or equal to **ICRXVR01** and less than or equal to the current version – **ICRXCURV** (currently set to 2)
  - **ICRXOFFN** – The number of offsets must be 3 (for versions 1 and 2 of the ICRX)
  - **ICRXFLGS** – The value of this flag-byte must be 0
  - **ICRXLEN** – The value must be greater than or equal to the length of the ICRX header – **ICRXHLN**, and less than or equal to the length of the ICRX provided in the R_cacheserv parameter list
  - **ICRXICRO** – Must be 0
  - **ICRXDIDO** – Must be equal to the length of the ICRX header – **ICRXHLN**
  - **ICRXUSRO** – Must be 0
- **IDID** fields:
  - **IDIDID** – The value must be the literal 'IDID'
  - **IDIDVERS** – The value must be greater than or equal to **IDIDVR01** and less than or equal to the current version – **IDIDCURV** (currently set to 1)
  - **IDIDOFFN** – The number of offsets must be 5 (for version 1 of the IDID)
  - **IDIDHSHN** – The three high order bits of this bit string must be off (0), and for any IDID section not specified (offset is 0), the corresponding bit must be off (0)
  - **IDIDSP** – The value must be the same as the value in **ICRXSP**
  - **IDIDLEN** – The value must be greater than the length of the IDID header – **IDIDHLN**, and the sum **ICRXDIDO** + **IDIDLEN** must be less than or equal to the length of the ICRX – **ICRXLEN**
  - **IDIDOFF1** – Must be equal to the length of the IDID header – **IDIDHLN**
  - **IDIDOFF2** – Must be 0
  - **IDIDOFF3**, **IDIDOFF4**, and **IDIDOFF5** – Must be either 0, or (if non-zero) must be contained within the IDID and be in sequential order.
- **IDID Section1** fields:
  - **IDID1OF1** – Must be equal to the length of the IDID Section1 header

- **IDID1UDL** – Must be greater than or equal to 1 and less than or equal to the value indicated by **RCVTDNL**
- **IDID1OF2** – Must be equal to the sum of the length of the IDID Section1 header, the length of **IDID1UDL**, and the value specified in **IDID1UDL**
- **IDID1RL** – Must be greater than equal to 1 and less than or equal to the value indicated by **RCVTRL**

Option
The name of a half word containing an option value for the specified function code.

See *Parameter Usage* for the function codes to which the **Option** parameter applies. Valid option values and their effect on the specified function code are described above under the **Function_code** parameter.

Version
The name of a data field containing the version identifier of this cache. Length is specified in **Version_length**.

When hardening to the RACF database, the cache version identifier is compared with the version already hardened. If the versions are the same, the new cache will not be hardened.

Note: This parameter applies only to read-only caches. Otherwise, it is ignored.

Version_length
Name of a fullword containing the length of the version of the cache being created or the amount of storage the caller has provided for retrieval. It's value must be between 1 and 255. When **Function_code** X'0004' (**Fetch**) and **Option** X'0002' are specified, Version_length will be set to the actual length of the cache version identifier. If return and reason codes indicate that an insufficient length value was specified, resulting in a return of partial data, use the updated **Version_length** value to obtain a larger results area and resubmit the request.

Note: This parameter applies only to read-only caches. Otherwise, it is ignored.

Cache_name
The name of a data field containing the name of the cache. The name must be 6 bytes in length and must start with an R. The remaining 5 characters may consist of any combination of the characters A through Z, numbers 0 through 9, and the special characters @, #, and $. Names starting with RZ are reserved for IBM use. Lowercase characters will be folded to uppercase.

The cache name is used internally to isolate multiple caches from each other. It is also used to generate names of the dataspace(s) that will form the cache, and also the profile names used to harden the contents of the cache to the RACF database as profiles in the CACHECLS class.

Note: This parameter applies only to read-only caches. The function code X'0006' and X'0007' read/write cache is not hardened to the RACF database.

Record_name_ptr
Function codes X'0002' (**Add**) and X'0004' (**Fetch**) with **Option** X'0001' only. Name of the address of the record name to be added or retrieved. The length is specified in Record_name_length.

Function code X'0006' (**Manage a read/write cache**) with Options X'0001' (**Store**), X'0002' (**Locate**), X'0003' (**Retrieve**), and X'0004' (**RetrieveAppl**) only: Name of the address of the application data record name to be stored (option X'0001') or located (option X'0002'), or name of a fullword where the address of the application data record name is to be placed during retrieval (options X'0003' and X'0004').

When an application data record is specified on a **Store** (option X'0001') request, the caller must also specify the application data record name. The caller is responsible for obtaining and freeing storage for the application data record name. The length is specified in **Record_name_length**. When an application data record is not specified on a **Store** request (**Data_length** is zero), specification of an application data record name is optional.

If the caller is in supervisor state or system key and specifies a valid ACEE on a **Store** request (option X'0001'), RACF will use the specified ACEE to build and store an application data record name and an application data record. The **Record_name_ptr, Record_name_length, Data_ptr,** and **Data_length** parameters will be ignored. See the description of the ACEE parameter for more information.

If no ACEE (see the parameter description for more information), source record, application data record name, and application data record are specified (**Source_length, Record_name_length,** and **Data_length** are all zeros) on a **Store** request, RACF will build and store the application data record name and the application data record using the task-level ACEE if found, or the address space ACEE.

An application data record name built by RACF has the same structure as the Identity Context Extension (ICTX) block described in *z/OS Security Server RACF Data Areas*. RACF fills in the ICTXID, ICTXVERS, and values for:

- **Authenticated user name:** If no authenticated user name is available, RACF sets ICTXUSRL to zero. When an application data record name is built from an ACEE, RACF sets ICTXUSRL and the value pointed to by ICTXUSR@ from the ACEE user ID values (ACEEUSRL and ACEEUSRI).
- **Registry name:** If no registry name is available, RACF sets ICTXREGL to zero. When an application data record name is built from an ACEE, RACF sets ICTXREGL and the value pointed to by ICTXREG@ to the name of the local RACF registry. The registry name is taken from an in-storage copy of the LOCALREGISTRY field in the IRR.ICTX.DEFAULTS.sysid profile or IRR.ICTX.DEFAULTS profile in the LDAPBIND class.
- **Host name:** If no host name is available, RACF sets ICTXHSTL to zero. When an application data record name is built from an ACEE, RACF sets ICTXHSTL and the value pointed to by ICTXHST@ to the SMF SYSID. The system identifier is the 4-character value specified for the SID parameter of the SMFPRMxx member of SYS1.PARMLIB. See *z/OS MVS Initialization and Tuning Reference* for additional information about SMFPRMxx.
- **Authentication mechanism object identifier (OID):** If no authentication mechanism object identifier (OID) is available, RACF sets ICTXMCHL to zero. When an application data record name is built from an ACEE, RACF sets ICTXMCHL and the value pointed to by ICTXMCH@ to OID value "1.3.18.0.2.33.1".

When an application data record name is built from an ACEE, ICTXLEN is set to the value that will be used for **Record_name_length**.

When **Locate** (option X'0002') is specified, the caller must provide an application data record name and is responsible for obtaining and freeing storage for the application data record name. The length is specified in **Record_name_length**.

When **Retrieve** (option X'0003') or **RetrieveAppl** (option X'0004') is specified, RACF obtains storage for the application data record name in the subpool provided by the **Subpool** parameter. Storage will be obtained in the primary address space. The caller is responsible for freeing this storage. **Record_name_length** will be set to the actual length of the record name retrieved.

This parameter is ignored for function code X'0007'.

Record_name_length
Function codes X'0002' (**Add**) and X'0004' (**Fetch**) with **Option** X'0001' only. Name of a fullword containing the length of the record name to be stored or retrieved. Valid values are 1 to 8192.

Function code X'0006' (**Manage a read/write cache**) with Options X'0001' (**Store**), X'0002' (**Locate**), X'0003' (**Retrieve**), and X'0004' (**RetrieveAppl**) only: Name of a fullword containing the length of the application data record name to be stored (option X'0001') or located (option X'0002'), or the length of the application data record name retrieved (options X'0003' and X'0004').

When **Store** (option X'0001') is specified, a zero length indicates that the application data record name has not been specified. Otherwise, valid values are 1 to 8192.

When **Retrieve** (option X'0003') or **RetrieveAppl** (option X'0004') is specified, **Record_name_length** will be set to the actual length of the record name retrieved.

This parameter is ignored for function code X'0007'.

Data-Ptr
Function codes X'0002' (**Add**) and X'0004' (**Fetch**) with **Option** X'0001' only. Name of the address of the data to be stored in cache (X'0002'), or name of the address where data are to be placed during retrieval (X'0004'). Length is specified in **Data_length**.

Function code X'0006' (**Manage a read/write cache**) with Options X'0001' (**Store**), X'0003' (**Retrieve**), and X'0004' (**RetrieveAppl**) only: Name of the address of the application data record to be stored (option X'0001'), or name of a fullword where the address of the application data record is to be placed during retrieval (options X'0003' and X'0004').

- When an application data record is specified on a **Store** (option X'0001') request, the length is specified in **Data_length**.
- If the caller is in supervisor state or system key and specifies a valid ACEE on a **Store** request (option X'0001'), RACF will use the specified ACEE to build and store an application data record name and an application data record. The **Record_name_ptr, Record_name_length, Data_ptr,** and **Data_length** parameters will be ignored. See the description of the ACEE parameter for more information.
- If no ACEE (see the parameter description for more information), source record, application data record name, and application data record are specified (**Source_length, Record_name_length,** and **Data_length** are all zeros) on a **Store** request, RACF will build and store the application data record name and the application data record using the task-level ACEE if found, or the address space ACEE.

An application data record built by RACF has the following structure:

Table 37. Structure of an application data record built by RACF

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	12	ApplData	Application data record.
0	0	UNSIGNED	4	ApplVersion	Format version. For this release, this field is set to X'00000001'.
4	4	UNSIGNED	4	ApplUserIDLen	Length of a security manager (RACF) user ID, or zero if the user ID is omitted. The user ID has a maximum length of 8 bytes.
8	8	UNSIGNED	4	ApplUserIDOff	Offset of user ID.
12	C	CHARACTER	*		Data related to the length and offset pairs begins here.

The length of the application data record is equal to the length of the ApplData structure plus the sum of the lengths of the data related to the length and offset pairs. When an application data record is built from an ACEE, RACF will set ApplUserIDLen and the value pointed to by ApplUserIDOff from the ACEE user ID values (ACEEUSRL and ACEEUSRI).

- When **Retrieve** (option X'0003') or **RetrieveAppl** (option X'0004') is specified, the caller can either specify the address where the output application data record is to be placed, or RACF will obtain an area and return the address of that area. If the caller specifies an area and length that is large enough to contain the application data record, RACF will return the record in that area, ALET qualified using the ParmALET. If the caller specifies an area and length that is too small to contain the record, RACF will obtain storage in the specified subpool (**Subpool** parameter) and will return the address of the application data record along with SAF return code 0, RACF return code 0 and RACF reason code 16. The caller can also just specify a length of zero to ask that RACF obtain storage without returning the 0/0/16 return code. If RACF obtains storage for the return of the data record, the storage will be obtained in the primary address space. **Data_length** will be set to the actual length of the application data record retrieved. The caller is responsible for freeing all application data record output areas, either provided by the caller or obtained by RACF. A length of zero indicates that no application data record was returned.

Function code X'0007' (**Manage an extended read/write cache**). Name of a fullword where the address of the application data record is to be placed during retrieval (option X'0002').

- When **RetrieveAppl** (option X'0002') is specified, the caller can either specify the address where the output application data record is to be placed, or RACF will obtain an area and return the address of that area. If the caller specifies an area and length that is large enough to contain the application data record, RACF will return the record in that area (ALET-qualified through ParmALET parameter). If the caller specifies an area and length that is too small to contain the record, RACF will obtain storage in the specified subpool (**Subpool** parameter) and will return the address of the application

data record along with SAF return code 0, RACF return code 0 and RACF reason code 16. The caller can also just specify a length of zero to ask that RACF obtain storage without returning the 0/0/16 return code. If RACF obtains storage for the return of the data record, the storage will be obtained in the primary address space. **Data_length** will be set to the actual length of the application data record retrieved. The caller is responsible for freeing all application data record output areas, either provided by the caller or obtained by RACF. A length of zero indicates that no application data record was returned.

Data_length
Function codes X'0002' (**Add**) and X'0004' (**Fetch**) with **Option** X'0001' only. Name of a fullword containing the length of the data in the record to be stored or the amount of storage the caller has provided for retrieval. Valid values are 1 to 2,000,000,000.

When **Function_code** X'0004' (**Fetch**) and **Option** X'0001' are specified, **Data_length** will be set to the actual length of the data requested. If return and reason codes indicate that an insufficient length value was specified, resulting in a return of partial data, use the updated **Data_length** value to obtain a larger results area and resubmit the request.

Function code X'0006'(**Manage a read/write cache**) with Options X'0001' (**Store**), X'0003' (**Retrieve**), and X'0004' (**RetrieveAppl**) only: Name of a fullword containing the length of the application data record to be stored (option X'0001'), or the length of the application data record retrieved (options X'0003' and X'0004').

- When **Store** (option X'0001') is specified, a zero length indicates that the application data record has not been specified. Otherwise, valid values are 1 to 8192.
- When **Retrieve** (option X'0003') or **RetrieveAppl** (option X'0004') is specified, the caller can either specify zero, indicating that RACF should obtain storage for the output application data record, or specify the output area and the length of the area. If the length of the caller-specified output area is large enough to contain the application data record, RACF will use the area. Otherwise, RACF will obtain storage for the output area. In either case, **Data_length** will be changed to the actual length of the returned application data record. A length of zero indicates that no application data record was returned.

Function code X'0007' (**Manage an extended read/write cache**) with Option X'0002' (**RetrieveAppl**) only: Name of a fullword containing the length of the application data record retrieved.

- When **RetrieveAppl** (option X'0002') is specified, the caller can either specify zero, indicating that RACF should obtain storage for the output application data record, or specify the output area and the length of the area. If the length of the caller-specified output area is large enough to contain the application data record, RACF will use the area (ALET-qualified through ParmALET parameter). Otherwise, RACF will obtain storage in the primary address space for the output area. In either case, **Data_length** will be changed to the actual length of the returned application data record. A length of zero indicates that no application data record was returned.

Data_timeout
The name of a fullword containing the number of seconds before the application data record times out. If zero is specified, a default value of 3600 seconds (1 hour) is used. Otherwise, a value of 1 to 3600 seconds must be specified. When this interval has expired, the application data record will not

be found by a **Locate** (option X'0002') request. Expired application data records become eligible for **RemoveExpired** (option X'0006') processing, either called explicitly or performed internally by RACF, when all of the cache references that can be used to retrieve those records have also expired.

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

Source_ptr
Function code X'0006' (**Manage a read/write cache**) with Options X'0001' (**Store**), X'0002' (**Locate**), and X'0003' (**Retrieve**) only: Name of the address of the source record to be stored (options X'0001' and X'0002'), or name of a fullword where the address of the source record is to be placed during retrieval (option X'0003'). The length is specified in **Source_length**.

When **Store** (option X'0001') or **Locate** (option X'0002') is specified, the caller is responsible for obtaining and freeing storage for the source record. The length is specified in **Source_length**.

When **Retrieve** (option X'0003') is specified, R_cacheserv obtains storage for the source record in the subpool provided by the **Subpool** parameter. Storage will be obtained in the primary address space. The caller is responsible for freeing this storage. The length is returned in **Source_length**.

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

Source_length
Function code X'0006' (**Manage a read/write cache**) with Options X'0001' (**Store**), X'0002' (**Locate**), and X'0003' (**Retrieve**) only: Name of a fullword containing the length of the source record to be stored (options X'0001' and X'0002'), or the length of the record retrieved (option X'0003').

When **Store** (option X'0001') is specified, a zero length indicates that the source record has not been specified. Otherwise, valid values are 1 to 8192.

When **Retrieve** (option X'0003') is specified, **Source_length** will be set to the actual length of the record retrieved.

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

Reference_timeout
The name of a fullword containing the number of seconds before the reference times out. If zero is specified, a default value of 3600 seconds (1 hour) is used. Otherwise, a value of 1 to 3600 seconds must be specified. When this interval has expired, the reference can no longer be used on a **Retrieve** (option X'0003') or **RetrieveAppl** (option X'0004') request.

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

Reference_userID
The name of an 8-byte area containing a reference user ID.

A successful **Store** (option X'0001') or **Locate** (option X'0002') request will return a cache reference, consisting of both a **Reference_userID** value and a **Reference** value. The **Reference_userID** and **Reference** value pair can be used on a subsequent **Retrieve** (option X'0003'), **RetrieveAppl** (option X'0004'), or **Remove** (X'0005') request. Only a **Reference_userID** and **Reference** value pair from a successful **Store** or **Locate** request will produce expected results on a **Retrieve**, **RetrieveAppl**, or **Remove** request.

An output reference user ID will be prefixed by the characters "**" X('5C5C'). The remaining 6 characters are reserved for use by the security manager (RACF).

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

Reference
The name of an 8-byte area containing a reference value.

A successful **Store** (option X'0001') or **Locate** (option X'0002') request will return a cache reference, consisting of both a **Reference_userID** value and a **Reference** value. The **Reference_userID** and **Reference** value pair can be used on a subsequent **Retrieve** (option X'0003'), **RetrieveAppl** (option X'0004'), or **Remove** (X'0005') request. Only a **Reference_userID** and **Reference** value pair from a successful **Store** or **Locate** request will produce expected results on a **Retrieve, RetrieveAppl,** or **Remove** request.

An output reference value will be a printable EBCDIC string, consisting of the characters "0-9", "A-Z", "$" (X'5B'), "#" (X'7B), or "@" (X'7C').

This parameter applies only to function code X'0006' (**Manage a read/write cache**)

ACEE_ALET
The name of a fullword that must be in the primary address space and contains the ALET for the ACEE referred to by the ACEE parameter.

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

ACEE
For function code 6, the name of an area containing the ACEE to be used to build and store the application data record name and the application data record. For function code 7, the name of an area containing the address of the ACEE, which has an attached IDID, to be used to build and store the application data record name and the application data record. The ACEE is not used for authority checking. This parameter is ignored unless specified by a caller in supervisor state or system key.

The ACEE must begin with the eyecatcher "ACEE". Otherwise, the area must contain binary zeros in the first 4 bytes. When the area contains binary zeros, RACF uses the task-level ACEE if found, or the address space ACEE.

ICRX_area
Function code X'0007' (**Manage an extended read/write cache**) only. Name of the address (ALET-qualified by the ParmALET parameter) of the extended identity context reference to be used as the source of the context reference for retrieval (option X'0002'), or for the removal (option X'0003') of a record that is no longer needed, or the name of a fullword where the primary address space address of the extended identity context reference is to be placed during store (option X'0001'). To indicate that a record is no longer valid and should not be used (option X'0003'), the **ICRX_area** contains the IDID identifying the record, but does not contain a context reference.

For more information on extended identity context references and distributed identity data (IDID), refer to *z/OS Security Server RACF Data Areas*.

ICRX_length
Function code X'0007' (**Manage an extended read/write cache**) only. Name of a

fullword containing the length of the ICRX to be used as the source of the context reference to be retrieved or removed (options X'0002' and X'0003'), or the actual length of the ICRX returned by store (option X'0001').

Return and reason codes

IRRSCH00 returns the following values in the reason and return code parameters:

Table 38. Return and reason codes for R_cacheserv

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful. For function code X'0003', the cache was both completed and also hardened successfully to the RACF database.
0	0	4	Function code X'0003' completed and hardened the cache successfully, but encountered problems deleting excess CACHECLS profiles from the RACF database. A record is cut to LOGREC with more diagnostic information.
0	0	8	Function code X'0003' completed the cache successfully, but the attempt to harden the cache to the RACF database failed. A record is cut to LOGREC with more diagnostic information.
0	0	12	Function code X'0003' completed the cache successfully but no attempt was made to harden the cache to the RACF database. Either the CACHECLS class was not active or the cachename profile had not been defined in the class.
0	0	16	The service was successful, but Data_length is too small to fit all of the application data record, so a new output area was obtained. Data_length is set to the actual length of the new output area and the field pointed to by Data_ptr is set to the address of the new output area. The caller is responsible for releasing both the provided output area and the new output area that was returned.
4	0	0	RACF is not installed.
8	8	0	Invalid function code.
8	8	4	Parameter list error.
8	8	8	An internal error was encountered. A record may be cut to LOGREC with more diagnostic information.
8	8	12	A recovery environment could not be established.
8	8	16	Not authorized to use this service.
8	8	20	Record not found during fetch.

Table 38. Return and reason codes for R_cacheserv (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	24	Record found during fetch, but Data_length is too small to fit all of the data. Data_length is set to the length needed to satisfy this request. Partial data is copied to the field pointed to by Data_ptr. If the specified Option value is X'0002', then Version_length is too small to fit all of the data. Version_length is set to the length needed to satisfy this request. Partial data is copied to the field pointed to by Version.
8	8	28	No CACHECLS profiles were found in the RACF database during the restore phase of **Fetch**, or the CACHECLS class is not active. No cache exists.
8	8	32	Error encountered in reading CACHECLS profiles from the RACF database during the restore phase of **Fetch**. No cache exists. A record is cut to LOGREC with more diagnostic information.
8	8	36	Only the caller of **Start** may **Add** or **End**. You may not add data to the cache, or **End** this cache.
8	8	48	This task has called **Start**, but has not yet called **End**. **Fetch** or **Delete** is not allowed by this task until **End** has been called.
8	8	72	Not invoked in task mode.
8	8	76	The cache reference is not valid.
8	8	80	Application data record not located. The name of the application data record may not be valid or the application data record may have expired.
8	8	84	An internal error was encountered on a remote system. A record with more diagnostic information may be cut to LOGREC on the failing system. A record that identifies the failing system is cut to LOGREC on this system.
8	8	88	Unable to determine the name of the local EIM registry name. Either the LOCALREGISTRY field has not been defined in the IRR.ICTX.DEFAULTS.sysid profile or the IRR.ICTX.DEFAULTS profile in the LDAPBIND class, or the LDAPBIND class is not active and RACLISTed. See *z/OS Integrated Security Services EIM Guide and Reference* for how to configure the Identity Cache.
8	12	9	The number of parameters is not valid.

Table 38. Return and reason codes for R_cacheserv (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	12	11	An invalid option was specified. Valid options for function code X'0006' are X'0001' through X'0007'. Valid options for function code X'0007' are X'0001' through X'0005'.
8	12	16	The length of the application data record name is not valid. A zero length indicates that the application data record name has not been specified. Otherwise, valid values are 1 to 8192.
8	12	18	The length of the application data record is not valid. A zero length indicates that the application data record has not been specified. Otherwise, valid values are 1 to 8192.
8	12	19	The application data record timeout interval is not valid. A zero value indicates that the default timeout interval of 3600 seconds should be used. Otherwise, valid values are 1 to 3600 seconds.
8	12	21	The length of the source record is not valid. A zero length indicates that the source record has not been specified. Otherwise, valid values are 1 to 8192.
8	12	22	The reference timeout interval is not valid. A zero value indicates that the default timeout interval of 3600 seconds should be used. Otherwise, valid values are 1 to 3600 seconds.
8	12	23	The reference user ID is not valid. A reference user ID must be prefixed by the characters "**" (X'5C5C').
8	12	25	The output subpool is not valid.
8	12	27	The specified ACEE is not valid. A valid ACEE must begin with the eyecatcher "ACEE". For function code X'0007', option X'0001', either no valid ACEE was found, or the ACEE did not point to an IDID with a valid eyecatcher and minimum length.
8	12	28	The length of the ICRX is not valid. It is either 0 when the parameter is required, does not have a valid eyecatcher, or is not the minimum length expected. These reason and return code values could also indicate that the ICRX contains an IDID that does not have a valid eyecatcher or minimum length.
8	16	0	An application data record was specified, but the application data record name was not specified.

Table 38. Return and reason codes for R_cacheserv (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	100	Offset to the ICRX field in error.	This return/reason code is issued when R_cacheserv is invoked to validate an ICRX. It indicates non-valid data in the ICRX portion of ICRX. The offsets are calculated from the beginning of the ICRX.
8	104	Offset to the IDID field in error.	This return/reason code is issued when R_cacheserv is invoked to validate an ICRX. It indicates non-valid data in the IDID portion of ICRX. The offsets are calculated from the beginning of the IDID.
8	108	Offset to the IDID Section 1 field in error.	This return/reason code is issued when R_cacheserv is invoked to validate an ICRX. It indicates non-valid data in Section 1 of the IDID portion of ICRX. The offsets are calculated from the beginning of Section 1 of the IDID.
8	112	Offset to the User's Distinguished Name field in error.	This return/reason code is issued when R_cacheserv is invoked to validate an ICRX. It indicates non-valid data in the User's Distinguished Name Data Section in Section 1 of the IDID portion of ICRX. The offsets are calculated from the beginning of the User's Distinguished Name Data Section.
8	116	Offset to the Registry Name field in error.	This return/reason code is issued when R_cacheserv is invoked to validate an ICRX. It indicates non-valid data in the Registry Name Data Section in Section 1 of the IDID portion of ICRX. The offsets are calculated from the beginning of the Registry Name Data Section.

Parameter usage

Function	Start new cache	Add record to cache	End cache creation	Fetch cache record	Delete cache
Function Code	X'0001'	X'0002'	X'0003'	X'0004'	X'0005'
ParmALET	In	In	In	In	In
NumParms	In	In	In	In	In
Option	N/A	N/A	In	In	In
Version	In	N/A	N/A	Out	N/A
Version_length	In	N/A	N/A	In/Out	N/A
Cache_name	In	In	In	In	In
Record_name_ptr	N/A	In	N/A	In	N/A
Record_name_length	N/A	In	N/A	In	N/A
Data_ptr	N/A	In	N/A	In/Out	N/A
Data_Length	N/A	In	N/A	In/Out	N/A
Data_timeout	N/A	N/A	N/A	N/A	N/A

Function	Start new cache	Add record to cache	End cache creation	Fetch cache record	Delete cache
Source_ptr	N/A	N/A	N/A	N/A	N/A
ACEE	N/A	N/A	N/A	N/A	N/A
Source_length	N/A	N/A	N/A	N/A	N/A
Reference_timeout	N/A	N/A	N/A	N/A	N/A
Reference_userID	N/A	N/A	N/A	N/A	N/A
Reference	N/A	N/A	N/A	N/A	N/A
Subpool	N/A	N/A	N/A	N/A	N/A
ACEE_ALET	N/A	N/A	N/A	N/A	N/A

Function	Manage a read/write cache						
Function_code	X'0006'						
Option	X'0001' **Store**	X'0002' **Locate**	X'0003' **Retrieve**	X'0004' **RetrieveAppl**	X'0005' **Remove**	X'0006' **RemoveExpired**	X'0007' **Destroy**
ParmALET	In	In	In	In	In	In	In
NumParms	In	In	In	In	In	In	In
Version	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Version_length	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Cache_name	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Record_name_ptr	In*	In	Out	Out	N/A	N/A	N/A
Record_name_length	In	In	Out	Out	N/A	N/A	N/A
Data_ptr	In*	N/A	In/Out	In/Out	N/A	N/A	N/A
Data_length	In	N/A	In/Out	In/Out	N/A	N/A	N/A
Data_timeout	In*	N/A	N/A	N/A	N/A	N/A	N/A
Source_ptr	In*	In	Out	N/A	N/A	N/A	N/A
Source_length	In	In	Out	N/A	N/A	N/A	N/A

* This parameter is optional, see Usage Notes.

Function	Manage an extended read/write cache				
Function_code	X'0007'				
Option	X'0001' **Store**	X'0002' **RetrieveAppl**	X'0003' **Remove**	X'0004' **Store** a reusable ICRX	X'0005' **Validate** a user-built ICRX
ParmALET	In	In	In	In	In
NumParms	In	In	In	In	In
Version	N/A	N/A	N/A	N/A	N/A
Version_length	N/A	N/A	N/A	N/A	N/A
Cache_name	N/A	N/A	N/A	N/A	N/A
Record_name_ptr	N/A	N/A	N/A	N/A	N/A
Record_name_length	N/A	N/A	N/A	N/A	N/A
Data_ptr	N/A	Out	N/A	N/A	N/A
Data_length	N/A	Out	N/A	N/A	N/A
Data_timeout	N/A	N/A	N/A	N/A	N/A
Source_ptr	N/A	N/A	N/A	N/A	N/A
Source_length	N/A	N/A	N/A	N/A	N/A
Reference_timeout	N/A	N/A	N/A	N/A	N/A

Function	Manage an extended read/write cache				
Reference_userID	N/A	N/A	N/A	N/A	N/A
Reference	N/A	N/A	N/A	N/A	N/A
Subpool	In	In	N/A	In	N/A
ACEE_ALET	In*	N/A	N/A	In*	N/A
ACEE	In*	N/A	N/A	In*	N/A
ICRX_area	Out	In	In	Out	In
ICRX_length	Out	In	In	Out	In

* This parameter is optional, see parameter description.

Usage notes

1. An ALET must be specified for the SAF_return_code, RACF_return_code, and RACF_reason_code parameters, and a single ALET specified for all of the remaining parameters, not including the ACEE_ALET and ACEE parameters. The ALET for the ACEE parameter must be specified separately, using the ACEE_ALET parameter.
2. The parameter list for this callable service is intended to be variable length to allow for future expansion. Therefore, a parameter containing a count of parameters is used: NumParms. This parameter tells how many parameters appear in the list following and **including** the NumParms parameter. NumParms must be set to 19 for function code X'0006' or 21 for function code X'0007'. However, for compatibility with prior releases, invokers who only use function codes X'0001' through X'0005' can continue to specify a NumParms value of 10.
3. Use of the **Add** function code first requires an invocation of R_cacheserv with the **Start** function code. After all records have been added, R_cacheserv must be invoked one additional time with the **End** function code to indicate that the cache has been filled and should be made available for use. Only the issuer of **Start** (same task) can **Add** and **End**.
4. To allow the R_cacheserv callable service to harden/restore the cache to/from the RACF database as profiles in the CACHECLS class, two steps must be taken:
   a. the class must be made active by the RACF SETROPTS CLASSACT command, that is, SETROPTS CLASSACT(CACHECLS)
   b. a base profile for this cache must be defined in the CACHECLS class using the RACF RDEFINE command, that is, RDEFINE CACHECLS cachename, where cachename is the Cache_name given as input to the R_cacheserv callable service.

   Unless both of these steps are taken, the harden and restore phases of the **End** and **Fetch** functions, respectively, will not be performed for the cache identified by Cache_name.
5. When the cache is hardened to the RACF database, the cache contents are written to the database as profiles containing 50K pieces of the cache with the last profile's size being less than or equal to 50K. The names of the profiles are constructed from the input Cache_name parameter by adding the values **_ddd**, where **ddd** is the sequential dataspace number (in decimal), starting with **001** and **_nnnnn**, where **nnnnn** is the number of the profile containing cache information for that dataspace, also in decimal. The first 50K of the cache is written as cachename_001_00001, the second as cachename_001_00002, and so on. The profiles will be created with the same owner as that of the base profile.

6. If a request is made to **Start** a cache, followed by any number of **Add** requests, then **Start** is requested again for the same cache name without an intervening **End** request, this will result in the **Start** of a new empty cache, causing all records that were previously added to be discarded.
7. If a **Start**, **Add**, or **End** (Option X'0001') results in a SAF return code of 8, the state of the cache is undefined and it is highly recommended that R_cacheserv be invoked again, specifying **End** with **Option** X'0002' to discard the new cache, leaving the existing cache intact. Note that if the SAF return code 8 was caused by an ABEND during **Start** or **Add**, **End** with Option X'0002' will result in SAF return code 8 with RACF return code 8 and RACF reason code 36, indicating that the new cache was already discarded during ABEND recovery processing.
8. If more than one record is added to the cache with the same name (specified using the **Record_name_ptr** parameter), **Fetch** results are unpredictable.
9. The dataspaces that form the cache are associated with the master address space and are persistent so that records in the cache can be fetched from any address space. Function code X'0005' can be used to delete the cache when its contents no longer need to be accessed.
10. Function codes X'0001' through X'0005', function code X'0006', and function code X'0007' are not compatible. In other words, function code X'0006' option X'0003' cannot be used to retrieve a record that was added to a cache using function code X'0002' and function code X'0004' cannot be used to fetch a record that was stored using function code X'0006' option X'0001'.
11. For function code X'0006', **Manage a read/write cache**, and function code X'0007', **Manage an extended read/write cache**, when a parameter list error is detected, R_cacheserv returns SAF return code 8, RACF return code 12, and RACF reason code nn, where nn indicates the position in the parameter list of the parameter in error. For example, the **NumParms** parameter is the 9th parameter in the parameter list, so if an invalid value is supplied, R_cacheserv will return SAF return code 8, RACF return code 12, and RACF reason code 9.
12. If a supervisor state or system key caller specifies a valid ACEE on a **Store** request (option X'0001'), RACF will use the specified ACEE to build and store an application data record name and an application data record. The **Record_name_ptr, Record_name_length, Data_ptr,** and **Data_length** parameters will be ignored. See the description of the ACEE parameter for more information.

    If no ACEE (see the parameter description for more information), source record, application data record name, and application data record are specified (**Source_length, Record_name_length,** and **Data_length** are all zeros) on a **Store** request, RACF will build and store the application data record name and the application data record using the task-level ACEE if found, or the address space ACEE..
13. R_cacheserv does not use the parmALET value when it obtains storage for the application data record name, the application data record, the ICRX, or the source record. Storage will always be obtained in the primary address space.
14. When RACF is enabled for sysplex communication and it determines that an R_cacheserv retrieve or remove request (function code X'0006', options X'0003', X'0004', or X'0005' and function code X'0007', options X'0002' and X'0003') is for data that is cached on another member of the sysplex, RACF will attempt to retrieve or remove the data from the other member. See *z/OS Security Server RACF System Programmer's Guide* for more information about how to enable RACF for sysplex communication.

15. RACF recommends that customers put the Integrated Cryptographic Service Facility (ICSF) CSNBRNG module in the link pack area (LPA) or the modified link pack area (MLPA) so that it can be used for generating reference values (**Reference** parameter). If RACF cannot find CSNBRNG in LPA or MLPA, it will default to using a less efficient software pseudo random number generator (PRNG) for generating reference values.
16. The RACF read/write cache, function code X'0006', has a capacity limit of 2 GB. Assuming that application data records are less than 500 bytes, the cache can contain a maximum of approximately 4 million records at any point in time. When data records are in the 501-1000 byte range, the maximum number of cached records will be approximately 2 million. However, RACF does cache cleanup regularly by doing an internal **RemoveExpired**, so it is unlikely that cache limits will be reached.
17. Function code X'0007', option X'0001' stores an ENVR object in the extended read/write cache. This ENVR object does not include any installation data pointed to by ACEEIEP.

Related services

None

R_chaudit (IRRSCA00): Change audit options

Function

The **R_chaudit** service verifies that the user has authority to change the audit options for the specified file and, if so, sets the audit bits from the input audit options parameter.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address

space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

If the SECLABEL class is active and the file or directory has a security label, the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the current security label of the process, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service has passed a system CRED. Security label checking will also be bypassed for the users with the AUDITOR attribute that are setting the audit options.

Format

```
CALL IRRSCA00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Audit_options,
               ALET, FSP,
               ALET, File_identifier,
               ALET, CRED
              )
```

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 words 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.

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.

Audit_options
The name of a word containing the audit options to be set. For RACF, the following options are defined:

- Audit options can be specified for each type of access:

  Byte 1 read access audit options

  Byte 2 write access audit options

  Byte 3 execute/search access audit options

  Byte 4 audit flag
- The following flags are defined for each of the first three bytes of audit options:

X'00' do not audit any access attempts

X'01' audit successful access attempts

X'02' audit failed access attempts

X'03' audit both types of attempts

- In the last byte (the audit flag), the last bit indicates whether user audit options or auditor audit options should be set:

  X'00' set user audit options

  X'01' set auditor audit options

Reserved bits in the audit options parameter must be zero.

FSP
The name of the IFSP for the file whose audit options are to be changed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSCA00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized to change the file's user audit options.
8	8	8	The user is not authorized to change the file's auditor audit options.
8	8	12	An internal error occurred during RACF processing.
8	8	24	Reserved bits in an input parameter were not zero.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. Two sets of audit bits exist for a file, one for auditor-specified options and one for user-specified options. The audit flag in the parameter list indicates which type of options should be set.

   If the audit flag indicates auditor options, the user must have auditor authority. Auditors can set the auditor options for any file, even those they do not have path access to or authority to use for any other reason.

   If the audit flag indicates user options, the user must be a superuser or must be the owner of the file (that is, the effective UID of the calling process is equal to the owner UID of the file.)

3. If the change is being made for an open file, that pathname in the CRED is not used.
4. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

None

R_chmod (IRRSCF00): Change file mode

Function

The **R_chmod** service checks whether the calling process is authorized to change the mode of the specified file (identified by the input IFSP) and, if so, changes the permission bits and the S_ISUID, S_ISGID, and S_ISVTX bits in the IFSP to the values specified by the mode parameter.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. To change the mode, the user must be a superuser or must be the owner of the file. If the user can change the mode and the user is not a superuser, the S_ISGID bit is cleared, except when the owner z/OS UNIX group identifier (GID) of the file is equal to the effective GID or to one of the supplementary groups of the calling process.
2. Only a superuser or directory/file owner can change the S_ISVTX bit.
3. If the caller is not superuser, or the file owner, an authorization check is performed for READ access to the resource named

SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class. If the authorization check is successful, the caller is treated as a superuser.

4. If the SECLABEL class is active and the file or directory has a security label, then the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the current security label of the process, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service has passed a system CRED.

Format

```
CALL IRRSCF00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Mode,
               ALET, FSP,
               ALET, File_Identifier,
               ALET, CRED
              )
```

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 words containing the ALETs must be in the primary address space.

SAF_return_code
The name of a full word 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.

Mode
The name of a word containing the mode value (the file type, the permission bits, and the S_ISUID, S_ISGID, and S_ISVTX bits) to be set in the IFSP for the file.

See “File type and file mode values” on page 4 for a definition of the security bits in the mode parameter. Reserved bits in the mode parameter must be zero.

FSP
The name of the IFSP for the file whose mode bits are to be changed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSCF00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized to change the mode of the file.
8	8	12	An internal error occurred during RACF processing.
8	8	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. The mode word is mapped by the z/OS UNIX macro BPXYMODE.
3. If the audit function code indicates an open file, the path name in the CRED is not used.
4. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

makeFSP, R_umask, R_chown, R_setfacl, ck_access

R_chown (IRRSCO00): Change owner and group

Function

The **R_chown** service checks to see whether the user is authorized to change the owner of the file, and, if so, changes the owner z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) to the specified values.

If the user is authorized to change the file, the S_ISUID and S_ISGID bits are cleared in the IFSP.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. This service implements the _POSIX_CHOWN_RESTRICTED feature in POSIX 1003.1.

   If the discrete profile named CHOWN.UNRESTRICTED does not exist in the UNIXPRIV class, or the caller has no access to it, then:

   - A user can change the owner z/OS UNIX user identifier (UID) value only if the user is a superuser.
   - A user can change the owner z/OS UNIX group identifier (GID) of a file if:
     - The user is a superuser,
     - Or, all of the following are true:
       - The effective UID of the calling process is equal to the owner UID of the file (that is, the user is the owner of the file).
       - The input UID is equal to the owner UID of the file or -1
       - The input z/OS UNIX group identifier (GID) is equal to the effective GID or to one of the supplemental groups of the calling process.

   If the discrete profile named CHOWN.UNRESTRICTED exists in the UNIXPRIV class, then:

   - A user can change the owner z/OS UNIX user identifier (UID) if:
     - The user is a superuser
     - The effective UID of the calling process is equal to the owner UID of the file (that is, the user is the owner of the file)
     - The caller has UPDATE access to CHOWN.UNRESTRICTED if the UID is being changed to 0, or
     - The caller has READ access to CHOWN.UNRESTRICTED if the UID is being changed to a value other than 0
   - A user can change the owner z/OS UNIX group identifier (GID) if:
     - The user is a superuser
     - Or, all of the following are true:
       - The effective UID of the calling process is equal to the owner UID of the file (that is, the user is the owner of the file).
       - The input UID is equal to the owner UID of the file or -1
       - The input z/OS UNIX group identifier (GID) is equal to the effective GID or to one of the supplemental groups of the calling process.
     - Or, the caller has READ access to CHOWN.UNRESTRICTED and the input GID is not equal to the effective GID or to one of the supplemental groups of the calling process.

2. If the caller is not superuser, an authorization check is performed on the resource name in the UNIXPRIV class indicated in Table 39. If the authorization check is successful, the caller is treated as a superuser.

Table 39. UNIXPRIV class resource names used in R_chown

Audit function code	Resource name	Access required
N/A	SUPERUSER.FILESYS.CHOWN	READ

3. If the SECLABEL class is active and the file or directory has a security label, then the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the process's current security label, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service has passed a system CRED.

Format

```
CALL IRRSCO00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, UID,
               ALET, GID,
               ALET, FSP,
               ALET, File_identifier,
               ALET, CRED
              )
```

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 words 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.

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.

UID
The name of a word containing the z/OS UNIX user identifier (UID) to be set as the file owner UID or -1 to indicate that:

1. This field is not changed in the IFSP.
2. The z/OS UNIX group identifier (GID) can be changed.

GID
The name of a word containing the z/OS UNIX group identifier (GID) to be set as the file owner GID or -1 to indicate that this field is not changed in the IFSP.

FSP
The name of the IFSP for the file whose owner z/OS UNIX user identifier (UID) and z/OS UNIX group identifier (GID) are to be changed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSCO00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The z/OS UNIX user identifier (UID) is not valid.
8	8	8	The z/OS UNIX group identifier (GID) is not valid.
8	8	12	An internal error occurred during RACF processing.
8	8	20	The user is not authorized to change the owner UID or GID.
8	8	32	The CRED user type is not supported.
8	8	36	The user is not authorized to set the specified GID.

Usage notes

1. This service is intended only for use by a z/OS UNIX file system and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.
2. If the input UID or GID (or both) is equal to -1, that field is not changed in the IFSP.
3. If the audit function code indicates an open file, the pathname in the CRED is not used.
4. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

query_file_security_options, R_chmod, R_setfacl

R_datalib (IRRSDL00 or IRRSDL64): OCSF data library

Function

The **R_datalib** service provides the function required to implement the OCSF (Open Cryptographic Services Facility) Data library functions using RACF key rings and z/OS PKCS #11 tokens.

Requirements

Authorization:
PSW key 8, non-APF authorized, problem state

Dispatchable unit mode:
Task

Cross memory mode:
PASN = HASN

AMODE:
31 or 64

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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. The mapping macro IRRPCOMP is for 31 bit callers and IRRPCOMX is for 64 bit callers. To allow for this, the last word in the parameter list , for 31 bit callers, must have a 1 in the high-order (sign) bit. Num_Parms takes care of 64 bit callers.

RACF authorization

There are two ways of authority checking for the R_datalib callable service: global profile checking in the FACILITY class and ring-specific profile checking in the RDATALIB class. Global profile checking applies to all the key rings. Ring-specific profile checking applies to a specific key ring. To use the ring-specific profile checking, the RDATALIB class must be RACLISTed.

With ring-specific profile checking, a resource with the format <ringOwner>.<ringName>.LST is used to provide access control to a specific key ring on R_datalib READ functions, that are, DataGetFirst, DataGetNext, and GetUpdateCode. A resource with the format <ringOwner>.<ringName>.UPD is

used to provide access control to a specific key ring on the UPDATE functions, that are, NewRing, DataPut, DataRemove, and DelRing.

Global profile checking using the IRR.DIGTCERT.<function> resource is also applicable in the following circumstances:

- For the CheckStatus and IncSerialNum functions, only global profile checking is used.
- For the other functions that first use ring-specific profile checking, global profile checking is used when there is no matching profile to the <ringOwner>.<ringName>.<function> resource.

With ring-specific profile checking, the ringOwner must be in uppercase. The ringName is folded into uppercase during profile checking. The ringNames that differ only in cases use the same profile.

If the data entered in the ringOwner and ringName fields has reached the field size limits, and you want to create a discrete profile, you can truncate the ring name from the end to make the whole profile name length 246 characters.

For example, if the owner ID is JOESMITH and the ring name is: THISISARINGWITH237CHARACTERS...RINGEND (with a length of 237), the discrete profile will be JOESMITH.THISISARINGWITH237CHARACTERS...RIN.UPD.

If the owner ID is JOES, the entire ring name can be used.

The following lists describe a detailed breakdown of authority checking.

- Authority required for the DataGetFirst, DataGetNext, and GetUpdateCode functions:

  Note: Supervisor or system key callers can bypass the authorization checks for the DataGetFirst, DataGetNext, and GetUpdateCode functions by setting the CDDL(X)_ATT_SKIPAUTH flag in the Attributes parameter.

  The resource <ringOwner>.<ringName>.LST in the RDATALIB class is checked first. If there is no match for <ringOwner>.<ringName>.LST, the IRR.DIGTCERT.LISTRING resource is used.

Table 40. Ring-specific profile checking for the DataGetFirst, DataGetNext, and GetUpdateCode functions

Function	Authority required
List certificates and get the sequence number for a real key ring	READ authority to <ringOwner>.<ringName>.LST
List certificates and get the sequence number for a virtual key ring	READ authority to <virtual ring owner>.IRR_VIRTUAL_KEYRING.LST **Note:** The virtual ring owner can be an ordinary user ID, a CERTAUTH user ID (CERTIFAUTH), or a SITE user ID (SITECERTIF).

Table 41. Global profile checking for the DataGetFirst, DataGetNext, and GetUpdateCode functions

Function	Authority required
List certificates and get the sequence number for one's own key ring, a CERTAUTH, or a SITE's virtual key ring	READ authority to IRR.DIGTCERT.LISTRING
List certificates and get the sequence number for other's ring	UPDATE authority to IRR.DIGTCERT.LISTRING

For information about the additional authority needed for the private key retrieval, see "Usage notes" on page 181.

- Authority required for the CheckStatus function

  Note: Supervisor or system key callers can bypass the authorization checks for the CheckStatus function by setting the CDDL(X)_ATT_SKIPAUTH flag in the Attributes parameter.

  The CheckStatus function requires READ authority to the resource IRR.DIGTCERT.LIST in the FACILITY class.

Table 42. Profile checking for the CheckStatus function

Function	Authority required
Return the TRUST or NOTRUST status for a specified certificate	READ authority to IRR.DIGTCERT.LIST

- Authority required for the DataAbortQuery function

  The DataAbortQuery function requires no authority.

- Authority required for the IncSerialNum function

  If the caller is RACF special, no authority checking is done; otherwise appropriate authority to the resource IRR.DIGTCERT.GENCERT in the FACILITY class is required: READ authority if the certificate is owned by the caller, or CONTROL authority if the certificate is a SITE or CERTAUTH certificate.

Table 43. Profile checking for the IncSerialNum function

Function	Authority required
Increment and return the last serial number field (CERTLSER) associated with one's own input certificate	READ authority to IRR.DIGTCERT.GENCERT
Increment and return the last serial number field (CERTLSER) associated with a SITE or CERTAUTH certificate	CONTROL authority to IRR.DIGTCERT.GENCERT

- Authority required for the NewRing function

  If the caller is RACF special, no authority checking is done; otherwise the resource <ringOwner>.<ringName>.UPD is checked first. If there is no match for <ringOwner>.<ringName>.UPD, the IRR.DIGTCERT.ADDRING and IRR.DIGTCERT.REMOVE resources are used.

Table 44. Ring-specific profile checking for the NewRing function

Function	Authority required
Create a new ring for <ringOwner> named <ringName>	READ authority to <ringOwner>.<ringName>.UPD

Table 44. Ring-specific profile checking for the NewRing function (continued)

Function	Authority required
Remove all certificates from an existing ring	READ authority to <ringOwner>.<ringName>.UPD

Table 45. Global profile checking for the NewRing function

Function	Authority required
Create a new ring for oneself	READ authority to IRR.DIGTCERT.ADDRING
Create a new ring for someone else	UPDATE authority to IRR.DIGTCERT.ADDRING
Remove all certificates from one's own ring	READ authority to IRR.DIGTCERT.REMOVE
Remove all certificates from someone else's ring	UPDATE authority to IRR.DIGTCERT.REMOVE

- Authority required for the DelRing function

  If the caller is RACF special, no authority checking is done; otherwise the resource <ringOwner>.<ringName>.UPD is checked first. If there is no match for <ringOwner>.<ringName>.UPD, the IRR.DIGTCERT.DELRING resource is used.

Table 46. Ring-specific profile checking for the DelRing function

Function	Authority required
Delete a ring owned by <ringOwner> named <ringName>	READ authority to <ringOwner>.<ringName>.UPD

Table 47. Global profile checking for the DelRing function

Function	Authority required
Delete one's own ring	READ authority to IRR.DIGTCERT.DELRING
Delete someone else's ring	UPDATE authority to IRR.DIGTCERT.DELRING

- Authority required for the DataRemove function

  If the caller is RACF special, no authority checking is done; otherwise the resource <ringOwner>.<ringName>.UPD is checked first. If there is no match for <ringOwner>.<ringName>.UPD, the IRR.DIGTCERT.REMOVE resource is used.

Table 48. Ring-specific profile checking for the DataRemove function

Function	Authority required
Remove one's own certificate	READ authority to <ringOwner>.<ringName>.UPD
Remove someone else's certificate	UPDATE authority to <ringOwner>.<ringName>.UPD
Remove a SITE or CERTAUTH certificate	CONTROL authority to <ringOwner>.<ringName>.UPD

Table 49. Global profile checking for the DataRemove function

Function	Authority required
Remove one's own certificate from one's own ring	READ authority to IRR.DIGTCERT.REMOVE
Remove someone else's certificate from one's own ring	
Remove one's own certificate from other's ring	CONTROL authority to IRR.DIGTCERT.REMOVE
Remove someone else's certificate from other's ring	
Removes a SITE or CERTAUTH certificate from other's ring	
Removes a SITE or CERTAUTH certificate from one's own ring	UPDATE authority to IRR.DIGTCERT.REMOVE

If CDDL(X)_ATT_DEL_CERT_TOO is also specified, IRR.DIGTCERT.DELETE is checked in addition to the checking on the <ringOwner>.<ringName>.UPD resource or the IRR.DIGTCERT.REMOVE resource.

Note: There are two types of mapping, 31-bit mapping and 64-bit mapping. For every CDDL_xx entry, which comes from the 31-bit mapping, there is a corresponding CDDLX_xx entry from the 64-bit mapping. In this document, CDDL(X) is used to indicate both of the mappings.

Table 50. Profile checking for the DataRemove function if CDDL(X)_ATT_DEL_CERT_TOO is specified

Function	Authority required (if the certificate also needs to be deleted)
Remove one's own certificate	READ authority to IRR.DIGTCERT.DELETE
Remove someone else's certificate	UPDATE authority to IRR.DIGTCERT.DELETE
Remove a SITE or CERTAUTH certificate	CONTROL authority to IRR.DIGTCERT.DELETE

- Authority required for the DataPut function

  If the caller is RACF special, no authority checking is done; otherwise the resource <ringOwner>.<ringName>.UPD is checked first. If there is no match for <ringOwner>.<ringName>.UPD, the IRR.DIGTCERT.CONNECT, and possibly IRR.DIGTCERT.ADD, or IRR.DIGTCERT.ALTER resources are used, because "Add" and "Alter" might be involved in the operation.

  With global profile checking, authorization to connect is always required. However, additional authorization to add or alter might be required depending on the following factors:

  – Whether the certificate already exists in RACF
  – Whether the certificate status needs to be changed to TRUST or HIGHTRUST through the CDDL(X)_ATT_TRUST or CDDL(X)_ATT_HIGHTRUST attribute

  Appropriate access to IRR.DIGTCERT.ALTER is also required when the following two conditions apply.

  – The DIGTCERT resources are used.
  – The DataPut function changes the existing certificate's status from NOTRUST to TRUST through the CDDL(X)_ATT_TRUST attribute.

The following tables show the breakdown of profile checking for the DataPut function with the two different methods: ring-specific profile checking and global profile checking.

Table 51. Ring-specific profile checking for the DataPut function - Authority required to connect with the Personal usage

Function	Authority required
Connect one's own certificate to the ring	READ authority to <ringOwner>.<ringName>.UPD
Connect someone else's certificate to the ring	• CONTROL authority to <ringOwner>.<ringName>.UPD (If the private key is not specified) • UPDATE authority to <ringOwner>.<ringName>.UPD (If the private key is specified)
Connect a SITE or CERTAUTH certificate to the ring	

Table 52. Ring-specific profile checking for the DataPut function - Authority required to connect with the SITE or CERTAUTH usage

Function	Authority required
Connect one's own certificate to the ring	UPDATE authority to <ringOwner>.<ringName>.UPD
Connect someone else's certificate to the ring	
Connect a SITE or CERTAUTH certificate to the ring	

Table 53. Global profile checking for the DataPut function - Authority required to connect with the Personal usage

Function	Authority required
Connect one's own certificate to one's own ring	READ authority to IRR.DIGTCERT.CONNECT
Connect someone else's certificate to one's own ring	UPDATE authority to IRR.DIGTCERT.CONNECT
Connect one's own certificate to someone else's ring	CONTROL authority to IRR.DIGTCERT.CONNECT
Connect someone else's certificate to someone else's ring	
Connect a SITE or CERTAUTH certificate to one's own ring	
Connect a SITE or CERTAUTH certificate to someone else's ring	

Note: For information about the additional authority required to add or alter, see Table 55 on page 164.

Table 54. Global profile checking for the DataPut function - Authority required to connect with the SITE or CERTAUTH usage

Function	Authority required
Connect one's own certificate to one's own ring	CONTROL authority to IRR.DIGTCERT.ADD and READ authority to IRR.DIGTCERT.CONNECT

Table 54. Global profile checking for the DataPut function - Authority required to connect with the SITE or CERTAUTH usage (continued)

Function	Authority required
Connect someone else's certificate to one's own ring	CONTROL authority to IRR.DIGTCERT.ADD and UPDATE authority to IRR.DIGTCERT.CONNECT
Connect one's own certificate to someone else's ring	CONTROL authority to IRR.DIGTCERT.ADD and CONTROL authority to IRR.DIGTCERT.CONNECT
Connect someone else's certificate to someone else's ring	
Connect a SITE or CERTAUTH certificate to one's own ring	UPDATE authority to IRR.DIGTCERT.CONNECT
Connect a SITE or CERTAUTH certificate to someone else's ring	CONTROL authority to IRR.DIGTCERT.CONNECT

Table 55. Global profile checking for the DataPut function - Authority required to add or alter a certificate

Function	Authority required
Add a certificate for oneself	READ authority to IRR.DIGTCERT.ADD
Add a certificate for someone else	UPDATE authority to IRR.DIGTCERT.ADD
Add a SITE or CERTAUTH certificate	CONTROL authority to IRR.DIGTCERT.ADD
Alter one's own certificate	READ authority to IRR.DIGTCERT.ALTER
Alter someone else's certificate	UPDATE authority to IRR.DIGTCERT.ALTER
Alter a SITE or CERTAUTH certificate	CONTROL authority to IRR.DIGTCERT.ALTER

- Authority required for the DataRefresh function

  If the caller is RACF special, no authority checking is done; otherwise if the DIGTCERT class is RACLISTed, the caller must have class authority for the DIGTCERT class.

ICSF considerations

R_datalib processing makes use of ICSF services. If your installation has established access control over ICSF services, then the callers of R_datalib will need to be granted READ authority to ICSF services according to the following table:

Table 56. ICSF services used by R_datalib

R_datalib function	Specific parameters	ICSF Service (CSFSERV class resource)
DataGetFirst and DataGetNext	When Ring_name is a z/OS PKCS #11 token	CSF1TRL and CSF1GAV
GetUpdateCode	When Ring_name is a z/OS PKCS #11 token	CSF1TRL
DataPut	When a label of an existing PKDS private key is specified	CSFPKRR

If your installation has also established access control over keys stored in ICSF, the issuer of the DataPut function must have READ access authority to ICSF key by

label set up by the profile in the CSFKEYS class too.

Format

31 bit invocation:

```
CALL IRRSDL00 Work_area,
                ALET, SAF_return_code,
                ALET, RACF_return_code,
                ALET, RACF_reason_code,
                Function_code,
                Attributes,
                RACF_user_ID,
                Ring_name,
                Parm_list_version,
                Parmlist
              )
```

or 64 bit invocation:

```
CALL IRRSDL64 ((Num_parms,
                 Work_area,
                ALET, SAF_return_code,
                ALET, RACF_return_code,
                ALET, RACF_reason_code,
                Function_code,
                Attributes,
                RACF_user_ID,
                Ring_name,
                Parm_list_version,
                Parmlist
              )
```

Parameters

In this section, there is a function-specific parameter list for each function code, in addition to the general parameter list for R_datalib.

General parameter list for R_datalib

Num_parms
The name of a fullword containing the number of parameters in the parameter list, including the Num_parms parameter. The number must be 14. This parameter must be supplied when IRRSDL64 is invoked in 64-bit mode. It must not be supplied when IRRSDL00 is invoked in 31-bit mode.

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 and must be on a double word boundary.

ALET
The name of a word containing the ALET for the following parameter. Each ALET can be different. The words 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.

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.

Function_code
The name of a 1-byte input area containing the function code

X'01' **DataGetFirst:** Locate and return the first trusted certificate in the ring specified in Ring_name, based on the selection criteria.

On a DataGetFirst function, the user may specify some selection criteria by setting Number_predicates to 1, and then supplying some attribute information, such as attribute type, and the length and address of the attribute data. The data in the returned certificate will match the attribute data supplied.

X'02' **DataGetNext:** Locate and return the next trusted certificate in the ring, based on the criteria specified in DataGetFirst.

X'03' **DataAbortQuery:** Free resources from previous DataGetFirst and DataGetNext requests.

X'04' **CheckStatus:** Return the TRUST/ NOTRUST status for a specified certificate.

X'05' **GetUpdateCode:** Return the sequence number for the ring specified. A change in the ring sequence number (from a previously obtained ring sequence number) indicates that the ring has changed. A ring is considered changed when the list of certificates in the ring has changed, or the digital certificate information for a certificate in the ring has changed.

X'06' **IncSerialNum:** Increment and return the last serial number field (CERTLSER) associated with the input certificate.

X'07' **NewRing:** Create a new key ring or remove all the certificates from an existing key ring.

The RACF_user_ID and Ring_name are used to identify the ring. If the RACF_user_ID is not specified, it is equal to the user ID of the caller by default.

The syntax of Ring_name follows that of the RACDCERT ADDRING resource, that are, the restrictions imposed by the TSO parse, RACF's conversion rules, and the ADDRING validation exit.

X'08' **DataPut:** Add a certificate to the RACF database (if it does not already exist), and connect it to a key ring.

RACF_user_ID and Ring_name are used to identify the ring. If the RACF_user_ID is not specified, it is equal to the user ID of the caller by default.

If the input certificate does not exist in the RACF database, it will be added to RACF with a specified or system-generated label under the specified user ID. The certificate will be added with either of the following status:

- A specified status through the CDDL(X)_ATT_TRUST or CDDL(X)_ATT_HIGHTRUST attribute, or
- A status determined by RACF.

For more information, see the Attributes section.

If the private key associated with the certificate is specified in a DER-encoded format or as a key label, the certificate will be added with the following key types accordingly in the RACF database:

- Software RSA key

- Software DSA key
- Software Elliptic Curve Cryptography (ECC) National Institute of Standards and Technology (NIST) key
- Software Elliptic Curve Cryptography (ECC) Brainpool key
- ICSF RSA Modulus-Exponent key token
- ICSF RSA Chinese Remainder Theorem key token
- Elliptic Curve Cryptography (ECC) National Institute of Standards and Technology (NIST) key token
- Elliptic Curve Cryptography (ECC) Brainpool key token

The DataPut function checks the validity of the private key in both the software and hardware cases. If the private key is a software key, it should be in a DER-encoded format. If the private key is a PKDS or TKDS hardware key, it must be the label of an existing PKDS or TKDS private key entry.

Note: Retained hardware keys and Clear hardware keys are not supported.

If the certificate specified to be added is not already in the RACF database, after it is added, it will be connected to the key ring with the specified USAGE and DEFAULT value.

If the certificate specified to be added is already connected to the key ring, it will be reconnected with the specified USAGE and DEFAULT value.

If the certificate specified to be added is already in the RACF database with no associated private key, it might be re-added with a specified private key under the original ID, label, and status if the TRUST or HIGHTRUST attribute is not specified.

If the DIGTCERT class is RACLISTed, a successful DataPut operation indicates that a DataRefresh call is needed.

X'09' **DataRemove:** Remove a certificate from the key ring, and optionally delete it from the RACF database if the certificate is not connected to any other rings.

RACF_user_ID and Ring_name are used to identify the ring. If the RACF_user_ID is not specified, it is equal to the user ID of the caller by default.

X'0A' **DelRing:** Delete a key ring.

X'0B' **DataRefresh:** Refresh the in-storage certificates in the RACF database if the DIGTCERT class is RACLISTed. If the DIGTCERT class is not RACLISTed, no action is performed. DataRefresh might be required after calling DataPut or DataRemove.

The DIGTCERT class can be RACLISTed for digital certificates (processing the in-storage version instead of the database version) after calling the DataPut or DataRemove function. Therefore you need to refresh the in-storage version to reflect any changes to the certificate profiles. This can be done either through a RACF SETROPTS command `SETROPTS RACLIST(DIGTCERT) REFRESH` by a RACF administrator, or through the DataRefresh function by the caller.

To call the DataRefresh function, the caller needs to have class authority for the DIGTCERT class, and the RACF subsystem needs to

be running. Having class authority for the DIGTCERT class enables the refresh of that class only; other updates such as deleting, updating, and adding certificate profiles are still prohibited.

If the function code is not one of the preceding values, a parameter list error is returned.

Attributes

The name of a 4-byte area containing bit settings that direct the function to be performed. The bit settings are mapped as follows:

- The DataGetFirst (X'01') and DataGetNext (X'02') functions

  X'80000000' - CDDL(X)_ATT_ALL_KEYTYPES flag. This flag directs R_datalib to differentiate between PCICC key types, ICSF key types, DSA key types, Diffie-Hellman(DH) key types, Elliptic Curve Cryptography(ECC) key types and PKCS #1 key types, when returning the Function Specific Parameter List field *Private_key_type*. When this flag is off, R_datalib will handle the PCICC key type as an ICSF key type and return the value X'00000002'. It will also handle the DSA key type, DH key type and ECC key type as a PKCS #1 key type and return the value X'00000001'.

  X'20000000' - CDDL(X)_ATT_SKIPAUTH flag. This flag directs R_datalib to bypass authorization checks to RACF key rings for a supervisor state or system key caller. This does not bypass the authorization required in order to retrieve private key information, nor does this bypass authorization checks for PKCS #11 tokens. This flag is ignored for problem state callers.

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- The CheckStatus (X'04') function

  X'20000000' - CDDL(X)_ATT_SKIPAUTH flag. This flag directs R_datalib to bypass authorization checks to RACF key rings for a supervisor state or system key caller. This does not bypass the authorization required in order to retrieve private key information, nor does this bypass authorization checks for PKCS #11 tokens. This flag is ignored for problem state callers.

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- The IncSerialNum (X'06') function

  X'80000000' - CDDL(X)_ATT_SET_MIN_SERIAL flag. This flag is used to indicate that the last used serial number field (CERTLUER) is to be incremented to at least the input serial number parameter. When this flag is set the serial number will only be changed if the current actual value is less than the input serial number value.

- The GetUpdateCode (X'05) function

  X'20000000' - CDDL(X)_ATT_SKIPAUTH flag. This flag directs R_datalib to bypass authorization checks to RACF key rings for a supervisor state or system key caller. This does not bypass the authorization required in order to retrieve private key information, nor does this bypass authorization checks for PKCS #11 tokens. This flag is ignored for problem state callers.

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- The NewRing (X'07') function

  X'80000000' - CDDL(X)_ATT_REUSE_RING flag. This flag directs R_datalib to reuse the existing key ring and remove all the certificates from it. When this flag is off, it indicates the creation of a new key ring.

All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- The DataPut (X'08') function

  X'80000000' - CDDL(X)_ATT_TRUST flag. This flag is used to add certificates, with the TRUST status. When this flag is off, RACF will determine the status based on the following factors in the same way that the RACDCERT ADD command behaves:

  – Whether the issuer of the certificate is trusted
  – Whether the signature of the certificate can be verified
  – Whether the certificate has expired
  – Whether the validity date range of the certificate is within that of its issuer

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements. If the certificate already exists, turning on this attribute will change its status from NOTRUST to TRUST when connecting it to the key ring. However, if the status is already HIGHTRUST, it will remain unchanged.

  X'40000000' - CDDL(X)_ATT_HIGHTRUST flag. This flag is used to add or change certificates with the HIGHTRUST status if the certificate to be added or changed is under CERTAUTH; otherwise this value will be treated as CDDL(X)_ATT_TRUST, that is, add or change certificates with the TRUST status.

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- The DataRemove (X'09') function

  X'80000000' - CDDL(X)_ATT_DEL_CERT_TOO flag. This flag is used to indicate the deletion of the certificate from the RACF database after being removed from the ring, if the certificate is not connected to any other rings. When this flag is off, it indicates the removal of the certificate from the key ring only. When this attribute is specified and the DIGTCERT class is RACLISTed, a successful DataRemove returns 4 4 12 instead of 0 0 0 to indicate that a DataRefresh call is needed.

  All other bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

- All other functions

  All bit positions are reserved and must be set to zero to ensure compatibility with future enhancements.

RACF_user_ID

The name of a 9-byte area that consists of a 1-byte length field followed by up to 8 characters for the user ID.

The user ID is case sensitive. For normal user IDs, the value must be specified in uppercase.

For the DataGetFirst function code, the RACF-reserved user IDs (“irrcerta” or “irrsitec” in lowercase) or their alternate forms (*AUTH* or *SITE* in uppercase) can also be specified.

The value “*TOKEN*” can be specified to indicate that a z/OS PKCS #11 token, rather than a RACF key ring, is the target of the operation. If the length is not specified, it must be equal to 0. If the current user ID is not specified, it is equal to the user ID of the ring owner or the token owner.

Ring_name
The name of a variable length input area that consists of a 1-byte length field followed by up to 237 characters that represent the ring name. The syntax rules for this field, are the same as those for the RACDCERT command. The value specified is case sensitive. The Ring_name is used for the following functions: DataGetFirst, GetUpdateCode, NewRing, DataPut, DataRemove, and DelRing.

For DataGetFirst and GetUpdateCode, the Ring_name can represent a real or virtual key ring or a z/OS PKCS #11 token name. A real key ring must be explicitly created before being used by this service. A virtual key ring is the collection of certificates assigned to a given user ID, including the RACF reserved user IDs for CERTAUTH (“irrcerta” or “*AUTH*”) and SITE (“irrsitec” or “*SITE*”). A virtual key ring can be specified by coding an asterisk (“*”) for the Ring_name with the corresponding RACF_user_ID, such as user01/* or *SITE*/*.

A z/OS PKCS #11 token is a collection of certificates and keys similar to a key ring, but it is managed by the Integrated Cryptographic Service Facility (ICSF). Like a real key ring, it must be explicitly created before being used by this service.

For DataGetNext, the Ring_name is specified on the initial DataGetFirst call and cannot be re-specified.

For CheckStatus, DataAbortQuery, IncSerialNum, and DataGetNext, the Ring_name field is ignored.

parm_list_version
A 4-byte input value which contains the version number for the following field, parm_list. This field must be set to zero.

Parm_list
Specifies the address of the function-specific parameter list for the function specified in Function_code. This information is defined in the following sections of function-specific parameter lists.

Function-specific parameter lists

Depending on the setting of the Function_code paramter, a number of function-specific parameters may apply. Table Table 57 on page 171 summarizes the parameter usage for the function-specific parameters for each of the possible function codes. The function codes are:

- X'01' – DataGetFirst
- X'02' – DataGetNext
- X'03' – DataAbortQuery
- X'04' – CheckStatus
- X'05' – GetUpdateCode
- X'06' – IncSerialNum
- X'07' – NewRing
- X'08' – DataPut
- X'09' – DataRemove
- X'0A' – DelRing
- X'0B' – DataRefresh

There are no function-specific parameters for Function codes X'07', X'0A', and X'0B', so these function codes are not shown in this table.

Table 57. R_datalib function specific parameter usage

Parameter	X'01'	X'02'	X'03'	X'04'	X'05'	X'06'	X'08'	X'09'
Results_handle	Input	Input	Input	n/a	n/a	n/a	n/a	n/a
Certificate_Usage	Output	Output	n/a	n/a	n/a	n/a	Input	n/a
Default	Output	Output	n/a	n/a	n/a	n/a	Input	n/a
Certificate_length	Input / Output	Input / Output	n/a	Input	n/a	Input	Input	n/a
Certificate_ptr	Input	Input	n/a	Input	n/a	Input	Input	n/a
Private_key_length	Input / Output	Input / Output	n/a	n/a	n/a	n/a	Input	n/a
Private_key_ptr	Input	Input	n/a	n/a	n/a	n/a	Input	n/a
Private_key_type	Output	Output	n/a	n/a	n/a	n/a	n/a	n/a
Private_key_bitsize	Output	Output	n/a	n/a	n/a	n/a	n/a	n/a
Label_length	Input / Output	Input / Output	n/a	n/a	n/a	n/a	Input / Output	Input
Label_ptr	Input	Input	n/a	n/a	n/a	n/a	Input / Output	Input
CERT_user_ID	Output	Output	n/a	n/a	n/a	n/a	Input / Output	Input
Subjects_DN_length	Input	Input	n/a	n/a	n/a	n/a	n/a	n/a
Subjects_DN_ptr	Input	Input	n/a	n/a	n/a	n/a	n/a	n/a
Record_ID_length	Input / Output	Input / Output	n/a	n/a	n/a	n/a	n/a	n/a
Record_ID_ptr	Input	Input	n/a	n/a	n/a	n/a	n/a	n/a
Ring_sequence_number	n/a	n/a	n/a	n/a	Output	n/a	n/a	n/a
Serial_number	n/a	n/a	n/a	n/a	n/a	Input / Output	n/a	n/a

Function-specific parameter list for DataGetFirst and DataGetNext:

Results_handle
An address pointing to an input area. The input area must be at least 20 bytes in length, and it is that input area (not Results_handle) that is mapped as follows:

dbToken
A 4-byte value reserved for use by RACF. This value must be preserved for subsequent calls to DataGetNext and DataAbortQuery.

Number_predicates
A 4-byte integer input value. A zero, X'00000000', indicates that there are no selection criteria. This value is only used on a DataGetFirst function. A one, X'00000001', indicates that a query on a particular attribute is being performed. All other values result in a parameter error.

Attribute_ID
A 4-byte integer input value which identifies the attribute that is being queried. This field is ignored if Number_predicates is zero, X'00000000'. If Number_predicates is one, X'00000001', this field must have one of the values listed below:

X'00000001'
Attribute data to match on is label.

X'00000002'
Attribute data to match on is default flag. Attribute data supplied by Attribute_length or Attribute_ptr will either be zero, X'00000000' or non-zero.

X'00000003'
Attribute data to match on is the DER-encoded subject's distinguished name.

If the Attribute_ID is not one of the preceding values, a parameter list error is returned.

Attribute_length
A 4-byte input value containing the length of the attribute data. This field is ignored if Number_predicates is zero, X'00000000'.

Attribute_ptr
An input address pointing to the attribute data. This field is ignored if Number_predicates is zero, X'00000000'.

Certificate_Usage
A 32 bit output flag value, indicating the usage of the certificate. Certificate_Usage can have the following values:

X'00000002'
Certauth

X'00000008'
Personal

X'00000000'
Other (site)

X'fffffff5'
Reserved

Default
A 4-byte output value. A X'00000000' value indicates that this certificate is not the default certificate for the ring. A non-zero value indicates that this is the default certificate for the ring.

Certificate_length
A 4-byte value containing the length of the certificate. On input, it contains the length of the field pointed to by certificate_ptr. On output, it contains the actual size of the certificate that is returned. A zero indicates that no certificate was returned.

Certificate_ptr
An input value containing the address of the DER encoded certificate output area.

Private_key_length
A 4-byte value containing the length of the private key. On input, it contains the length of the field pointed to by private_key_ptr. On output, it contains the actual size of the private key that is returned. A zero indicates that no private_key was returned.

If private_key_length is zero, then private_key_bitsize and private_key_type are not returned.

Private_key_ptr
An input value containing the address of the private key output area.

Private_key_type
A 4-byte output value indicating the form of the private key. The valid values are:

X'00000001'
PKCS #1 private key, DER encoded

X'00000002'
ICSF key token label. If the first character is an '=' sign, it is a key token from the TKDS, otherwise it is from the PKDS.

X'00000003'
PCICC key token label

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, PCICC key labels are returned as ICSF key labels.

X'00000004'
DSA private key, DER encoded

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, DSA keys are returned as PKCS #1 keys.

X'00000006'
Diffie-Hellman (DH) private key, DER encoded

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, DH keys are returned as PKCS #1 keys.

X'00000007'
ECC private key, DER encoded

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, ECC keys are returned as PKCS #1 keys.

X'00000009'
ECC key token label in the PKDS

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, ECC keys are returned as ICSF key label (type 2).

X'0000000B'
RSA key token label in the TKDS

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, RSA TKDS labels are returned as ICSF key labels (type 2).

X'0000000D'
ECC key token label in the TKDS

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, ECC TKDS labels are returned as ICSF key labels (type 2).

X'0000000E'
DSA key token label in the TKDS

Note: This value is returned only when the CDDL(X)_ATT_ALL_KEYTYPES attributes flag is set. If it is not set, DSA TKDS labels are returned as ICSF key labels (type 2).

Private_key_bitsize
A 4-byte output value indicating the size of the private key, expressed in bits.

Label_length
A 4-byte value containing the length of the label. On input, it contains the length of the field pointed to by label_ptr. This field must be at least 32 bytes long. On output, it contains the actual size of the label that is returned. A zero value indicates that no label was returned.

Label_ptr
An input value containing the address of the label output area.

CERT_user_ID
A 9-byte output area containing a 1-byte length, followed by the user ID which owns the certificate.

The 1-byte length must specify a length of 8. The user ID must be left-justified and padded with blanks.

Subjects_DN_length
A 4-byte input value containing the length of the DER encoded subject's distinguished name. On input, it contains the length of the field pointed to by Subjects_DN_ptr. On output, it contains the actual size of the subject's distinguished name that is returned.

Subjects_DN_ptr
a 4-byte input value containing the address of the subject's distinguished name output area.

Record_ID_length
On input, it contains the length of the field pointed to by Record_ID_ptr. This field must have a length of at least 246 bytes. On output, it contains the actual size of the record ID that is returned. A zero value indicates that no record ID was returned.

Record_ID_ptr
An input value pointing to a caller-provided 246-byte output area. This output area contains the record_ID returned from the callable service.

Function-specific parameter list for DataAbortQuery:

Results_handle
The address value, which was returned from a prior DataGetFirst or DataGetNext request. This value is provided by the caller. The data area pointed to by Results_handle must have its fields preserved from prior DataGetFirst and DataGetNext calls.

Note: A DataAbortQuery call is required regardless of the return and reason codes from the corresponding DataGetFirst call.
The DataAbortQuery function requires no authority.

Function-specific parameter list for CheckStatus:

Certificate_length
The 4-byte input value containing the length of the certificate.

Certificate_ptr
The input address value containing the address of the DER encoded certificate.

Function-specific parameter list for GetUpdateCode:

Ring_sequence_number
A 4-byte output field containing the sequence number for the ring specified.

Function-specific parameter list for IncSerialNum:

Certificate_length
The 4-byte input value containing the length of the certificate.

Certificate_ptr
The input address value containing the address of the DER encoded certificate.

Serial_number
The 8-byte output area to contain the returned serial number. This serial number is also an input field when the CDDL(X)_ATT_SET_MIN_SERIAL flag is set.

Function-specific parameter list for NewRing: None.

Function-specific parameter list for DataPut:

Certificate_Usage
A 32-bit flag value that indicates the usage of the certificate. Certificate_Usage can have the following values:

- X'00000002' CERTAUTH
- X'00000008' PERSONAL
- X'00000000' Other (SITE)
- X'80000000' The usage of the certificate itself (by default)

Certificates owned by a regular user ID have PERSONAL usage; certificates owned by SITE have SITE usage; certificates owned by CERTAUTH have CERTAUTH usage.

Default
A 4-byte value. An X'00000000' value indicates that this certificate is not the default certificate for the ring. A non-zero value indicates that this is the default certificate for the ring.

Certificate_length
A 4-byte value containing the length of the certificate.

Certificate_ptr
A value containing the address of the DER-encoded X.509 certificate.

Private_key_length
A 4-byte value containing the length of the private key or a key label (in case of a PKDS hardware key). If it is a key label, the maximum length is 64 characters. A zero indicates no input private key.

Private_key_ptr
A value containing the address of a DER-encoded private key or a key label, if Private_key_length is not zero. If it is a key label, Private_key_ptr must already exist. This parameter is ignored if the certificate already resides in RACF and already has a private key associated with it. (The key will not be updated.) If the certificate already resides in RACF but there is no private key associated with it, the certificate will be re-added with this Private_ key_ptr.

Label_length
A 4-byte value containing the length of the label that will be assigned to the certificate to be added. A zero value indicates that the certificate will be added to RACF before being connected to the key ring. The certificate will be added with a system-assigned label, in the form of 'LABELnnnnnnnn', under the ID specified by CERT_user_ID . The maximum label length is 32 bytes. On return, this parameter will be updated with the actual label length of the certificate if it already resides in RACF; otherwise it will be updated with the system-assigned label length if a system-assigned label is created.

Label_ptr
A value containing the address of the certificate label. On return, the label that is pointed to by this parameter will be updated with the existing certificate label if the certificate already resides in RACF; otherwise it will be updated with the system-assigned label if a system-assigned label is created. The area it points to must be 32 bytes.

CERT_user_ID
A 9-byte area containing a 1-byte length of the user ID, followed by the user ID to which the certificate will be added. The user ID must be left-aligned and padded with blanks. For CERTAUTH certificates, this parameter should be set to either "irrcerta" or "*AUTH*". For SITE certificates, this parameter should be set to either "irrsitec" or "*SITE*". If the length is zero, the value of the RACF_user_ID is used. If the RACF_user_ID is not specified, the caller's user ID is used. This parameter is updated with the certificate owner ID if the certificate already resides in RACF. In the cases of CERTAUTH, 'irrcerta' is returned. In the cases of SITE, "irrsitec" is returned.

Function-specific parameter list for DataRemove:

Label_length
A 4-byte value containing the length of the certificate label. The maximum value is 32.

Label_ptr
A 4-byte value containing the address of the certificate label.

CERT_user_ID
A 9-byte area containing a one-byte length of the user ID, followed by the user ID. CERT_user_ID indicates the owner of the certificate. The user ID must be left-aligned and padded with blanks. For CERTAUTH certificates, this parameter should be set to either 'irrcerta' or '*AUTH*'. For SITE certificates, this parameter should be set to either 'irrsitec' or '*SITE*'. If the length is 0, the value of RACF_user_ID is used, and if the RACF_user_ID is not specified, the caller's ID is used.

Function-specific parameter list for DelRing: None.

Function-specific parameter list for DataRefresh: None.

Return and reason codes

R_datalib might return one of several values as the SAF and RACF return and reason codes. This section defines the return codes that can be returned by any of the functions. In addition, each function of R_datalib has its own unique set of return codes, which are also listed in this section.

Table 58. R_datalib return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred. Attributes were not specified as 0 or the last word in the parameter list did not have the higher order bit on.
8	8	8	Not RACF-authorized to use the requested service.
8	8	12	Internal error caused recovery to get control.
8	8	16	Unable to establish a recovery environment.
8	8	20	Requested Function_code not defined.
8	8	24	Parm_list_version number not supported.
8	8	28	Error in Ring_name length or RACF_userid length.
8	8	72	Caller not in task mode.
8	8	92	Other internal error.
8	8	96	The linklib (steplib or joblib) concatenation contains a non-APF authorized library.
8	12	0x00xxyyyy	An unexpected error is returned from ICSF. The hexadecimal reason code value is formatted as follows: • xx - ICSF return code • yyyy - ICSF reason code

Function-specific reason and return codes for DataGetFirst and DataGetNext:

Table 59. DataGetFirst and DataGetNext return codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	32	Length error in attribute_length, Record_ID_length, label_length, or CERT_user_ID.
8	8	36	dbToken error. The token may be zero, in use by another task, or may have been created by another task.
8	8	40	Internal error while validating dbToken.
8	8	44	No trusted certificate found.
8	8	48	An output area is not long enough. One or more of the following input length fields were too small: Certificate_length, Private_key_length, or Subjects_DN_length. The length field(s) returned contain the amount of storage needed for the service to successfully return data.

Table 59. DataGetFirst and DataGetNext return codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	52	Internal error while obtaining record private key data.
8	8	56	Parameter error - Number_predicates or Attribute_ID in error.
8	8	80	Internal error while obtaining the key ring or z/OS PKCS #11 token certificate information or record trust information.
8	8	84	The key ring profile for RACF_user_ID/Ring_name or z/OS PKCS #11 token is not found, or the virtual key ring user ID does not exist.

Function-specific reason and return codes for DataAbortQuery:

Table 60. DataAbortQuery return codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	36	dbToken error. The token might be zero, in use by another task, or might have been created by another task.
8	8	40	Internal error while validating dbToken.

Function-specific reason and return codes for CheckStatus:

Table 61. CheckStatus return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Certificate is trusted or certificate not registered with RACF.
8	8	60	Internal error - Unable to decode certificate.
8	8	64	Certificate is registered with RACF as not trusted.
8	8	68	Parameter error - zero value specified for Certificate_length or Certificate_ptr.

Function-specific reason and return codes for GetUpdateCode:

Table 62. GetUpdateCode return codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	84	The key ring profile for RACF_user_ID/Ring_name or z/OS PKCS #11 token is not found, or the virtual key ring user ID does not exist.
8	8	88	Internal error - Unable to obtain the key ring or the z/OS PKCS #11 token data.

Function-specific reason and return codes for IncSerialNum:

Table 63. IncSerialNum return codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Success, serial number returned.
8	8	76	Certificate is invalid.
8	8	80	Certificate is not installed or is marked NOTRUST, or does not have a private key.
8	8	84	Parameter error. A zero value was specified for one of the following parameters: • a certificate length or certificate owned by another user. • a minimum serial number.

Function-specific reason and return codes for NewRing:

Table 64. NewRing return codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	32	The profile for Ring_name is not found for REUSE.
8	8	36	The profile for Ring_name already exists. REUSE was not specified.
8	8	40	The Ring_name is not valid.
8	8	44	The RACF_user_ID is not valid or not found.

Function-specific reason and return codes for DataPut:

Table 65. DataPut return codes

SAF return code	RACF return code	RACF reason code	Explanation
4	4	0	Success but the certificate's status is NOTRUST.
4	4	4	Success but the DIGTCERT class needs to be refreshed to reflect the update.

Table 65. DataPut return codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
4	4	8	Success but the Label information is ignored because the certificate already exists in RACF.
4	4	12	Success but the Label information is ignored because the certificate already exists in RACF, and its status is NOTRUST.
4	4	16	Success but the Label information is ignored because the certificate already exists in RACF, and the DIGTCERT class needs to be refreshed to reflect the update.
8	8	32	Parameter error - incorrect value specified for Certificate_length or Certificate_ptr, or the label area is too small.
8	8	36	Unable to decode the certificate.
8	8	40	The private key is neither of a DER encoded format nor of a key label format.
8	8	44	Bad encoding of private key or unsupported algorithm or incorrect key size.
8	8	48	The specified private key does not match the existing private key.
8	8	52	Cannot find the key label.
8	8	56	ICSF error when trying to find the key label.
8	8	60	Not authorized to access ICSF key entry.
8	8	64	The specified certificate label already exists in RACF.
8	8	68	The user ID specified by CERT_user_ID does not exist in RACF.
8	8	76	The certificate cannot be installed.
8	8	80	The certificate exists under a different user.
8	8	84	Cannot find the profile for Ring_name.

Function-specific reason and return codes for DataRemove:

Table 66. DataRemove return codes

SAF return code	RACF return code	RACF reason code	Explanation
4	4	0	Success but cannot delete the certificate from RACF because it is connected to other rings.
4	4	4	Success but cannot delete the certificate from RACF because of an unexpected error.
4	4	8	Success but cannot delete the certificate from RACF because of insufficient authority.
4	4	12	Success but the DIGTCERT class needs to be refreshed to reflect the update.
4	4	16	Success but cannot delete the certificate because it has been used to generate a request through RACDCERT GENREQ.
8	8	32	Parameter error – incorrect value specified for Label_length, Label_ptr or CERT_user_ID.
8	8	36	Cannot find the certificate with the specified label and owner ID.
8	8	40	The profile for Ring_name is not found.

Function-specific reason and return codes for DelRing:

Table 67. DelRing return codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	32	The profile for Ring_name is not found.

Function-specific reason and return codes for DataRefresh:

Table 68. DataRefresh return codes

SAF return code	RACF return code	RACF reason code	Explanation
4	4	0	The refresh is not needed.

Usage notes

1. For real key rings, a certificate's ring usage is set when the certificate is connected to the key ring.
2. For virtual key rings, all certificates within the ring have the same usage as follows:

- CERTAUTH for the CERTAUTH virtual key ring (RACF reserved user ID irrcerta or *AUTH*).
- SITE for the SITE virtual key ring (RACF-reserved user ID irrsitec or *SITE*).
- PERSONAL for the virtual key rings of all other non-reserved user IDs.

3. For z/OS PKCS #11 tokens, a certificate's token usage is set when the certificate is bound to the token.
4. Applications can call the R_datalib callable service (IRRSDL00) to extract the private keys from certain certificates after they have access to the key ring. A private key is returned only when the following conditions are met:
   a. For RACF real key rings:
      - User certificates

        An application can extract the private key from a user certificate if the following conditions are met:
        - The certificate is connected to the key ring with the PERSONAL usage option.
        - One of the following two conditions is true:
          - The caller's user ID is the user ID associated with the certificate if the access to the key ring is through the checking on IRR.DIGITCERT.LISTRING in the FACILITY CLASS, or
          - The caller's user ID has READ or UPDATE authority to the <ringOwner>.<ringName>.LST resource in the RDATALIB class. READ access enables retrieving one's own private key, UPDATE access enables retrieving other's.
      - CERTAUTH and SITE certificates

        An application can extract the private key from a CERTAUTH or SITE certificate if the following conditions are met:
        - The certificate is connected to its key ring with the PERSONAL usage option.
        - One of the following three conditions is true:
          - The caller's user ID is RACF special regardless of access checking method, or
          - The caller's user ID has CONTROL authority to the IRR.DIGTCERT.GENCERT resource in the FACILITY class if the access to the key ring is through the checking on IRR.DIGITCERT.LISTRING in the FACILITY CLASS, or
          - The caller's user ID has CONTROL authority to the <ringOwner>.<ringName>.LST resource in the RDATALIB class.
   b. For RACF virtual key rings:

      An application can extract the private key from a user certificate if either of the following conditions is met:
      - The caller's user ID is the user ID associated with the certificate if the access to the key ring is through the checking on the IRR.DIGITCERT.LISTRING in the FACILITY CLASS, or
      - The caller's user ID has READ or UPDATE authority to the <virtual ring owner>.IRR_VIRTUAL_KEYRING.LST resource in the RDATALIB class. READ access enables retrieving one's own private key, UPDATE access enables retrieving other's.

      Note: Private keys can never be extracted from the CERTAUTH or SITE virtual key ring.

c. For z/OS PKCS #11 tokens:

   An application can extract the private key from a user certificate if all of the following conditions are met:

   - The certificate's token usage is PERSONAL.
   - The caller has permission to read private objects in the token, as determined by ICSF.
   - A private key object exists for the certificate (CKA_ID attributes match).
   - The private key object contains all the attributes defined in PKCS #1.

5. The DataAbortQuery function must be called once for each DataGetFirst call, whether or not DataGetNext calls are made between the DataGetFirst and DataAbortQuery calls. The caller must pass the same dbToken to DataAbortQuery call as was returned from the DataGetFirst call. If these conditions are not met, system resources will not be freed.
6. ICSF services must be loaded from an APF-authorized library when they are required. If the ICSF library is part of the STEPLIB or JOBLIB concatenation, the entire concatenation must be APF-authorized.

Related services

None

R_dceauth (IRRSDA00): Check a user's authority

Function

The R_dceauth service enables an application server to check a RACF-defined user's authority to access a RACF-defined resource. It is intended to be used only by the z/OS UNIX kernel on behalf of an application server.

The client's identity can be specified by:

- The ACEE
- The *RACF_userid* parameter
- The cell and principal UUID parameter pair

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR ASC mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

RACF authorization

1. RACF checks the ACEE, *RACF_userid*, and the UUID parameters in the following order:
   - If the ACEE parameter has been specified, this parameter is used to identity the user for this authorization request.
   - If the ACEE parameter has not been specified, and the *RACF_userid* parameter is present, this parameter is used to identify the user for this authorization request.
   - If neither the ACEE nor the *RACF_userid* parameter is present, the *Cell_string_uuid* and the *Principal_string_uuid* parameters are used to identify the user for this authorization request.
   - If the ACEE, *RACF_userid*, and UUID parameters have not been supplied, RACF uses the current task-level ACEE if it is found. If there is no task-level ACEE, RACF uses the address-space ACEE, if it is present, to identify the user for this authorization request.

Format

```
CALL IRRSDA00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
                     ACEE_ptr,
               ALET, Principal_string_uuid,
                     Cell_string_uuid,
                     RACF_userid,
                     RACF_class,
                     entity_name,
                     entity_length,
                     access_requested
             )
```

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 ALET can be different. The words 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.

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.

ACEE_ptr
The name of a fullword containing the address of an area that contains a

previously created ACEE. If the caller does not specify an ACEE, this area must contain binary zeros. The ACEE parameter is **not** specified by an ALET. This parameter must be in the primary address space.

ALET
The name of a word which must be in the primary address space and which contains the ALET for the following fields:

- Principal_string_uuid
- Cell_string_uuid
- RACF_userid
- RACF_class
- Entity_name
- Entity_length
- Access_requested

Principal_string_uuid
The name of an area containing the string form of the client's DCE UUID. If the caller does not specify the client's DCE UUID, then the first character of the area must be a NULL byte. (That is, the first byte of the 36-byte area must contain X'00'.)

Cell_string_uuid
The name of an area containing the string form of the cell DCE UUID. The string form of the cell UUID, if supplied by the caller, must be 36 bytes long.

The *Cell_string_uuid* must be the name of a 36-byte area that contains one of the following:

- The string form of the cell UUID
- A null byte (X'00') as the first character of the 36-byte area. If the home cell UUID is not applicable and the caller wants to obtain cross linking information using only the DCE principal UUID, the caller must pass the *Cell_string_uuid* parameter with the first byte of this field containing a null byte of X'00'.

RACF_userid
The name of a 9-byte area, which consists of a 1-byte length field followed by up to eight characters. It must be specified in uppercase. If not specified, the length must equal zero.

RACF_class
The name of an 8-byte area containing the RACF class name of the resource (such as TAPEVOL). The class name must be

- Left justified
- Padded to the right with blanks
- Specified in uppercase
- Defined to RACF in the RACF class descriptor table.

Entity_name
The name of an area that contains the RACF entity or resource name (such as TAPE01). It must be specified in uppercase and should not contain generic characters.

Entity_length
The name of an area that contains the halfword length of the entity_name. The valid range of this parameter is 1 to 246 characters.

Access_requested
The requested access (READ, UPDATE, CONTROL, ALTER) to the resource, which is the name of a 1-byte area containing:

Requested access	Value
READ Access	X'02'
UPDATE Access	X'04'
CONTROL Access	X'08'
ALTER Access	X'80'

Return and reason codes

IRRSDA00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Access granted
4	0	0	RACF not installed, or RACF is not active
8	8	4	The resource is not defined to RACF
8	8	8	User is not authorized to access the resource
8	8	12	Internal processing error
8	8	16	Recovery environment could not be established
8	8	20	No mapping to a RACF user ID exists for the supplied UUID pair
8	8	24	Parameter list error
8	8	28	DCEUUIDS class is not active. RACF is not able to map the supplied UUIDs to a RACF user ID.
8	8	32	RACF was unable to create a security environment for the user ID specified.
8	8	36	The user ID is not RACF defined.

Usage notes

1. This service may **not** be used to determine access to z/OS UNIX resources, such as shared UNIX files or data sets.
2. The DATASET class is not valid for this service.

Parameter	Direction	Value
SAF_return_code	Output	
RACF_return_code	Output	
RACF_reason_code	Output	
ACEE_ptr	Input	Optional
Principal_string_uuid	Input	Optional
Cell_string_uuid	Input	Optional

Parameter	Direction	Value
RACF_userid	Input	Optional
RACF_class	Input	Required
entity_name	Input	Required
entity_length	Input	Required
access_requested	Input	Required

3. If the user ID parameter is specified, the user's default security label, if any, will be used for authorization checking when the SECLABEL class is active.
4. When the class is SETR RACLISTed, RACROUTE REQUEST=FASTAUTH will be issued instead of REQUEST=AUTH to increase performance. There are some differences between FASTAUTH and AUTH:
   - Regarding authorization checking

     If the ACEE specified as input to FASTAUTH does not grant authority to the specified resource, and that ACEE has a nested ACEE and the user has authority to use it, then the nested ACEE is also checked to see if it grants authority. If so, authority is granted. AUTH does not use nested ACEEs.
   - Regarding parmlist checking

     FASTAUTH does very little parmlist checking because it constitutes a performance path. There may be some unusual situations that give different results than AUTH. For instance, an input entity length that is greater than the maximum profile length for the class will result in an abend for an AUTH request and an “8/8/12 internal error” RC , while a FASTAUTH request will use the maximum profile length as the entity length, likely resulting in an “8/8/4 profile not found” RC.
   - Regarding logging

     AUTH uses SETR LOGOPTIONS settings in determining when to create an audit record, while FASTAUTH does not.

Related services

R_dceruid

R_dceinfo (IRRSDI00): Retrieve or set user fields

Function

The RACF R_dceinfo callable service retrieves or sets the following fields from a user profile **DCE** segment:

- The DCE principal name associated with this RACF user
- DCE UUID of this user
- DCE cell name that this user is defined to (HOME CELL)
- DCE cell UUID that is associated with DCE cell that this user is defined to (HOMEUUID)
- A flag byte that indicates whether z/OS DCE creates a DCE security context (autologin) automatically

The action performed by this callable service is based on the function code passed by the caller in the R_dceinfo parameter list.

- When the function code is set to EXTRACT, R_dceinfo retrieves the information requested from the user's DCE segment.

- When the function code is set to REPLACE, R_dceinfo replaces the fields that have been specified in the parameter list.

Requirements

Authorization:
Any PSW key in: Supervisor state for REPLACE DCE fields Supervisor or problem state for EXTRACT DCE fields

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR ASC mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. The replace function fails if the user ID specified in the parameter has not been previously defined as a DCE RACF user.
2. Field level access checking does not occur when retrieving or replacing fields with this service.

Format

```
CALL IRRSDI00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Function_code,
               ALET, RACF_userid,
               ALET, Field_list,
               ALET, Output_area,
               ALET, Output_area_length
              )
```

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 words containing the ALETs must be in the primary address space.

SAF_return_code
The name of a full word 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.

Function_code
The name of a 1-byte area containing the function code:

X'01' Retrieve DCE fields

X'02' Replace DCE fields

RACF_userid
The name of a 9-byte area, containing a 1-byte length field, followed by a user ID up to eight characters long. It must be specified in uppercase.

Field_list
The name of an area containing the fields to be replaced or retrieved. The format of the parameter list is:

Offset	*Length*	*Description*
0	2	The length in bytes of the DCE field list
2	2	Total number of fields in the DCE field list
4	8	The name of the field
12	2	The length of the field data
14	variable	Field data

The ordered triplet (name of the field, length of the field, and field data) is a repeating data structure. This triplet can repeat for the total number of fields in the input Field_list.

- If the function code is **X'01'** (**retrieve DCE fields**), the caller is expected to provide the following fields in the **Field_list**:
  - The total length in bytes of the input field list
  - The total number of fields to be retrieved
  - The name of the fields to be retrieved
  - The length of the data

    Note:

    For the retrieve function, there is no input field data. The caller is expected to provide a length of binary zero.

    The name of the field and the place holder of zero may repeat for the total number of fields to be retrieved.
- If the function code is **X'02'** (**replace DCE fields**), the caller is expected to be in supervisor state and to provide the following fields in the **Field_list**:
  - The total length in bytes of the input field list

- The total number of fields to be retrieved
- The name of the field to be retrieved
- The length of the data
- Field data

A problem-state caller is not authorized to replace DCE information.

Output_area
The name of an area that contains the fields obtained by the R_dceinfo service when the function code is **X'01'** (**Retrieve DCE fields**). The format of the output area is:

Offset	*Length*	*Description*
0	2	Total length of the output area of the retrieved data
2	2	Number of fields retrieved
4	8	Name of the retrieved field
12	2	Length of the retrieved field
14	variable	Field data

The ordered triplet (name of the field, length of the field, and field data) is a repeating data structure. This triplet can repeat for the number of times shown in the output_area 'number of fields retrieved' count.

Output_area_length
The name of a fullword that contains the length of the output_area that is supplied by the caller of this service.

Return and reason codes

IRRSDI00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Service successful.
4	0	0	RACF not installed.
8	8	4	Caller is not authorized.
8	8	12	Internal error during RACF processing.
8	8	16	Recovery Environment could not be established.
8	8	20	User does not have a DCE segment.
8	8	24	Length of the output area is too small to contain the data retrieved.
8	8	28	Parameter list error.
8	8	32	User ID specified does not exist.

Usage notes

1. If the caller is in problem state, the *RACF_userid* specified must be the same RACF user as found in either the task level ACEE or the address space level ACEE.

2. If the caller is in supervisor state, the task and address space ACEEs are not checked. Therefore, an authorized caller may extract or replace DCE segment fields for any user who has a DCE segment.
3. The retrieve function returns fields that have been previously populated. Associated with the returned fields is a length indicator. The length indicator is set to zero if a field does not exist.
4. It is the responsibility of the caller to obtain and free the output area. If the fields to be retrieved from RACF are larger than the output area, RACF fails the request and returns the actual length required in the *output_area_length* parameter.
5. The field names supplied by the caller may be:
   - UUID
   - DCENAME
   - HOMECELL
   - HOMEUUID
   - DCEFLAGS

   The field names must be supplied as 8-character fields, left justified, and padded with blanks. They must be specified in uppercase.
6. The DCEFLAGS field is a 1-byte field with the following meaning:
   - Value of X'00' means that z/OS DCE does *not* attempt to sign on this user to z/OS DCE automatically
   - Value of X'01' means that z/OS DCE attempts to automatically sign on this user to z/OS DCE

Parameter Usage:

Parameter	GET_INFO Direction	PUT_INFO Direction
SAF_return_code	Output	Output
RACF_return_code	Output	Output
RACF_reason_code	Output	Output
Function_code	Input	Input
RACF_userid	Input	Input
Field_list	Input	Input
Output_area	Output	n/a
Output_area_length	Output	n/a

Related services

R_dcekey

R_dcekey (IRRSDK00): Retrieve or set a non-RACF password

Function

The RACF R_dcekey callable service enables z/OS DCE to retrieve or set a DCE password (*key*) or retrieve an LDAP bind password. The three functions supported are:

- This service retrieves the DCE password from a DCE segment. The password is decrypted using the key that was stored in the user's DCE segment when the password was encrypted.
- This service sets the DCE password in a user profile DCE segment. The password is encrypted using the key stored in the DCE.PASSWORD.KEY profile in the RACF KEYSMSTR general resource class.
- This service retrieves the LDAP bind password from the PROXY segment of a general resource profile in the LDAPBIND Class or the IRR.PROXY.DEFAULTS profile in the FACILITY Class. The password is decrypted using the key that was stored in the profile's PROXY segment when the password was encrypted (for example, when the RDEFINE or RALTER PROXY(...) command was issued.)

The operation of **R_dcekey** is based on the function code values of **get_key**, **put_key**, and **get_ldap_pw** in the parameter list. See "Usage notes" on page 195 for detailed information.

Requirements

Authorization:
Any PSW key in supervisor state or problem state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR ASC mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

- For function codes get_key and put_key, the user ID specified in the RACF_entity parameter must have a DCE segment.
- For function code get_ldap_pw, the LDAPBIND Class profile specified in the RACF_entity parameter must have a PROXY segment previously created through a RDEFINE or RALTER command. If the RACF_entity is not specified, the IRR.PROXY.DEFAULTS profile in the FACILITY Class must have a PROXY segment previously created through a RDEFINE or RALTER command.
- For callers not running in system key or supervisor state, the use of the R_dcekey service is authorized by FACILITY class resources:

- The ACEE associated with the address space is used to determine the caller. If the caller is running in a clean environment with a RACF user or group that has at least READ authority to the BPX.SERVER resource, use of R_dcekey is permitted and no subsequent access checks are made.
- Otherwise, the current TCB is checked for an ACEE. If one is found, it will be used to determine the caller. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to determine the caller. If the caller is running in a clean environment with a RACF user or group that has at least READ authority to the IRR.RDCEKEY resource, use of R_dcekey is permitted.

  If the FACILITY class is inactive, or the above resources are not defined, only servers running in system key or supervisor state may use the R_dcekey service. For more information about running in a clean environment, see the discussion of Program Control in the *z/OS Security Server RACF Security Administrator's Guide*.

Format

```
CALL IRRSDK00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Function_code,
               ALET, RACF_entity,
               ALET, key_area,
               ALET, key_area_length
              )
```

Parameters

Work_area
The name of a 1024-byte work area for SAF and RACF use. 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 words 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.

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.

Function_code
The name of a 1-byte area containing the function code:

X'01' get_key (retrieve current DCE key)

X'02' put_key (set DCE key)

X'03' get_ldap_pw (get the LDAP bind password from the PROXY segment)

RACF_entity
Formally called RACF_userid. The name of a 247-byte area, which consists of a 1-byte length field followed by up to 246 characters. This field is to contain the RACF entity for the password being set or retrieved.

For functions get_key and put_key, this field is the RACF user ID.

For function get_ldap_pw, this field is a LDAPBIND Class general resource profile name. Setting the length byte to x'00' indicates to retrieve the default LDAP bind password from the IRR.PROXY.DEFAULTS profile in the FACILITY Class.

Key_area
The name of an area containing the DCE key, preceded by a 2-byte length field.

Key_area_length
The name of a fullword that contains the length of the key_area.

Return and reason codes

IRRSDK00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Service successful.
4	0	0	RACF not installed.
8	8	4	Entity not defined to RACF
8	8	8	Caller not authorized to use service
8	8	12	Internal error during RACF processing
8	8	16	Recovery environment could not be established.
8	8	20	Entity missing required segment (DCE or PROXY)
8	8	24	No DCE key or LDAP bind password exists for the entity
8	8	28	The key_area supplied by the caller is too small
8	8	32	Parameter list error
8	8	36	RACF KEYSMSTR inactive, or the profile DCE.PASSWORD.KEY profile was not defined to the RACF KEYSMSTR with a SSIGNON segment
8	8	40	Invalid data was found in SSIGNON segment of the DCE.PASSWORD.KEY profile in the RACF KEYSMSTR
8	8	44	RACF was unable to retrieve or update the master key/master key token in the SSIGNON segment of the DCE.PASSWORD.KEY profile in the RACF KEYSMSTR
8	8	48	Unexpected error returned from RACROUTE which may be caused by specifying the entity incorrectly. Symptom record written
8	8	52	RACF cannot locate the CCA support routine

SAF return code	RACF return code	RACF reason code	Explanation
8	8	56	The invocation of the CCA support routine has failed

Usage notes

1. When the function is get_key, this service returns the current DCE key in clear text form to the output area supplied by the caller. This is the value defined by the *key_area* parameter. The length of the *key_area* is supplied by the *key_area_length* parameter.
2. If the *key_area* supplied by the caller is too small to contain the user's current DCE key or the profile's LDAP password, the service sets the required length in the *key_area_length* parameter.
3. When the function is put_key, this service replaces the current DCE key in the specified user profile DCE segment with the value specified in the *key_area* parameter.
4. When the function is get_ldap_pw, this service returns the LDAP bind password in clear text form to the output area supplied by the caller. This is the value defined by the key_area parameter. The length of the key_area is supplied by the key_area_length parameter.

Parameter usage

Parameter	GET_KEY Direction	PUT_KEY Direction	GET_LDAP_PW Direction
SAF_return_code	Output	Output	Output
RACF_return_code	Output	Output	Output
RACF_reason_code	Output	Output	Output
Function_code	Input	Input	Input
RACF_entity	Input	Input	Input
Key_area	Output	Input	Output
Key_area_length	Input/Output	n/a	Input/Output

Related services

R_dceinfo

R_dceruid (IRRSUD00): Determine the ID of a client

Function

The **R_dceruid** service enables z/OS DCE servers to determine the RACF user ID of the client from the string forms of the client's DCE UUID pair, which consists of the *home cell* UUID and the *principal* UUID. It also enables the servers to determine the DCE UUIDs of a client from the RACF user ID.

Note that this service can *only* convert a DCE UUID to a RACF user ID and a RACF user ID to a DCE UUID for users who have:

- A populated DCE segment associated with their user profile

- A DCEUUIDS-class profile that defines the association between the DCE UUIDs and the RACF user ID

The R_dceruid service is sensitive to the profiles defined to the RACF DCEUUIDS class.

To resolve a conversion request:

- If a caller specifies a UUID pair on this service's invocation to convert the UUID pair to the corresponding RACF user ID, the service searches the RACF DCEUUIDS-class profiles defined to RACF with that UUID pair.
- If a caller specifies only a principal UUID on the service's invocation to convert the principal UUID to the corresponding RACF user ID, the service searches the RACF DCEUUIDS-class profiles defined to RACF with only that principal UUID.

Requirements

Authorization:
Any PSW key in supervisor state or problem state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. This service can only translate a RACF user ID to a DCE UUID and a DCE UUID to a RACF user ID for users who have:
   - A populated DCE segment associated with the user profile
   - A DCEUUIDS-class profile defined to RACF that associates a DCE UUID pair with a RACF user ID
2. Use of the R_dceruid service is authorized by the profile IRR.RDCERUID in the RACF FACILITY class. The user ID of the application server (the RACF identity associated with the application server), or a group to which the server is connected, must be permitted with READ access to the profile IRR.RDCERUID in the RACF FACILITY class. Assigning a UACC of READ to the profile IRR.RDCERUID is not recommended.

Format

```
CALL IRRSUD00 (Work_area,
               ALET,SAF_return_code,
               ALET,RACF_return_code,
               ALET,RACF_reason_code,
               ALET,Function_code,
               ALET,Principal_string_uuid,
               ALET,Cell_string_uuid,
               ALET,RACF_userid
              )
```

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 words 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.

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.

Function_code
The name of a 1-byte area containing the function code:

X'01' Return RACF user ID

X'02' Return DCE UUIDs

Principal_string_uuid
The name of an area containing the string form of the principal DCE UUID. The area must be 36 characters long.

- If the function code is **return RACF user ID**, this area must contain the principal DCE UUID as input.
- If the function code is **return DCE UUIDs**, this area is used as an output area when the principal DCE UUID is returned.

Cell_string_uuid
The name of an area containing the string form of the cell DCE UUID. The string form of the cell UUID, if supplied by the caller, must be 36 bytes long.

- If the function code is `return RACF user ID`, the *Cell_string_uuid* must be the name of a 36-byte area that contains one of the following:
  - The string form of the cell UUID
  - A null byte (X'00') as the first character of the 36-byte area. If the home cell UUID is not applicable and the caller wants to obtain cross linking information using only the DCE principal UUID, the caller must pass the *Cell_string_uuid* parameter with the first byte of this field containing a null byte of X'00'.
- If the function code is `return DCE UUIDS`, this area is used as an output area when the home cell UUID is returned.

RACF_userid
The name of a 9-byte area, which contains a 1-byte length field followed by up to 8 characters. It must be specified in uppercase.

When using this callable service to return the RACF user ID associated with a DCE UUID, if the APPLDATA field in the appropriate DCEUUIDS class profile has not been populated with a RACF user ID, the service returns a 0 in the 1-byte length field.

Return and reason codes

IRRSUD00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Service successful
4	0	0	RACF not installed, or RACF is not active
8	8	0	RACF user ID specified does not exist.
8	8	4	No mapping to a RACF user ID exists for this UUID.
8	8	8	Not authorized to use this service.
8	8	12	Internal error during RACF processing
8	8	16	Recovery environment could not be established.
8	8	20	Local cell DCE UUID could not be determined for this RACF to DCE UUID conversion request.
8	8	24	The RACF DCEUUIDS class is not active. UUID to RACF conversion request could not be performed.
8	8	28	Parameter list error.
8	8	32	No mapping to a UUID exists for this RACF user ID.

Usage notes

1. This callable service allows z/OS DCE servers to associate a DCE UUID pair with a RACF user ID.
   - If the function code is **return RACF user ID**, the **R_dceruid** service returns the RACF user ID associated with the string form of the DCE UUIDs supplied by the caller, if it is available.
   - If the function code is **return DCE UUID**, the **R_dceruid** service returns the string form of the DCE UUIDs associated with the RACF user ID supplied by the caller, if it is available. RACF stores the UUIDs as uppercase characters and therefore returns the UUIDs in uppercase.

Parameter usage

Parameter	UUID to RACF userID Direction	RACF userID to UUID Direction
SAF_return_code	Output	Output
RACF_return_code	Output	Output

Parameter	UUID to RACF userID Direction	RACF userID to UUID Direction
RACF_reason_code	Output	Output
Function_code	Input	Input
Principal_string_uuid	Input	Output
Cell_string_uuid	Input	Output
RACF_userid	Output	Input

Related services

R_dceinfo, R_usermap

R_exec (IRRSEX00): Set effective and saved UIDs/GIDs

Function

The **R_exec** service sets the effective and saved z/OS UNIX user identifiers (UIDs) and z/OS UNIX group identifiers (GIDs) for a process to the specified input values. Input flags indicate whether the UIDs or GIDs or both should be changed.

R_exec returns the new values of the real, effective, and saved UIDs and GIDs in the output areas provided.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSEX00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Flags,
               ALET, UID,
               ALET, GID,
               ALET, UID_output_area,
               ALET, GID_output_area
              )
```

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 words 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.

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.

Flags
The name of a byte containing the settings of the SETUID and SETGID flags for the file being executed. The flags are:

- X'01' - SETUID
- X'02' - SETGID
- X'03' - Both SETUID and SETGID

UID
The name of a fullword containing the z/OS UNIX user identifier (UID) to be set. The UID must be defined to RACF.

GID
The name of a fullword containing the z/OS UNIX group identifier (GID) to be set. The GID must be defined to RACF.

UID_output_area
The name of a 3-word area in which the new real, effective, and saved z/OS UNIX user identifiers (UIDs), in that order, are returned.

GID_output_area
The name of a 3-word area in which the new real, effective, and saved z/OS UNIX group identifiers (GIDs), in that order, are returned.

Return and reason codes

IRRSEX00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The z/OS UNIX user identifier (UID) is not defined to RACF.
8	8	8	The z/OS UNIX group identifier (GID) is not defined to RACF.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. An audit record is optionally written, depending on the options in effect for the system.
3. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

None

R_fork (IRRSFK00): Fork a process

Function

When called from the parent process, the **R_fork** service returns the address, subpool, and key of the storage containing the user security information for the calling process.

When called from the child process, the **R_fork** service returns the address of an area containing a copy of the security information pointed to on the initial call. The storage pointed to by the address is obtained by the subpool and key returned on the previous call.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
None. Caller handles recovery.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSFK00   (Work_area,
                 ALET, SAF_return_code,
                 ALET, RACF_return_code,
                 ALET, RACF_reason_code,
                 ALET, Flag,
                 ALET, Data_key,
                 ALET, Data_address,
                 ALET, Data_length,
                 ALET, Data_subpool
                )
```

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 words 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.

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.

Flag
The name of a fullword flag that describes whether fork processing is being done in the parent's or child's address space:

X'00000000'
Fork processing is being done in parent's address space.

X'00000001'
Fork processing is being done in child's address space.

X'00000002'
Fork processing is being done in parent's address space for additional security information.

X'00000003'
Fork processing is being done in child's address space for additional security information.

Data_key
The name of a word in which:

- The storage key of the parent's USP or additional security information is returned by IRRSFK00 during parent fork processing.
- The storage key of the parent's USP or additional security information is supplied to IRRSFK00 during child fork processing.

Data_address
The name of a word in which the address of:

- The parent's USP or additional security information is returned by IRRSFK00 during parent fork processing.
- A copy of the parent's USP or additional security information is supplied to IRRSFK00 during child fork processing.

Data_length
The name of a word that contains the length of the data addressed by Data_address.

Data_subpool
The name of a word in which:

- The storage subpool for the parent's USP or additional security information is returned by IRRSFK00 during parent fork processing.
- The storage subpool for the parent's USP or additional security information is supplied to IRRSFK00 during child fork processing.

Return and reason codes

IRRSFK00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
0	0	4	Additional security information available.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. The key is meaningful only when the following subpools are used:
   - 129–132
   - 227–231
   - 241
3. The following user security information is propagated from the parent address space to the child address space:
   - Controlled status
   - Keep-controlled indicators
   - Saved messages

Note: Only a subset of this information is copied during the first call to R_fork. A second call may be necessary. RACF reason code 4 indicates that there is additional security information related to the parent address space that should be propagated to the child address space. If a reason code 4 is received when a flag value of 0 was passed, R_fork should be called again with a flag value of 2 specified. If a reason code 4 is received when a flag value of 1 was passed, R_fork should be called again with a flag value of 3 specified.

4. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

None

R_GenSec (IRRSGS00 or IRRSGS64): Generic security API interface

Function

The **R_GenSec** service allows for the invocation of a subset of the GSS-API and PassTicket functions through a native SAF service. The underlying processing of these functions is provided by RACF through the IBM Network Authentication Service on z/OS or by RACF alone.

Requirements

Authorization:
Any PSW key in supervisor or problem state

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31 or 64

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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 first parameter list must have the

number of parameters passed. For the Generic Security API Parameter List (IRRPCOMX), the main parameter mappings will be used by both AMODE 31 and AMODE 64 callers. The address double words from 31 bit callers should have the first word filled with zeros and the second word filled with the 31 bit address. Sub-parameter addresses will be in the format of the AMODE of the caller.

RACF authorization

For callers not running in system key or supervisor state, the use of R_GenSec is authorized by the resource IRR.RTICKETSERV for function code 1 and IRR.GSSERV for function code 2 in the FACILITY class. The application server must be running with a RACF user or group 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_GenSec service.

For all callers, the use of the R_Gensec service to use PassTicket services (function code 3) is authorized by the resources in the PTKTDATA class that correspond to the application ID and target userid used in the PassTicket operation. The application server must be running with a RACF user or group that has the authority specified in the table below. If the PTKTDATA class is inactive, or the resource is not defined, the request will fail because of insufficient authority. All callers, regardless of PSW key or state, must pass the authorization check. Generic profiles can be used for authorization.

Operation	**Profile name**	**Required access**
Generate PassTicket	IRRPTAUTH.*application.target-userid*	UPDATE
Evaluate PassTicket	IRRPTAUTH.*application.target-userid*	READ

See *z/OS Security Server RACF Security Administrator's Guide* for more information about configuring RACF to use PassTicket services.

The PassTicket evaluation function is meant to be used to evaluate PassTicket for users who do not exist in RACF, for example temporary or generated userids. However it can be used with RACF-defined users. There is no revocation of users because of failed password attempts, so you must take care in granting access to the PassTicket evaluation function.

The PassTicket evaluation service only evaluates that a PassTicket is computationally valid for a given userid and application. It does not actually log the user in to the system or create any kind of z/OS security context for that user.

To log in a user using a PassTicket, use a standard z/OS function such as __login() or RACROUTE REQUEST=VERIFY.

Format

31 bit invocation:

```
CALL IRRSGS00 (Num_parms,
               Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Option_word,
               Function_code,
               Function_parm_count,
               Function_parms
              )
```

or 64 bit invocation:

```
CALL IRRSGS64 (Num_parms,
               Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Option_word,
               Function_code,
               Function_parm_count,
               Function_parms
              )
```

Parameters

Num_parms
The name of a fullword containing the number of parameters in the parameter list, including the Num_parms parameter. The number must be 12 for this release.

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 containing the ALET for the following parameter. Each parameter preceded by an ALET parameter must have an ALET specified. Each ALET can be different. The words 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.

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.

Option_word
The name of a fullword containing binary zeros. The area referenced by this parameter is reserved for future use.

Function_code
The name of a 2-byte area containing the function code. The function code is one of the following values:

Value	Description
1	Extract from token

Value	Description
2	Invoke GSS-API service
3	PassTicket function

Function_parm_count
The name of a fullword containing the number of function-specific parameters passed.

Function_parms
The name of an area that contains the remaining parameters for the specific function to be invoked. The list of addresses that make up this area are dependent on the subfunction code and the AMODE of the caller. The addresses are 4 bytes for AMODE 31 callers and 8 bytes for AMODE 64 callers.

Description of data types

Context Handle
24-byte data area

Credential Handle
24-byte data area

Buffer Control Block
12 or 16-byte data area containing 4-byte subpool number, 4-byte data length and data address. The subpool number is used as input and output, the length and data address values are for output only. The length of the address is dependent on the AMODE of the caller.

Note: For unauthorized callers the subpool must be in the range 0-127.

String block
8 or 16-byte data area containing 4-byte length and data address. The length of the address is dependent on the AMODE of the caller. For AMODE 31, it is length followed by 4 byte address. For AMODE 64, it is length followed by 4 bytes of reserved space followed by an 8-byte address.

Expiration Time
4-byte area with value in seconds (hex)

Major/Minor Status
4-bytes of status bits

Note: All output data areas must be supplied to the service so the specified data can be returned. The only area that the service will allocate storage for is the data portion of the Buffer Control Block and is the responsibility of the caller to free at the end of processing. The header of the Buffer Control Block must be supplied by the caller.

Extract client principal functions (Function code 1):

This function provides a similar service to R_ticketserv service function code 1. The major difference is that it can be invoked from AMODE 64.

Subfunction codes:

Value	Description
1	Return principal name

Return principal name (1): This function will extract the Kerberos principal name from the input context token.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word containing the context token length (input)
- Address of a GSS-API context token (input)
- Address of a 24-byte string block for use by the server
- Address of the string block for the principal name (output). The data area supplied should be 240 bytes.
- Address of a word to contain the return code (output)

GSS-API functions (Function code 2):

The context and credential handles used by these services are not the same as the context and credential handles used by the GSS-API C functions and cannot be used interchangeably. The input and output tokens used by these functions, however, are compatible with the input and output tokens used by the GSS-API C functions, which means that a token created by a R_GenSec service can be processed by a GSS-API C functions and vice versa.

All references to addresses in the description of this callable service are considered to be 4 bytes in length for AMODE 31 callers and 8 bytes for AMODE 64 callers.

Storage allocations performed by this service will be in the home address space and will be owned by the task issuing the call to R_GenSec. It is up to the caller to free any storage that is allocated and returned to the caller.

Any GSS-API error code which can be generated by the IBM Kerberos runtime could be returned for the major and minor status code parameters. Refer to the messages and codes section in the *z/OS Integrated Security Services Network Authentication Service Administration* for a description of the various error codes. Refer to the GSS-API function descriptions in the *z/OS Integrated Security Services Network Authentication Service Programming* for a description of the GSS-API functions, associated input and output parameters and status codes.

Flag Values:

Flag name	Value
GSS_C_DELEG_FLAG	1
GSS_C_MUTUAL_FLAG	2
GSS_C_REPLAY_FLAG	4
GSS_C_SEQUENCE_FLAG	8
GSS_C_CONF_FLAG	16
GSS_C_INTEG_FLAG	32
GSS_C_ANON_FLAG	64
GSS_C_PROT_READY_FLAG	128

Status codes:

Status codes are returned in the following format:

Calling Error	Routine Error	Supplementary Information
Bit 0 7 8	15 16	31

Status codes	Bit position within status type
Calling Error	
GSS_S_CALL_INACCESSIBLE_READ	1
GSS_S_CALL_INACCESSIBLE_WRITE	2
GSS_S_CALL_BAD_STRUCTURE	3
Routine Error	
GSS_S_COMPLETE	0
GSS_S_BAD_MECH	1
GSS_S_BAD_NAME	2
GSS_S_BAD_BINDINGS	4
GSS_S_BAD_SIG	6
GSS_S_NO_CRED	7
GSS_S_NO_CONTEXT	8
GSS_S_DEFECTIVE_TOKEN	9
GSS_S_DEFECTIVE_CREDENTIAL	10
GSS_S_CREDENTIALS_EXPIRED	11
GSS_S_FAILURE	13
Supplemental Information	
GSS_S_CONTINUE_NEEDED	0
GSS_S_DUPLICATE_TOKEN	1
GSS_S_OLD_TOKEN	2

Miscellaneous definitions:

Name	Value
GSS_NO_CREDENTIAL	0
GSS_C_NO_BUFFER	0
GSS_C_NO_CONTEXT	0
GSS_C_INDEFINITE	x'FFFFFFFF'

Subfunction codes:

Value	Subfunction
1	Initiate a GSS-API security context
2	Continue initiation of a GSS-API security context
3	Accept a GSS-API security context
4	Delete a GSS-API security context
5	Release a GSS-API credential
6	Get the MIC for a message

Value	Subfunction
7	Verify the MIC for a message
8	Wrap a message
9	Unwrap a message
10	Export a GSS-API security context
11	Import a GSS-API security context
12	Export a GSS-API credential
13	Import a GSS-API credential
14	Acquire a GSS-API initiator credential

Initiate® a GSS-API security context (1)

This function will initiate a GSS-API security context and return a context token. This token would then be sent to the context acceptor. The RACF userid associated with the thread that makes the request will be the owner of the security context. Refer to the description of the gss_init_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the string block for the target principal or service name (input). The name has a maximum length of 240 bytes. A fully-qualified principal name is expressed as /.../realm-name/principal-name. A local principal name can be expressed as principal-name without a realm prefix. A service name is expressed as /:/host-name/service-name.
- Address of a word containing the request flags (input). Refer to the description of the gss_init_sec_context() function for the flag definitions
- Address of a word containing the requested context expiration time in seconds (input). An expiration time of 0 will request the default expiration time of two hours while an expiration time of -1 will request the maximum expiration time.
- Address of a 24-byte credential handle (input). The RACF userid associated with the thread that makes the request must be the credential owner. The credential may have been created on any system in the sysplex. In order to use a credential created on a different system, the Kerberos ticket associated with the credential must not contain a client address list.
- Address of a 24-byte buffer which will contain the context handle for the new security context (output). The caller is responsible for deleting the security context when it is no longer needed.
- Address of a word which will contain the return flags (output). Specify a zero address if the return flags are not needed. Refer to the description of the gss_init_sec_context() function for the flag definitions.
- Address of a word which will contain the actual context expiration time in seconds (output). Specify a zero address if the context expiration time is not needed.

- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.

Continue initiation of a GSS-API security context (2)

This function will continue context initiation using the context token returned by the remote partner after accepting the context. This sub-function is called only if sub-function 1 completed with GSS major status of GSS_S_CONTINUE_NEEDED. Refer to the description of the gss_init_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a 24-byte buffer containing the context handle returned by function 1 (input). The RACF userid associated with the thread that makes the request must be the context owner and the context must have been created on the local system.
- Address of a word containing the length of the context token (input)
- Address of the context token returned by the context acceptor (input)
- Address of a word which will contain the return flags (output). Specify a zero address if the return flags are not needed. Refer to the description of the gss_init_sec_context() function for the flag definitions.
- Address of a word which will contain the actual context expiration time in seconds (output). Specify a zero address if the context expiration time is not needed.

Accept a GSS-API security context (3)

This function will accept a GSS-API security context. The RACF userid associated with the thread that makes the request will be the owner of the security context and delegated credentials. The Kerberos principal associated with the RACF userid must be the same as the target principal specified by the context initiator. The output token must be returned to the context initiator if it has a non-zero length (a length of zero indicates no output token is needed). See the description of the gss_accept_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a word containing the length of the input context token (input)
- Address of the input context token received from the context initiator (input)

- Address of a 24-byte buffer which will contain the context handle for the new security context (output). The caller is responsible for deleting the security context when it is no longer needed.
- Address of the string block for the source principal name (output). The string buffer should be 240 bytes. The principal name will be returned in global format (/.../realm-name/principal-name). Specify a zero address of the source principal name is not needed.
- Address of a word which will contain the return flags (output). Specify a zero address if the return flags are not needed. Refer to the description of the gss_accept_sec_context() function for the flag definitions.
- Address of the context expiration time in seconds (output). Specify a zero address if the expiration time is not needed.
- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.
- Address of a 24-byte buffer which will contain the credential handle for the delegated credentials (output). The caller is responsible for deleting the credential when it is no longer needed. Specify a zero address if the delegated credentials are not needed. Delegated credentials are available only if the GSS_C_DELEG_FLAG flag is set in the return flags.

Delete a GSS-API security context (4)

This function will delete a GSS-API security context. The RACF userid associated with the thread that makes the request must be the security context owner and the security context must have been created on the local system. See the description of the gss_delete_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte context handle (input)

Release a GSS-API credential (5)

This function will release a GSS-API credential. The RACF userid associated with the thread that makes the request must be the credential owner and the credential must have been created on the local system. See the description of the gss_release_cred() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte credential handle (input)

Get the MIC for a message (6)

This function will generate the MIC (message integrity code) for a message. The RACF userid associated with the thread that makes the request must be the owner of the GSS-API security context and the context must have been created on the local system. See the description of the gss_get_mic() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte context handle (input)
- Address of a word containing the message length (input). The maximum message length is 65536.
- Address of the message (input)
- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.

Verify the MIC for a message (7)

This function will verify the MIC (message integrity code) for a message. The RACF userid associated with the thread that makes the request must be the owner of the GSS-API security context and the context must have been created on the local system. See the description of the gss_verify_mic() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte context handle (input)
- Address of a word containing the message length (input). The maximum message length is 65536.
- Address of the message (input)
- Address of a word containing the length of the input token (input)
- Address of the input token (input)

Wrap a message (8)

This function will sign and optionally encrypt a message. The RACF userid associated with the thread that makes the request must be the owner of the GSS-API security context and the context must have been created on the local system. See the description of the gss_wrap() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)

- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a 24-byte context handle (input)
- Address of a word containing the confidentiality request flag (input). Set the flag to 1 to request encryption or to 0 to request no encryption. A request for encryption will be ignored if the current system configuration does not support message encryption.
- Address of a word containing the message length (input). The maximum message length is 65536.
- Address of the message (input).
- Address of a word which will contain the confidentiality state (output). The state will be set to 1 if the message was encrypted and to 0 otherwise. Specify a zero address if the confidentiality state is not needed.
- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.

Unwrap a message (9)

This function will verify the signature and optionally decrypt a message. The RACF userid associated with the thread that makes the request must be the owner of the GSS-API security context and the context must have been created on the local system. See the description of the gss_unwrap() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte context handle (input)
- Address of a word containing the input token length (input)
- Address of the input token (input)
- Address of the buffer control block for the unwrapped message (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.
- Address of a word which will contain the confidentiality state (output). The state will be set to 1 if the message was encrypted and to 0 otherwise. Specify a zero address if the confidentiality state is not needed.

Export GSS-API security context (10)

This function will export a GSS-API security context. The security context will no longer be available upon completion of the export request. The RACF userid associated with the thread that makes the request must be the owner of

the GSS-API security context and the context must have been created on the local system. See the description of the gss_export_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte context handle (input)
- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.

Import GSS-API security context (11)

This function will import a GSS-API security context. The RACF userid associated with the thread that makes the request will be the owner of the new context. See the description of the gss_import_sec_context() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a word containing the length of the input token (input)
- Address of the input token (input)
- Address of a 24-byte buffer which will contain the context handle for the new security context (output). The caller should delete the security context when it is no longer needed.

Export GSS_API credential (12)

This function will export a GSS-API credential. The credential will still be available upon completion of the export request. The RACF userid associated with the thread that makes the request must be the owner of the GSS-API credential. The credential may have been created on any system in the sysplex. A credential can be exported only if it is an initiate credential (GSS_C_INITIATE was specified when the credential was created). See the description of the gss_export_cred() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of the 24-byte credential handle (input)

- Address of the buffer control block for the output token (output). The security server will obtain storage in the requested subpool for the return data. The caller must set the subpool number and the security server will set the length and address values in the buffer control block. The caller is responsible for releasing the return data when it is no longer needed.

Import GSS-API credential (13)

This function will import a GSS-API credential. The RACF userid associated with the thread that makes the request will be the owner of the new credential. See the description of the gss_import_cred() function for more information.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a word containing the length of the input token (input).
- Address of the input token (input)
- Address of a 24-byte buffer which will contain the credential handle for the new credential (output). The caller should release the credential when it is no longer needed.

Acquire GSS-API initiator credential (14)

This function will acquire a GSS-API credential which can be used to initiate a GSS-API security context. The RACF userid associated with the thread that makes the request will be the owner of the new credential. The Kerberos principal associated with the RACF userid will be used to obtain the initial ticket-granting ticket for the credential. This initial ticket will be forwardable and will not contain a client address list.

Function-specific parameters:

- Address of a word containing the subfunction code (input)
- Address of a word which will contain the GSS-API major status code (output)
- Address of a word which will contain the GSS-API minor status code (output)
- Address of ACEE to run under the authority of (input)
- Address of a word containing the requested credential expiration time in seconds (input). An expiration time of 0 will request the default expiration time of 2 hours while an expiration time of -1 will request the maximum expiration time. The actual credential expiration time will be limited by the lifetime of the Kerberos ticket-granting ticket.
- Address of a 24-byte buffer which will contain the credential handle for the new credential (output). The caller should release the credential when it is no longer needed.
- Address of a string block for the principal name (output). Specify a zero address if the principal name is not needed. The string buffer should be large enough for a 240-byte name. The principal name will be returned in global format (/.../realm-name/princ-name).
- Address of a word which will contain the actual credential expiration time in seconds (output). Specify a zero address if the credential expiration time is not needed.

PassTicket operation (Function code 3):

This function provides a similar service to R_ticketserv service function code 3. The major difference is that it can be invoked from AMODE 64.

Table 69. **Subfunction codes**

Value	Subfunction
1	Generate PassTicket
2	Evaluate PassTicket

Generate PassTicket(1): This function will generate a PassTicket for a specified userid and application name.

The function-specific parameters are:

- Address of a word containing the subfunction code (input).
- Address of a String block containing an 8-byte pre-allocated area to return the PassTicket (output). The string block length must be 8 or larger to indicate an acceptable buffer has been provided.
- Address of a String block containing 1-8 byte userid (input).
- Address of a String block containing a 1-8 byte application name (input).

Evaluate PassTicket(2): This function will evaluate a PassTicket for a specified userid and application name.

The function-specific parameters are:

- Address of a word containing the subfunction code (input).
- Address of a String block 8 byte PassTicket (input).
- Address of a String block containing 1-8 byte userid (input).
- Address of a String block containing a 1-8 byte application name (input).

Return and reason codes

R_GenSec may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	0	Invalid function code.
8	8	4	Parameter list error.
8	8	8	An internal error was encountered.
8	8	12	A recovery environment could not be established.
8	8	16	Not authorized to use this service.
8	12	8	Invocation of the Kerberos server program call (PC) interface failed with a 'parameter buffer overflow' return code. This indicates an internal error in IRRSKG00.

SAF return code	RACF return code	RACF reason code	Explanation
8	12	12	Invocation of the Kerberos server program call (PC) interface failed with an 'unable to allocate storage' return code. The region size of the Kerberos server-started task (SKRBKDC) should be increased.
8	12	16	Invocation of the Kerberos server program call (PC) interface failed with a 'local services are not available' return code. This indicates that the Kerberos server-started task (SKRBKDC) address space has not been started or is terminating.
8	12	20	Invocation of the Kerberos server program call (PC) interface failed with a 'abend in PC service routine' return code. The symptom record associated with this abend can be found in the logrec data set.
8	12	24	Invocation of the Kerberos server program call (PC) interface failed with an 'unable to obtain control lock' return code. This can occur if the task holding the lock is not being dispatched (for example, a dump is in progress).
8	16	28	PassTicket generation failure.
8	16	32	PassTicket evaluation failure. Possible reasons are: • PassTicket to be evaluated is not a successful PassTicket. • The PassTicket to be evaluated was already evaluated before and replay protection is in effect. • No PTKTDATA profile exists to match the specified application. • An internal error occurred.
8	16	X'nnnnnnnn'	The Kerberos server was not able to successfully process the function. X'nnnnnnnn' is the Network Authentication Service function status code. See Network Authentication Service documentation for more information.

Usage notes

1. This service is intended for use by z/OS application servers, which are not executing in a Language Environment®. It allows z/OS application servers to perform GSS-API functions.

   Note: Language Environment-enabled applications may also exploit this service, should they choose to do so.
2. This service requires that the z/OS Network Authentication Service server be installed and running. Otherwise, SAF return code 8, RACF return code 12, RACF reason code 16 will be returned to the invoker.

3. Separate ALETs must be specified for the SAF_return_code, RACF_return_code and RACF_reason_code parameters. Other parameters are not ALET qualified.
4. The ACEE parameter in the GSS-API subfunctions can only be specified by callers that are running in an authorized state. Problem state callers will receive a parameter error if the ACEE parameter has a non-zero pointer value.
5. All parameters after the return and reason codes must be in storage from the primary address space.
6. The service status codes are documented in *z/OS Integrated Security Services Network Authentication Service Administration*

Related services

R_ticketserv

R_getgroups (IRRSGG00): Get/Set supplemental groups

Function

The **R_getgroups** service checks the high-order bit of the input group count. See **Group_count** under "Parameters" on page 220 for more information.

The GIDs are not explicitly added to or deleted from the supplemental group list. A GID is in this list if the user was a member of the group when the user's ACEE was created through a RACROUTE REQUEST=VERIFY request and if the GID was assigned to the group before the **initUSP** service was performed for the process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSGG00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, User_key,
               ALET, Group_count,
               ALET, Grouplist,
               ALET, Number_of_GIDs
              )
```

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 words 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.

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.

User_key
The name of a byte containing the user's key. This key is used to store into the output grouplist area. The key is in the four high-order bits of the byte.

Group_count
The name of a word containing the number of GID entries that can be stored in the *Grouplist* area. IRRSGG00 uses the high-order bit to determine how to process the value in the parameters.

If the high-order bit of the input *Group_count* is:

1. On, the caller must store into this area the list of GIDs of the supplemental groups to be set as the supplemental groups of the current process.
2. Off, IRRSGG00 checks the input *Group_count* value. If it is:
   a. 0, the *Grouplist* area is not used. IRRSGG00 returns the total supplemental GID count of the current process in the *Number_of_GIDs* parameter.
   b. Less than the total supplemental GID count:
      1) An error code is returned.
      2) The GIDs of the supplemental groups for the current process are put into the *Grouplist* area, which can only accommodate the number of GIDs specified in the *Group_count* parameter.
      3) The count of the supplemental GIDs actually placed in the *Grouplist* area is returned in the *Number_of_GIDs* parameter.

4) The *Group_count* field is set to the total supplemental GID count of the current process.

   The supplemental groups in the *Grouplist* area are listed in the same order as the group connections shown in the output of the LISTUSER command.

c. Greater than or equal to the total supplemental GID count:

   1) The GIDs of the supplemental groups for the current process are put into the *Grouplist* area.

   2) The supplemental GID count of the current process is put into the *Number_of_GIDs* parameter.

Grouplist
The name of an area in which the GIDs of the supplemental groups for a process are returned. The *Group_count* parameter indicates the number of entries this area can contain. The GIDs are returned as consecutive 4-byte entries.

Number_of_GIDs
The name of a word in which the number of GIDs put in the *Grouplist* area is returned.

Return and reason codes

IRRSGE00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	*Group_count* is less than the number of supplemental groups (see item 2b on page 220 under the **Group_count** parameter).
8	8	8	The grouplist address is not valid.
8	8	12	An internal error occurred during RACF processing.

Usage note

- This service is intended only for use by the MVS BCP and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.

Related services

None

R_getgroupsbyname (IRRSUG00): Get groups by name

Function

The **R_getgroupsbyname** service checks the input group count and, if it is zero, returns the number of supplemental groups for the specified user ID. If the input count is not zero and it is less than the number of groups, an error code is

returned. If the count is not less than the number of groups, the GIDs of the supplemental groups for the specified user ID are put into the grouplist area, and the number of GIDs is returned.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

A RACF user can be connected to more than NGROUPS_MAX groups, but only up to the first NGROUPS_MAX z/OS UNIX groups will be associated with the user for this service.

The first NGROUPS_MAX z/OS UNIX groups to which a user is connected as shown by a LISTUSER command, are the groups that get associated with the user.

Format

```
CALL IRRSUG00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, User_key,
               ALET, Userid_length,
               ALET, Userid,
               ALET, Group_count,
               ALET, Grouplist,
               ALET, Number_of_GIDs
              )
```

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 words 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.

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.

User_key
The name of a byte containing the user's key. This key is used to store into the output grouplist area and number_of GIDs word. The key is in the four high-order bits of the byte.

Userid_length
The name of a byte containing the length of the user ID.

Userid
The name of an 8-byte area containing the user ID whose groups are to be returned. The user ID must be left-justified in the area. The userid_length parameter specifies the actual length of the name.

Group_count
The name of a fullword containing the number of GID entries in the input grouplist area.

Grouplist
The name of an area in which the GIDs of the supplemental groups are returned. The GIDs are returned as consecutive 4-byte entries. The group_count parameter indicates the number of entries this area can contain.

Number_of_GIDs
The name of a word in which the number of GIDs actually put in the grouplist area is returned.

Return and reason codes

IRRSUG00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The group count is less than the number of supplemental groups.
8	8	8	The grouplist address is not valid.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	20	RACROUTE VERIFY processing failed.
8	8	24	The user ID is not defined to RACF.

Usage notes

1. This service is intended only for use by the MVS BCP.

Related services

None

R_GetInfo (IRRSGI00): Get security server fields

Function

The R_GetInfo SAF callable service allows servers to retrieve a subset of security server information. Invokers provide a function code value to identify which subset of information is requested.

Requirements

Authorization:
Any PSW key in supervisor or problem state

Dispatchable unit mode:
Task

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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 Num_parms parameter must indicate the number of parameters in the parameter list.

RACF authorization

For callers not running in system key or supervisor state, the use of the R_GetInfo service is authorized by FACILITY class resources:

1. BPX.SERVER

   The ACEE associated with the address space is used to determine the caller. If the caller is running with a RACF user or group that has at least READ authority to the BPX.SERVER resource, use of R_GetInfo is permitted and no subsequent access checks are made.

2. IRR.RGETINFO.EIM

   Determining if function code X'0001' is found. The current TCB is checked for an ACEE. If one is found, it will be used to determine the caller. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to determine the caller. If the caller is running with a RACF user or group that has at least READ authority to the IRR.RGETINFO.EIM resource, use of R_GetInfo is permitted.

3. IRR.RGETINFO.REALM

   Determining if function code X'0002' is found. The current TCB is checked for an ACEE. If one is found, it will be used to determine the caller. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to determine the caller. If the caller is running with a RACF user or group that has at least READ authority to the IRR.RGETINFO.REALM resource, use of R_GetInfo is permitted.

If the FACILITY class is inactive, or the above resources are not defined, only servers running in system key or supervisor state may use the R_GetInfo service.

Format

```
CALL IRRSGI00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Num_parms,
               Parm_ALET
               Function_code
               Option,
               RACF_entity,
               RACF_class,
               Result_entries
             )
```

Parameters

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

ALET
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.

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.

Num_parms
The name of a fullword containing the number of parameters in the parameter list, including the Num_parms parameter. This number must be 14 for z/OS Version 1, Release 7 or later.

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

Function_code
The name of a half-word (2-byte) area containing the function code. Use the following values for z/OS Version 2, Release 7, or later:

X'0001' - Get Enterprise Identity Mapping (EIM) information
Get Enterprise Identity Mapping (EIM) information. This function is further defined by the **Option** parameter.

- If the **Option** parameter is X'0001', return the LDAPBIND profile name from the Enterprise Identity Mapping (EIM) segment of the specified **RACF_entity** value, where **RACF_entity** identifies a RACF userid.
- If the **Option** parameter is X'0002', return the local registry name, the Kerberos registry name, and the X.509 registry name from the Enterprise Identity Mapping (EIM) segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class.
- If the **Option** parameter is X'0003', return EIM and PROXY segment data:
  - Distinguished name of the Enterprise Identity Mapping (EIM) domain
  - Enterprise Identity Mapping (EIM) options
  - The LDAP host name
  - The distinguished name to use for LDAP binding
  - Whether or not a password for LDAP binding has been specified (YES or NO)

  for the specified **RACF_class** name and **RACF_entity** value, where **RACF_entity** identifies a RACF profile name.

Requested information will be returned in the **Result_entries** output area.

X'0002' - Retrieve REALM information
Return the value of the APPLDATA field from the profile in the REALM class with the profile name specified in the RACF_entity parameter.

- The **Option** parameter must be 1 when the function code is X'0002'.

Option
The name of a half-word (2-byte) area containing an option value for the specified function code.

See “Parameter usage” on page 229 for the function codes which the **Option** parameter applies. Valid option values and their effect on the specified function code are described above under the **Function_code** parameter.

RACF_entity
The name of a 247-byte area, which consists of a one-byte length field followed by up to 246 characters. This field is used to identify the RACF entity for information retrieval.

For **Function_code** X'0001', Get Enterprise Identity Mapping (EIM) information.

- When the Option parameter is X'0001', this field is a RACF user ID.
- When the Option parameter is X'0002', this field is the name of a profile in the REALM class.
- When the Option parameter is X'0003', this field is a RACF profile name.

RACF_class
The name of an 8-byte area containing a RACF class name. The class name is assumed to be :

- Left justified.
- Padded to the right with blanks.
- Specified in uppercase.
- A static IBM class name, a static installation-defined class name, or a dynamically defined installation class name.
- A general resource class. It cannot be a USER, GROUP, or DATASET.
- If function code is X'0002', this field must specify REALM padded with blanks to 8 bytes.

Result_entries
The name of an area for information to be retrieved. On input, the first 4 bytes of this area contain the length of the area. The format of the Result_entries structure is:

DEC OFFSET	HEX OFFSET	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	28	ResultArea	R_Getinfo result area
0	0	UNSIGNED	4	ResultAreaLen	Length of result area
4	4	UNSIGNED	4	ResultAreaUsed	Length of result area used
8	8	CHARACTER	20	*	Reserved
28	1C	CHARACTER	0	ResultAreaData	Function specific result data

When the **Function_code** is X'0001', Get Enterprise Identify Mapping (EIM) information and **Option** is X'0001', the function specific result data (ResultAreaData) will be mapped as follows:

DEC OFFSET	HEX OFFSET	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	249	ProfData	LDAPBIND profile result data
0	0	UNSIGNED	2	ProfLen	LDAPBIND profile length
2	2	CHARACTER	247	ProfName	LDAPBIND profile name

When the **Function_code** is X'0001', Get Enterprise Identify Mapping (EIM) information and **Option** is X'0002', the function specific result data (ResultAreaData) will be mapped as follows:

DEC OFFSET	HEX OFFSET	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	768	RegData	Registry name result data
0	0	CHARACTER	256	RegLocal	Local registry name
256	100	CHARACTER	256	RegKerb	Kerberos registry name
512	200	CHARACTER	256	RegX509	X.509 registry name

When the **Function_code** is X'0001', Get Enterprise Identify Mapping (EIM) information and **Option** is X'0003', the function specific result data (ResultAreaData) will be mapped as follows:

DEC OFFSET	HEX OFFSET	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	3205	SegData	Enterprise Identity Mapping (EIM) segment results data

0	0	UNSIGNED	4	SegOptions	OPTIONS field data:
					`X'55555555'(Enable)`
					`X'55555556'(Disable)`

4	4	CHARACTER	1024	SegDomainDN	DOMAINDN field data
1028	404	CHARACTER	1024	SegLDAPHost	LDAPHOST field data
2052	804	CHARACTER	1024	SegBINDDN	BINDDN field data
3076	C04	CHARACTER	129	SegBINDPW	BINDPW specified YES or NO

When the **Function_code** is X'0002', retrieve REALM information. the function specific result data (ResultAreaData) will be mapped as follows:

DEC OFFSET	HEX OFFSET	TYPE	LENGTH	NAME	DESCRIPTION
0	0	FIXED	4	RealmLen	Length of RealmName
4	4	CHARACTER	Variable up to 256	RealmName	Realm Name

Return and reason codes

IRRSGI00 may return the following values in the reason and return code parameters:

Table 70. Return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF not installed.
8	8	4	Internal error during RACF processing.
8	8	8	Recovery environment could not be established.
8	8	12	Not authorized to use this service.

Table 70. Return and reason codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	8	16	A RACF userid was not specified and this service was unable to determine one from either the task or address space level ACEE.
8	8	20	Entity not found, or when **Function_code** is X'0001' and **Option** is X'0002', the FACILITY class IRR.PROXY.DEFAULTS profile could not be found.
8	8	24	Reserved.
8	8	28	Enterprise Identity Mapping segment not found.
8	8	32	PROXY segment not found.
8	8	36	Extract failed. If function code is X'0002', the RACF return and reason codes from Extract will be placed in the ResultAreaData and a LOGREC will be created with the return and reason codes.
8	8	40	No data found in the APPLDATA field of the specified profile in the REALM class. No data returned.
8	12	8	The Num_parms parameter is in error.
8	12	10	The Function_code parameter is in error.
8	12	11	The Option parameter is in error.
8	12	12	The RACF_entity length is zero or exceeds the maximum allowed.
8	12	13	The CLASS parameter is invalid. For function x'0002', it must be specified as 'REALM ' (padded to 8 blanks.)
8	12	14	The length of the Result_entries structure is too small to return the requested information.

Parameter usage

Function	Function_code	Option	RACF_entity	RACF_class	Result_entries
Get Enterprise Identity Mapping (EIM) information	X'0001'	X'0001'	In	N/A	In/Out
		X'0002'	N/A	N/A	In/Out
		X'0003'	In	In	In/Out
Get Realm	X'0002'	X'0001'	In	N/A	In/Out

Usage notes

1. Function code X'0001' is provided specifically for use by the Enterprise Identity Mapping (EIM) component of the Integrated Security Services Element.
2. It is the responsibility of the caller to obtain and free the Result_entries data area. ResultAreaLen must be set to the total length of the area provided, including the length of the ResultArea structure and the length of the appropriate function specific data area.

On output, ResultAreaUsed will be set to the total length of the data returned. If ResultAreaLen is insufficient to return the requested information, an error will be returned and ResultAreaUsed will set to the total length required, if the size of the provided ResultArea structure is sufficient to contain this value.

3. When R_GetInfo detects a parameter list error, it will return SAF return code 8, RACF return code 12, and RACF reason code *nn*, where *nn* indicates the position in the parameter list of the parameter in error. For example, the **Num_parms** parameter is the eighth parameter in the parameter list, and if an invalid value is supplied for **Num_parms**, R_GetInfo will return SAF return code 8, RACF return code 12, and RACF reason code 8.
4. A length of zero can be specified for **RACF_entity** for **Function_code** X'0001' and **Option** X'0001'. In this case, RACF will attempt to determine the userid from the ACEE associated with the current TCB. If there is no ACEE associated with the current TCB, the address space level ACEE will be used. If no userid can be determined, an error will be returned.
5. When **Function_code** is X'0001', all CHARACTER values returned in the function specific result data areas for **Option** X'0001', X'0002', and X'0003' will be terminated by X'00'. For example, if the local registry name is "ABC", the first four bytes of the RegLocal field will contain X'C1C2C300'.
6. When **Function_code** is X'0002', all CHARACTER values returned in the function specific result data areas will be terminated by X'00'. The ResultAreaUsed will reflect this extra '00'x byte.

R_IPC_ctl (IRRSCI00): Perform IPC control

Function

The **R_IPC_ctl** service performs a function based on the function code value in the parameter list.

- When the function is Check Owner for Remove ID, **R_IPC_ctl** checks whether the current process has the appropriate authority to the IISP.
- When the function is Check Owner and Set, **R_IPC_ctl** sets the owner's UID and GID and the mode permission bits if the current process has the appropriate authority.
- When the function is Check Superuser and Set, **R_IPC_ctl** sets the owner's UID and GID and the mode permission bits if the current process is a superuser.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user/any task if system user type is specified

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. The access checks performed are defined in XPG4 System Interfaces and Headers under msgctl, semctl, and shmctl interfaces for commands IPC_SET and the IPC_RMID. The access checks are as follows:
   a. The effective z/OS UNIX user identifier (UID) for the calling process is used for all access checks.
   b. If the CREI user type is system, IRRSCI00 grants authorization when the function is Check Owner for Remove ID, or updates the IPCP when the function is either Check Owner and Set or Check Superuser and Set.
   c. The user is considered a superuser if the effective UID is zero or if the ACEE indicates trusted or privileged authority.
   d. If the function is Check Owner for Remove ID, the user must be either a superuser or the effective UID of the process must match either the owner's UID or creator's UID in the IISP for a successful completion. Otherwise, the user is not authorized.

      Note: If the caller is unauthorized as stated above, an authorization check is performed on the resource name in the UNIXPRIV class indicated in Table 71. If the authorization check is successful, the caller is treated as a superuser.

Table 71. UNIXPRIV class resource names used in R_IPC_ctl

Function code	Resource name	Access required
1-Check Owner for Remove ID	SUPERUSER.IPC.RMID	READ

2. If the function is Check Superuser and Set, the user must be a superuser in order to set the owner's z/OS UNIX user identifier (UID), owner's z/OS UNIX group identifier (GID), and mode fields from the input parameters into the IISP for a successful completion. Otherwise, the user is not authorized.
3. If the function is Check Owner and Set, the user must be either a superuser or the effective UID of the process must match either the owner's UID or creator's UID in the IISP in order to set the owner's UID, owner's GID, and mode fields from the input parameters into the IISP for a successful completion. Otherwise, the user is not authorized.
4. If the SECLABEL class is active and the ISP contains a security label, the accessing process must have an equivalent security label. With MLIPCOBJ active, requests will be failed if either the accessing process or the ISP does not contain a security label.

Format

```
CALL IRRSCI00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Function_code,
               ALET, Owner_UID,
               ALET, Owner_GID,
               ALET, Mode_Permissions,
               ALET, ISP,
               ALET, CREDIPC
              )
```

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 words 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.

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.

Function_code
The name of a 1-byte area containing the function code:

X'01' Check Owner for Remove ID

X'02' Check Owner and Set

X'03' Check Superuser and Set

Owner_UID
The name of a 4-byte area containing the new owner's UID to be set.

Owner_GID
The name of a 4-byte area containing the new owner's GID to be set.

Mode_Permissions
The name of a 4-byte area containing the new mode permission bits to be set. The following is a list of defined permission bits mapped by BPXYMODE:

S_IRUSR
Permits the process that owns the IPC member to read it.

S_IWUSR
Permits the process that owns the IPC member to alter it.

S_IRGRP
Permits the group associated with the IPC member to read it.

S_IWGRP
Permits the group associated with the IPC member to alter it.

S_IROTH
Permits others to read the IPC member.

S_IWOTH
Permits others to alter the IPC member.

Alter and write have the same meaning for access checks. Alter applies to semaphores, and write applies to message queueing and shared memory segments.

ISP
The name of the IISP for the file being accessed.

CREDIPC
The name of the CREI structure for the current IPC system callable service. The CREI contains the IPC identifier and the IPC key. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSCI00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized.
8	8	12	An internal error occurred during RACF processing.
8	8	32	The CREI user type is not supported.

Usage notes

1. This service is intended for use only by the MVS BCP.
2. An audit record is optionally written, depending on the audit options in effect for the system.
3. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

makeISP, ck_IPC_access

R_kerbinfo (IRRSMK00): Retrieve or set security server network authentication service fields

Function

The **R_kerbinfo** callable service can be used to either retrieve RACF Network Authentication Service information, or to update the count of unsuccessful attempts to use a Network Authentication Service key.

The action performed by this callable service is based on the function code passed by the caller in the R_kerbinfo parameter list:

- When the function code is set to X'01', R_kerbinfo retrieves local Network Authentication Service principal information. The caller must identify the principal by providing a RACF name or a Network Authentication Service principal name.
- When the function code is set to X'02', R_kerbinfo increments the count of invalid attempts by a Network Authentication Service principal to use a key. The caller must identify the principal by providing a Network Authentication Service principal name.
- When the function code is set to X'03', R_kerbinfo resets the count of invalid attempts by a Network Authentication Service principal to use a key to zero. The caller must identify the principal by providing a Network Authentication Service principal name.
- When the function code is set to X'04', R_kerbinfo retrieves Network Authentication Service realm information. The caller may identify the realm by providing a Network Authentication Service realm profile name, or by providing a NULL name, in which case RACF will return information about the local realm, KERBDFLT.

Field data is returned to the invoker in a data structure containing a set of repeating ordered triplets, which are of the format name of the field, length of the field, and field data.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Ant task

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary only

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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 one in the high-order (sign) bit.

Format

```
CALL IRRSMK00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Function_code,
               RACF_name,
               KERB_name,
               Data_area,
              )
```

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 words 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.

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.

Function_code
The name of a 1-byte area in the primary address space containing the function code:

X'01'
Retrieve local Network Authentication Service principal information.

X'02'
Increment a Network Authentication Service principal's count of invalid key attempts.

RACF will process this request much the same as it does when an invalid password is supplied during TSO logon. When the number of attempts exceeds the number of incorrect password attempts which RACF allows, set using the SETROPTS command PASSWORD REVOKE suboperand, the RACF user ID will be revoked, an ICH408I message will be issued and an SMF Type 80 record will be written.

X'03'
Reset a Network Authentication Service principal's count of invalid key attempts to zero.

RACF will process this request much the same as a successful TSO logon request and will update the date and time of the last successful logon (the LJDATE/LJTIME fields in the RACF user profile) to the current date and time.

X'04'
Retrieve Network Authentication Service realm information.

RACF_name
The name of a 9-byte area in the primary address space consisting of a 1-byte length field followed by up to 8 characters. If a value is specified for RACF_name, it must be defined RACF user ID and specified in uppercase. If a RACF user ID is not specified, the length must equal zero.

RACF_name may only be specified with function code X'01' and will be used to identify a local Network Authentication Service principal by the principal's RACF user identity.

KERB_name
The name of an area in the primary address space containing the 240-byte Network Authentication Service name. The name must be left-justified and padded with blanks. The only way to get the local realm information is to specify a null in the KERB_name field. If a Network Authentication Service realm profile name is specified, it must be realm qualified (for example, follow the DCE-like convention of /.../REALM_A/KRBTGT/REALM_B) and must be folded to all uppercase.

Note: Local Network Authentication Service principal names and realm names returned in the Data_area NAME field will not be realm qualified and will be case sensitive.

If the caller does not want to specify a KERB_name, then the first character of the area must be a NULL byte (that is, the first byte of the 240 byte area must contain X'00').

KERB_name may be specified with any function code. For function codes X'01,',X'02', and X'03', it is used to identify a local Network Authentication Service principal. For function X'04', it is used to identify a Network Authentication Service realm profile name.

Data_area
The name of an area in the primary address space for fields to be retrieved. The format of the Data_area structure is:

Offset	*Length*	*Description*
0	2	The length in bytes of the entire Data_area structure
2	2	Total number of fields
4	8	The name of the field
12	2	The length of the field data
14	variable	Field data

The ordered triplet (name of the field, length of the field, and field data) is a repeating data structure. This triplet will repeat for the total number of fields in the input Data_area.

The following table lists the fields that will be returned for function code X'01' (retrieve local principal information) and X'04' (retrieve realm information). The caller-supplied Data_area structure must be allocated large enough for all of the fields associated with the function to be returned. The fields that will be returned, along with each field's maximum length in bytes and the order that they will be returned, can be found in the following table:

Field name	**Maximum length**	**Data type**	**Description**
USERID	8	Character	RACF userid (N/A for function X'04')

Field name	Maximum length	Data type	Description
REVOKED	1	Boolean	Flag indicating that user has been revoked (N/A for function X'04')
EXPIRED	1	Boolean	Flag indicating that user's key has expired (N/A for function X'04')
NAME	240	Character	Kerberos name
MINTKTLF	4	Integer	Minimum ticket life value (N/A for function X'01')
MAXTKTLF	4	Integer	Maximum ticket life value
DEFTKTLF	4	Integer	Default ticket life value (N/A for function X'01')
SALT	240	Character	Current key salt
ENCTYPE	4	Binary	Specifies the encryption types allowed for this profile. See Table 72 on page 238 for ENCTYPE values.
CURKEYV	1	Integer	Current key version (1-255)
CURKEY	98	Character	Current key. See format in the following table.
PREVKEYV	1	Integer	Previous key version (1-255)
PREVKEY	98	Character	Previous key. See format in the following table.
CHKADDRS	1	Boolean	Flag indicating that the Kerberos server should check addresses in tickets.

The minimum length of the Data_area structure was previously 838 bytes. If fewer than 849 bytes are supplied the service will behave as before the new support. If the length is greater than or equal to 849 bytes the new triplet will be returned along with the previous data. This accounts for the two 2-byte length fields at the beginning and 14 sets of field triplets.

The value fields CURKEY and PREVKEY returned for information retrieval, function codes X'01' and X'04', have the following format:

Offset	*Length*	*Description*
0	2	The length of the DES key
2	8	DES key
10	2	The length of the DES3 key
12	24	DES3 key
36	2	The length of the DES key with derivation
38	8	DES key with derivation
46	2	The length of the AES128 key
48	16	AES128 key
64	2	The length of the AES256 key
66	32	AES256 key

After a password change, profiles contain the new AES key values only. Therefore, you might see the following profiles:

- A profile where neither CURKEY or PREVKEY contains the new AES key values. This occurs when the profile was defined before z/OS V1R9 and no password change has occurred after migrating to z/OS V1R9.
- A profile that has CURKEY with AES key values and PREVKEY with no AES key values. This occurs when the profile was defined before z/OS V1R9 and one password change has occurred after migrating to z/OS V1R9.

The value of the ENCTYPE field is defined as follows:

Table 72. ENCTYPE field value

Value of ENCTYPE field (by bit position)	Encryption type allowed
'00000000'X	None
'000000001'X	DES
'000000002'X	DES3
'000000004'X	DES with derivation
'000000008'X	AES128
'000000010'X	AES256

All keys will be returned regardless of the setting of ENCTYPE.

Return and reason codes

IRRSMK00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
4	4	0	User revoked prior to call.
4	4	4	User revoked by this call.
8	8	0	Invalid function code.
8	8	4	Parameter list error.
8	8	8	Internal error during RACF processing.
8	8	12	Recovery environment could not be established.
8	8	20	RACF profile does not have required segments.
8	8	24	Length of the output area is too small to contain the data retrieved.
8	8	32	RACF profile specified does not exist.
8	8	36	Mapping to RACF profile failed.

Usage notes

1. The caller is in supervisor state, so the task and address space ACEEs are not checked. Therefore, for example, an authorized caller may extract KERB segment fields, or update the invalid key count, for any user who has a KERB segment.

2. This service returns fields that have been previously populated. Associated with the returned fields is a length indicator. The length indicator is set to zero if a field does not exist.
3. If RACF_name and KERB_name are both provided for function X'01', R_kerbinfo will use RACF_name.
4. If RACF_name is provided for any function other than X'01', a parameter list error will be returned.
5. If KERB_name is not supplied on a function X'04' request (the first character is NULL), information about the local z/OS Kerberos Security Server, KERBDFLT, will be returned. Alternatively, KERB_name may be explicitly set to KERBDFLT.
6. It is the responsibility of the caller to obtain and free the Data_area. If the fields to be retrieved from RACF are larger than the Data_area, RACF fails the request and returns an error. If the size of the Data_area structure is sufficient to contain this value, the Data_area length (offset 0) is set to the total length required.
7. Field level access checking does not occur when retrieving fields with this service.
8. Field names are returned as 8-character fields, left-justified, and padded with blanks. They are specified in uppercase.
9. Fields that are not applicable for a function code, such as USERID for function code X'04', will be returned with the length set to zero.
10. If function code X'02' causes a user to be revoked, an ICH408I message will be issued and an SMF Type 80 record will be cut.
11. If the length of data_area structure specified is less than the minimum length defined in the parameter description a subset of the data will be returned unless it is less than 838 bytes which will result in return and reason codes 8,8,24.

Parameter usage

Table 73. Parameter usage

Parameter	Function X'01'	Function X'02'	Function X'03'	Function X'04'
SAF_return_code	Output	Output	Output	Output
RACF_return_code	Output	Output	Output	Output
RACF_reason_code	Output	Output	Output	Output
Function_code	Input	Input	Input	Input
KERB_name	Input	Input	Input	Input
Data_area	Input/Output	N/A	N/A	Input/Output

Related services

R_ticketserv, R_usermap

R_PgmSignVer (IRRSPS00): Program Sign and Verify

Function

The R_PgmSignVer service provides the functions required to apply a digital signature to a z/OS program object, and the functions required to verify such a

signature. The signing services are intended for use by the z/OS program binder. The verification services are intended for use by the z/OS loader.

The signing services consist of the following functions:

1. Initialize signing - Allocates and initializes a work area to perform message digestion (hash) against the program's data, and reads the digital certificates from the program signing key ring.
2. Digest intermediate program data - Hashes a portion of the program's data for signing.
3. Generate signature - Hashes the final portion of the program's data, if provided, and generates the digital signature by encrypting the calculated hash using the private key from the default certificate in the key ring. Any resources obtained by the initialize signing function are freed before returning.
4. Cleanup - Frees resources obtained by the initialize signing function. To be called if the signature generation is not completed by the generate signature function (for example, for recovery cleanup).

The verification services consist of the following functions:

1. Initialize signature-verification - Allocates and initializes a work area to perform message digestion (hash) against the program's data, and hashes any initial program data that is supplied.
2. Digest intermediate program data - Hashes a portion of the program's data for verification.
3. Final verification - Hashes the final portion of the program's data, if provided. The signature provided with the program is then decrypted with the public key from the end-entity certificate that accompanies the signature, and the two hash values are compared to verify the signature. Any resources obtained by the initialize signature-verification function are freed before returning.
4. Cleanup - Frees resources obtained by the initialize signature-verification function. To be called if that signature-verification is not completed by calling the final verification function (for example, for recovery cleanup)
5. Interrogate directive - Generate appropriate return code and perform auditing according to security settings when a program signature cannot be verified.

Requirements

Authorization:
Any PSW key in supervisor or problem state.

Note: In the following documentation, a caller that is in either supervisor state or a system key is referred to as "authorized". Otherwise, the caller is referred to as "unauthorized".

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31 or 64

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have a FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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. The Num_parms parameter must be in the primary address space.

Linkage conventions

Callers in 31-bit addressing mode should link-edit the IRRSPS00 stub module with their code, and use the IRRPCOMP mapping macro. Callers in 64-bit addressing mode should link-edit the IRRSPS64 stub module with their code, and use the IRRPCOMY mapping macro.

RACF authorization

For unauthorized callers of the program signing services, the caller must have sufficient authority to use the key ring specified in the parameter list (or if not specified, then as defined in the appropriate profile in the FACILITY class) and the private key contained within it as determined by the R_datalib callable service and ICSF. See "R_datalib (IRRSDL00 or IRRSDL64): OCSF data library" on page 158 for more information.

For the signature-verification services, there are no authorization requirements, regardless of the caller's state.

Format

```
CALL IRRSPS00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Num_parms,
               Function_code,
               Function_parmlist
              )
```

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 word containing the ALET for the following parameter. Each parameter must have an ALET specified. Each ALET must be 0 for this service. The words 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.

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.

Num_parms
Specifies the name of a fullword that contains the total number of parameters in the parameter list. The contents of this field must be set to binary ten.

Function_code
The name of a 2-byte area containing the Function code. The function code has one of the following values:

X'0001'
Initialize signing. (Function name SIGINIT.) This function must be called before calling any of the other signing functions.

X'0002'
Digest intermediate program data for signature generation. (Function name SIGUPDAT.) This function is optional. It should be called only if all the program's data cannot be processed on one call to generate signature. It may be called multiple times before calling generate signature.

X'0003'
Generate signature. (Function name SIGFINAL.) This function finalizes the signature generation and returns the result. It also frees any work area storage that may have been allocated.

X'0004'
Terminates the signing operation and frees resources allocated by SIGINIT. (Function name SIGCLEAN.) This function should be called only if signature generation is not to be finalized with a call to SIGFINAL. Note that all R_PgmSignVer functions will perform this cleanup if they return an error to the caller. The caller needs to call the cleanup function only if it is terminating for its own reason.

X'0005'
Initialize signature-verification and optionally digest initial program data. (Function name VERINIT.) This function must be called before calling any of the other verification functions except VERINTER (interrogate directive).

X'0006'
Digest intermediate program data for signature-verification. (Function name VERUPDAT.) This function is optional. It should be called only if all the program's data cannot be processed on the VERINIT and VERFINAL calls. It may be called multiple times before performing final verification.

X'0007'
Perform final verification. (Function name VERFINAL.) This function finalizes the signature-verification and returns the result. It also audits the event and frees any work area storage that may have been allocated. If all the program data can be specified in a single call, then VERFINAL can be called without first calling VERINIT. See Table 80 on page 246 for more information.

X'0008'
Terminates the signing operation and frees resources allocated by

VERINIT. (Function name VERCLEAN.) This function should be called only if signature generation is not to be finalized with a call to VERFINAL. Note that all R_PgmSignVer functions will perform this cleanup if they return an error to the caller. The caller only needs to call the cleanup function if it is terminating for its own reason.

X'0009'
Interrogate directive. (Function name VERINTER.) This function examines the directive (supplied within the ICHSFENT in the function-specific parameter list) to determine the appropriate action. This would be used for the cases where VERFINAL will not be called. For example, when digital signature processing is required but the module does not have a digital signature. This function is not available to unauthorized callers.

Function_parmlist
Specifies the name of the function code specific parameter list area for the Function_code specified.

All address fields are 8-byte addresses. When referring to 31-bit storage addresses, the caller must make sure that the high-order word of the address field is set to binary zeros.

Table 74. Function_parmlist for SIGINIT

Field	Attributes	Usage	Description
PGSN_SI_PLIST	Structure	In	Function-specific parameter list for signing initialization.
PGSN_SI_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'SIGINIT '.
PGSN_SI_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_SI_PGM_NAME_LEN	4 byte numeric	In	Length of the name of the program being signed. The length must not exceed 8 characters.
PGSN_SI_PGM_NAME@	Address of	In	Address of the name of the program being signed. **Note:** This parameter is used to derive the name/token that is used for subsequent calls. As such, it does not necessarily need to be the program name, but must be a unique value which does not result in a name collision with other signing operations.
PGSN_SI_KEYRING_NAME@	Address of	In	Address of the name of the SAF key ring that contains the certificates to be used for signing. The address is meaningful only if PGSN_SI_KEYRING_LEN is a non-zero value. The name that this address points to has the following syntax: *owning-userid/ring-name* The owning-userid (but not the slash) may be omitted if the key ring is owned by the user ID associated with the calling application.
PGSN_SI_KEYRING_LEN	4 byte numeric	In	Length of the name of the SAF key ring that contains the certificates to be used for signing. Set this field to binary zero to have the security manager determine the key ring to use.

Table 74. Function_parmlist for SIGINIT (continued)

Field	Attributes	Usage	Description
PGSN_SI_SIGINFO_LEN	4 byte numeric	Out	Length of the ZOSSignatureInfo structure which will be returned as part of the signature area structure in the SIGFINAL call.
PGSN_SI_DIGEST_ALG	1 byte numeric	In	Numeric value indicating what message digest algorithm to use for the signing. Set this field to binary zero to have the security manager determine the algorithm to use. A value of 1 indicates that SHA256 is to be used.

Table 75. Function_parmlist for SIGUPDAT

Field	Attributes	Usage	Description
PGSN_SU_PLIST	Structure	In	Function-specific parameter list for intermediate signing.
PGSN_SU_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'SIGUPDAT'.
PGSN_SU_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_SU_PGM_NAME_LEN	4 byte numeric	In	Length of the name of the program being signed. The length must not exceed 8 characters.
PGSN_SU_PGM_NAME@	Address of	In	Address of the name of the program being signed. Must be the same as the value supplied on the SIGINIT call.
PGSN_SU_PGM_DATA@	Address of	In	Address of a structure specifying the intermediate range(s) of data to sign. The structure is mapped by PGSN_DATA_RANGE. See usage note 7 on page 255 in "Usage notes for program verification" on page 254 for the format of this structure.

Table 76. Function_parmlist for SIGFINAL

Field	Attributes	Usage	Description
PGSN_SF_PLIST	Structure	In	Function-specific parameter list for final signing.
PGSN_SF_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'SIGFINAL'.
PGSN_SF_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_SF_PGM_NAME_LEN	4 byte numeric	In	Length of the name of the program being signed. The length must not exceed 8 characters.
PGSN_SF_PGM_NAME@	Address of	In	Address of the name of the program being signed. Must be the same as the value supplied on the SIGINIT call.
PGSN_SF_PGM_DATA@	Address of	In	Address of a structure specifying the final range(s) of data to sign. The structure is mapped by PGSN_DATA_RANGE. See usage note 7 on page 255 in "Usage notes for program verification" on page 254 for the format of this structure.
PGSN_SF_SIG_AREA@	Address of	Out	Address of the allocated signature area structure. See usage note 6 on page 253 in "Usage notes for program signing" on page 253 for the format of the area.

Table 76. Function_parmlist for SIGFINAL (continued)

Field	Attributes	Usage	Description
PGSN_SF_SUBPOOL	1 byte numeric	In	Subpool to be used for allocation of the signature data structure. For unauthorized callers, this must be a value in the range 1 – 127.

Table 77. Function_parmlist for SIGCLEAN

Field	Attributes	Usage	Description
PGSN_SC_PLIST	Structure	In	Function-specific parameter list for signing cleanup.
PGSN_SC_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'SIGCLEAN'.
PGSN_SC_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_SC_PGM_NAME_LEN	4 byte numeric	In	Length of the name of the program being signed. The length must not exceed 8 characters.
PGSN_SC_PGM_NAME@	Address of	In	Address of the name of the program being signed. Must be the same as the value supplied on the SIGINIT call.

Table 78. Function_parmlist for VERINIT

Field	Attributes	Usage	Description
PGSN_VI_PLIST	Structure	In	Function-specific parameter list for verification initialization.
PGSN_VI_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'VERINIT '.
PGSN_VI_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_VI_PGM_NAME_LEN	4 byte numeric	In	For unauthorized callers, length of the name of the program being verified. The length must not exceed 8 characters. Ignored for authorized callers.
PGSN_VI_PGM_NAME@	Address of	In	For unauthorized callers, address of the name of the program being verified. Ignored for authorized callers.
PGSN_VI_CONTEXT@	Address of	Out	For authorized callers, address of the allocated verify context that the caller should pass in to subsequent verification calls. Ignored for unauthorized callers.
PGSN_VI_PGM_DATA@	Address of	In	Address of a structure specifying the initial range(s) of data to verify. The structure is mapped by PGSN_DATA_RANGE. See usage note 7 on page 255 in "Usage notes for program verification" on page 254 for the format of this structure.
PGSN_VI_SIGINFO@	Address of	In	Address of the ZOSSignatureInfo structure extracted from the program object being verified.
PGSN_VI_SIGINFO_LEN	4 byte numeric	In	Length of the ZOSSignatureInfo structure extracted from the program object being verified.
PGSN_VI_DIGEST_ALG	1 byte numeric	In	Numeric value indicating what message digest algorithm to use for the verification. A value of 0 means the value contained in the ZOSSignatureInfo structure should be used. This is the only supported value.

Table 79. Function_parmlist for VERUPDAT

Field	Attributes	Usage	Description
PGSN_VU_PLIST	Structure	In	Function-specific parameter list for intermediate verification.
PGSN_VU_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'VERUPDAT'.
PGSN_VU_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_VU_PGM_NAME_LEN	4 byte numeric	In	For unauthorized callers, length of the name of the program being verified. The length must not exceed 8 characters. Ignored for authorized callers.
PGSN_VU_PGM_NAME@	Address of	In	For unauthorized callers, address of the name of the program being verified. Must be the same as the value supplied on the VERINIT call. Ignored for authorized callers.
PGSN_VU_CONTEXT@	Address of	In	For authorized callers, address of the verify context area allocated on the VERINIT call. Ignored for unauthorized callers.
PGSN_VU_PGM_DATA@	Address of	In	Address of a structure specifying the intermediate range(s) of data to verify. The structure is mapped by PGSN_DATA_RANGE. See usage note 7 on page 255 in "Usage notes for program verification" on page 254 for the format of this structure.

Table 80. Function_parmlist for VERFINAL

Field	Attributes	Usage	Description
PGSN_VF_PLIST	Structure	In	Function-specific parameter list for final verification.
PGSN_VF_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'VERFINAL'.
PGSN_VF_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_VF_PGM_NAME_LEN	4 byte numeric	In	For unauthorized callers, length of the name of the program being verified. The length must not exceed 8 characters. Ignored for authorized callers. If the length is zero, it is assumed that no VERINIT call was made, and the signature is generated based on the data supplied in this call, using the default digest algorithm.
PGSN_VF_PGM_NAME@	Address of	In	For unauthorized callers, address of the name of the program being verified. Must be the same as the value supplied on the VERINIT call. Ignored for authorized callers.
PGSN_VF_CONTEXT@	Address of	In	For authorized callers, address of the verify context area allocated on the VERINIT call. Ignored for unauthorized callers. If the address is zero, it is assumed that no VERINIT call was made, and the signature is generated based on the data supplied in this call, using the default digest algorithm.

Table 80. Function_parmlist for VERFINAL (continued)

Field	Attributes	Usage	Description
PGSN_VF_PGM_DATA@	Address of	In	Address of a structure specifying the final range(s) of data to verify. The structure is mapped by PGSN_DATA_RANGE. See usage note 7 on page 255 in "Usage notes for program verification" on page 254 for the format of this structure.
PGSN_VF_LOGSTRING@	Address of	In	Address of an area that consists of a 1 byte length field followed by character data (up to 255 bytes) to be included in any audit records that are created. If the address or the length byte is 0, this parameter is ignored.
PGSN_VF_ICHSFENT@	Address of	In	For authorized callers, address of the FASTAUTH entity parameter mapping containing the directive (previously retrieved from RACF by Contents Supervision). This parameter is optional. See usage notes 6 on page 255 and 16 on page 256 in "Usage notes for program verification" on page 254. Ignored for unauthorized callers.
PGSN_VF_SIGINFO@	Address of	In	Address of the ZOSSignatureInfo structure extracted from the program object being verified. This field is required if VERFINAL is the only call being made. It is ignored if it was already passed to VERINIT.
PGSN_VF_SIGINFO_LEN	4 byte numeric	In	Length of the ZOSSignatureInfo structure extracted from the program object being verified. This field is required if VERFINAL is the only call being made. It is ignored if it was already passed to VERINIT.

Table 81. Function_parmlist for VERCLEAN

Field	Attributes	Usage	Description
PGSN_VC_PLIST	Structure	In	Function-specific parameter list for verification cleanup.
PGSN_VC_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'VERCLEAN'.
PGSN_VC_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
PGSN_VC_PGM_NAME_LEN	4 byte numeric	In	For unauthorized callers, length of the name of the program being verified. The length must not exceed 8 characters. Ignored for authorized callers.
PGSN_VC_PGM_NAME@	Address of	In	For unauthorized callers, address of the name of the program being verified. Must be the same as the value supplied on the VERINIT call. Ignored for authorized callers.
PGSN_VC_CONTEXT@	Address of	In	For authorized callers, address of the verify context area allocated on the VERINIT call. Ignored for unauthorized callers.

Table 82. Function_parmlist for VERINTER

Field	Attributes	Usage	Description
PGSN_ID_PLIST	Structure	In	Function-specific parameter list for interrogating the directive.
PGSN_ID_EYE	8 characters	In	Eyecatcher, 8 characters. Actual value must be set by invoker: 'VERINTER'.

Table 82. Function_parmlist for VERINTER (continued)

Field	Attributes	Usage	Description
PGSN_ID_VERS	4 byte numeric	In	The version number for this function-specific parameter list. The contents of this field must be set to binary zero.
*	4 characters	In	Reserved
PGSN_ID_ ICHSFENT@	Address of	In	For authorized callers, address of the FASTAUTH entity parameter mapping (previously retrieved from RACF by Contents Supervision). Ignored for unauthorized callers.
PGSN_ID_LOGSTRING@	Address of	In	Address of an area that consists of a 1 byte length field followed by character data (up to 255 bytes) to be included in any audit records that are created. If the address or the length byte is 0, this parameter is ignored.
PGSN_ID_EVENT	1 byte numeric	In	Constant indicating what sigver event was detected: • x'01' – Digital signature processing is required but the module does not have a digital signature. • x'02' – Digital signature processing is required. The PDSE directory entry for the module indicates it's signed but the digital signature is missing.

Return and reason codes

R_PgmSignVer may return the following values in the return and reason code parameters:

Table 83. Return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Successful completion
4	0	0	RACF not installed
8	8	4	An internal error has occurred during RACF processing of the requested function.
8	8	8	Unable to establish a recovery environment.
8	8	12	Function not available for unauthorized callers.
8	100	xx	A parameter list error has been detected. The RACF reason code identifies the parameter in error. The reason code is the offset of the parameter in error, relative to the start of COMP or COMY.
8	104	yy	A function-specific parameter list (pointed to by the *Function_parmlist* parameter) error has been detected. The RACF reason code identifies the field in error. The reason code is the offset of the field in error, relative to the start of the function-specific parameter list. When the field is an address, the error may pertain to the address itself, or to something to which it points.

In addition to the above, R_PgmSignVer may return function specific return and reason codes:

Table 84. SIGINIT specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Signature operation is already in progress for the specified program name.
8	8	104	Security Manager is unable to determine the key ring to use.
8	8	108	Syntax error in supplied key ring name, or in the key ring name contained within the APPLDATA of the RACF FACILITY class profile.
8	8	112	Key ring does not exist or does not contain a default certificate.
8	8	116	Caller not authorized to use R_datalib to access the key ring .
8	8	120	Certificate chain in the key ring is incomplete.
8	8	124	Certificate chain contains more than 10 certificates, or key ring contains more than 50 certificates. (Some of these might not constitute part of the trust chain. However, you should not connect any certificates that do not.)
8	8	128	CA certificate in the key ring does not have certificate signing capability. (KeyUsage extension present but keyCertSign flag is off or BasicConstraints extension is present but cA flag is off.)
8	8	132	Default certificate in key ring does not have a private key.
8	8	136	Default certificate in key ring does not have code signing capability. (KeyUsage extension present but digitalSignature or nonRepudiation flag is off.)
8	8	140	The certificate signature algorithm of one or more certificates in the key ring is not supported.
8	8	144	The key type of one or more certificates in the key ring is not supported. This reason code will also be issued if the private key of the signing certificate is stored in ICSF.
8	8	148	The specified message digest algorithm not supported.
8	8	152	CA or signing certificate is expired or not yet active.
8	12	xx	Unexpected error returned from R_datalib. RACF reason code is the DataGetFirst/DataGetNext reason code returned by R_datalib.
8	16	xx	Unexpected error returned from IEANTCR. RACF reason code is the return code returned by IEANTCR.
8	20	0x00xxyyyy	An unexpected error is returned from ICSF. The hexadecimal reason code value is formatted as follows: **xx** ICSF return code. **yyyy** ICSF reason code.

Table 85. SIGUPDAT specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Signature operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	2nn	xx	Unexpected error from the cryptographic module. The return code is 200+*nn* where *nn* identifies the function being performed. The reason code is the return code from the cryptographic module. This information should be reported to IBM service.

Table 86. SIGFINAL specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Signature operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	16	xx	Unexpected error returned from IEANTDL. RACF reason code is the return code returned by IEANTDL.
8	2nn	xx	Unexpected error from the cryptographic module. The return code is 200+*nn* where *nn* identifies the function being performed. The reason code is the return code from the cryptographic module. This information should be reported to IBM service.

Table 87. SIGCLEAN specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Signature operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	16	xx	Unexpected error returned from IEANTDL. RACF reason code is the return code returned by IEANTDL.

Table 88. VERINIT specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Verification operation is already in progress for the specified program name.
8	12	xx	Unexpected error returned from IEANTCR. RACF reason code is the return code returned by IEANTCR.
8	16	116	The program verification module (IRRPVERS) is not loaded. See *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF System Programmer's Guide* for information about configuring and loading the verification module with the IRRVERLD program.

Table 88. VERINIT specific return and reason codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
8	2nn	xx	Unexpected error from the cryptographic module. The return code is 200+*nn* where *nn* identifies the function being performed. The reason code is the return code from the cryptographic module. This information should be reported to IBM service.

Table 89. VERUPDAT specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Verification operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	16	116	The program verification module (IRRPVERS) is not loaded. See *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF System Programmer's Guide* for information about configuring and loading the verification module with the IRRVERLD program.
8	2nn	xx	Unexpected error from the cryptographic module. The return code is 200+*nn* where *nn* identifies the function being performed. The reason code is the return code from the cryptographic module. This information should be reported to IBM service.

Table 90. VERFINAL specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	See reason codes below for SAF return code 8, RACF return code 16	Signature failed verification. Continue the load.
8	8	100	Verification operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	16	See below	Signature failed verification. Fail the load.
The following group of reason codes are considered problems with the program signature (the zOSSignatureInfo structure). These would cause the load to fail when FAILLOAD(BADSIGONLY) or FAILLOAD(ANYBAD) is in effect.			
		4	The ZOSSignatureInfo structure is missing or not correct.
		8	Signature algorithm in ZOSSignatureInfo is not supported.
		12	Signer certificate is revoked. The certificate status is NOTRUST.
		16	Certificate chain is incomplete.

Table 90. VERFINAL specific return and reason codes (continued)

SAF return code	RACF return code	RACF reason code	Explanation
		20	One or more CA certificates do not have certificate signing capability. (KeyUsage extension present but keyCertSign flag is off or BasicConstraints extension is present but cA flag is off.)
		24	End-entity certificate does not have code signing capability. (KeyUsage extension present but digitalSignature or nonRepudiation flag is off.)
		28	The certificate signature algorithm of one or more certificates is not supported.
		32	The type or size of key found in one or more certificates is not supported.
		36	CA or signing certificate was expired or not yet active at the time the module was signed.
		40	Digital signature not valid.
		44	Unsupported certificate format.
The following group of reason codes are the additional conditions that would cause the load to fail due to signature processing, but do not represent a bad signature. These would cause the load to fail when FAILLOAD(ANYBAD) is in effect, but not FAILLOAD(BADSIGONLY).			
		100	The program appears to be correctly signed but one of the following conditions exists: • The root CA certificate in the zOSSignatureInfo structure of the program object is not connected to the signature-verification key ring. • The root CA certificate is marked NOTRUST.
		104	The FACILITY class profile, IRR.PROGRAM.SIGNATURE.VERIFICATION, is missing.
		108	The APPLDATA information in the FACILITY class profile, IRR.PROGRAM.SIGNATURE.VERIFICATION, is missing or not correct.
		112	The signature-verification key ring is missing.
		116	The program verification module (IRRPVERS) is not loaded. See *z/OS Security Server RACF Security Administrator's Guide* and *z/OS Security Server RACF System Programmer's Guide* for information about configuring and loading the verification module with the IRRVERLD program.
		120	An error occurred while performing a cryptographic self test on the IRRPVERS module during initialization. Contact IBM support.
8	2nn	xx	Unexpected error from the cryptographic module. The return code is 200+*nn* where *nn* identifies the function being performed. The reason code is the return code from the cryptographic module. This information should be reported to IBM service.

Table 91. VERCLEAN specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
8	8	100	Verification operation has not been initialized for the specified program name.
8	12	xx	Unexpected error returned from IEANTRT. RACF reason code is the return code returned by IEANTRT.
8	16	xx	Unexpected error returned from IEANTDL. RACF reason code is the return code returned by IEANTDL.

Table 92. VERINTER specific return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Continue the load.
8	8	0	Fail the load.

Usage notes

Usage notes for program signing

1. This service tracks the resources used for signing using a task-related name/token pair. The 16–byte token name has the following format:

   `IRRPSIGN`*`program-name`*

   Where *program-name* is one of the parameters provided by the caller. Consequently, for any given series of SIGINIT, SIGUPDAT, SIGFINAL, and SIGCLEAN calls used to sign a single program object, the program name value must be the same.
2. Calls to this service using different program name values are considered independent operations.
3. For a given program name, SIGINIT must be called before calling any of SIGUPDAT, SIGFINAL, or SIGCLEAN.
4. For a given program name, SIGINIT cannot be called a second time without terminating the first SIGINIT with a call to SIGFINAL or SIGCLEAN.
5. For a given program name, it is the caller's responsibility to call the SIGCLEAN function if signature generation is not completed by calling SIGFINAL. Note that all R_PgmSignVer functions will perform this cleanup if they return an error to the caller. The caller must call the cleanup function if it is terminating for its own reason.
6. The signature area allocated and returned to the caller in the PGSN_SF_SIG_AREA@ parameter by SIGFINAL has the following format:

Table 93. PGSN_SF_SIG_AREA@ signature area format

Offset	Length	Description
0	4	Eyecatcher, "PSSD".
4	4	Length of entire area, including the eyecatcher.
8	1	Subpool used to obtain the area storage.
9	3	Reserved.
12	4	Length of z/OS signature information area.

Table 93. PGSN_SF_SIG_AREA@ signature area format (continued)

Offset	Length	Description
16	*	ZOSSignatureInfo structure to be included in the signed program object. See the next usage note for the format.

7. The ZOSSignatureInfo structure returned in the signature area is the signature data that is to be placed in the signed program object. It is DER encoded according to the following ASN.1 definition:

```
ZOSSignatureInfo ::= SEQUENCE {
   signDetails     SignatureDetails
   certs           SET OF Certificate -- In reverse hierarchy order, EE to root
   signature       BIT STRING         -- PKCS #1 format - Encrypted DigestInfo
}

SignatureDetails ::= SEQUENCE {        -- DER encoding included in data signed
   version         INTEGER(0)
   signatureAlg    AlgorithmIdentifier -- From PKCS #1
   signatureTime   OCTET STRING(12)    -- TIME DEC,ZONE=UTC,DATETYPE=YYYYMMDD
                                       -- format (EBCDIC)
}
```

8. The only supported algorithm for the signatureAlg field is sha256WithRSAEncryption with NULL parameters.
9. It is the caller's responsibility to free the signature area when it is no longer needed.
10. The only supported message digest algorithm is SHA256.
11. The only supported certificate key type is RSA. The maximum RSA key size is 4096 bits.
12. The supported certificate signature algorithms are:
    - sha256WithRSAEncryption
    - sha1WithRSAEncryption
13. All numeric parameters are treated as unsigned.
14. All length parameters must be non-zero unless otherwise indicated.
15. On SIGINIT, if the key ring to use is not specified, the security manager determines the key ring that is based on security settings. See the *z/OS Security Server RACF Security Administrator's Guide* for information on these security settings and on how to populate the key ring . There can be no more than 10 certificates within the trust chain, starting with the code signer and ending with the self-signed certificate authority certificate.
16. If no program data is ever passed in by the caller, a digital signature is generated solely for the SignatureDetails structure documented above.

Usage notes for program verification

1. For unauthorized callers, this service tracks the resources used for verification in a 'context' using a task related name/token pair. The 16–byte token name has the following format:

   IRRPVERF*program-name*

   Where program-name is one of the parameters provided by the caller. Consequently, for any given series of VERINIT, VERUPDAT, VERFINAL, and VERCLEAN calls used to verify the signature of a single program object, the program name must be the same.
2. Calls to this service using different program names are considered independent operations.

3. For a given program name, VERINIT must be called before calling any of VERUPDAT, VERFINAL (with the exception documented in the descriptions of the PGSN_VF_CONTEXT@ and PGSN_VF_PGM_NAME_LEN fields in the VERFINAL parameter list), or VERCLEAN.
4. For a given program name, VERINIT cannot be called a second time without terminating the first VERINIT with a call to VERFINAL or VERCLEAN.
5. For a given program name, it is the caller's responsibility to call the VERCLEAN function in the event that signature generation will not be completed by calling VERFINAL. Note that all R_PgmSignVer functions will perform this cleanup if they return an error to the caller. The caller only needs to call the cleanup function if it is terminating for its own reason.
6. If auditing is required, it is performed in the VERFINAL (or VERINTER) call. Auditing is only performed when the ICHSFENT is provided by an authorized caller, subject to the audit settings from the directive within and the outcome of the VERFINAL service.
7. Some signature generation and all verification functions allow, from a pointer in the function-specific parameter list, the specification of an array of ranges of data to be hashed. This is optional. If the address is 0, no data will be hashed. The ranges are defined using the structure mapped by PGSN_DATA_RANGE in the IRRPCOMP mapping macro. This structure must exist in storage within the primary address space. The structure consists of an ALET followed by a fullword specifying the number of ranges which follow (if the number of ranges is 0, no data will be hashed). This is followed by an array of pointer pairs. Each pointer is an 8-byte pointer. AMODE(31) callers must set the high order fullword of the pointer fields to 0. The first pointer is the address of the first byte of the range, and the second pointer is the address of the last byte of the range (they can be the same, for a length of 1). The maximum number of ranges which can be specified per call is defined in the PGSN_DATA_NUM_RANGES_MAX constant.

Field	Attributes	Description
PGSN_DATA_RANGE	Structure	Ranges of data to verify.
PGSN_DATA_ALET	4 byte numeric	The ALET for the address space containing the data.
PGSN_DATA_NUM_RANGES	4 byte numeric	The number of data ranges in the following array, not to exceed PGSN_DATA_NUM_RANGES_MAX.
PGSN64_DATA_RANGE_LIST	Array	Repeating array of the following data items.
PGSN_DATA_START@	Address of	Address of the first byte in the range.
PGSN_DATA_END@	Address of	Address of the last byte in the range.

8. The default message digest algorithm is SHA256. This is the only supported message digest algorithm.
9. The ZOSSignatureInfo structure is DER encoded. It has the following ASN.1 definition:

```
ZOSSignatureInfo ::= SEQUENCE {
   signDetails     SignatureDetails
   certs           SET OF Certificate -- In reverse hierarchy order, EE to root
   signature       BIT STRING         -- PKCS #1 format - Encrypted DigestInfo
}

SignatureDetails ::= SEQUENCE {       -- DER encoding included in data signed
   version         INTEGER(0)
```

```
    signatureAlg    AlgorithmIdentifier -- From PKCS #1
    signatureTime  OCTET STRING(12)      -- TIME DEC,ZONE=UTC,DATETYPE=YYYYMMDD
                                         -- format (EBCDIC)
}
```

10. The only supported algorithm for the signatureAlg field is sha256WithRSAEncryption with NULL parameters.
11. The only supported certificate key type is RSA. The maximum RSA key size is 4096 bits.
12. The supported certificate signature algorithms are:
    - sha256WithRSAEncryption
    - sha1WithRSAEncryption
13. All numeric parameters are treated as unsigned.
14. All length parameters must be non-zero unless otherwise indicated.
15. The program signature-verification key ring is specified using the APPLDATA field of FACILITY class profile IRR.PROGRAM.SIGNATURE.VERIFICATION. See *z/OS Security Server RACF Security Administrator's Guide* for more information about creating profiles.
16. If there is no ICHSFENT, and thus no directive, which is supplied by the caller, the verification occurs on the signature, but there is no check for the root CA certificate being trusted, and no auditing performed.

Related services

None.

R_PKIServ (IRRSPX00): Request public key infrastructure (PKI) services

Function

The R_PKIServ SAF callable service allows applications to request the generation, retrieval, and administration of X.509 V.3 certificates and certificate requests through z/OS Cryptographic Services PKI Services. See *z/OS Cryptographic Services PKI Services Guide and Reference* for more information on this service.

Requirements

Authorization:
Any PSW key in supervisor or problem state

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have a FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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.

RACF authorization

The RACF authorization mechanism for this callable service varies depending on the type of function requested (end user versus administrative) and the requested provider (SAF versus PKI Services).

For the end user functions, this interface is protected by FACILITY class profiles (resources) of the form IRR.RPKISERV.*(function)*[.*ca-domain*], where *(function)* is one of the end user function names described under Function_code below. If the CA_domain parameter supplied on the R_PKIServ call is not null (has a length greater than 0), the profile is qualified with the CA domain name. If the CA_domain parameter supplied on the R_PKIServ call is null, the qualifier is not used. For example, if the function name is GENCERT and the CA_domain parameter is "Customers", the FACILITY class resource is IRR.RPKISERV.GENCERT.CUSTOMER. However, if the CA_domain parameter is null, the FACILITY class resource is IRR.RPKISERV.GENCERT.

The user ID (from the ACEE associated with the address space) for the application is used to determine access:

NONE
Access is denied.

READ
Access is permitted based on subsequent access checks against the caller's user ID. To determine the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.

UPDATE
Access is permitted based on subsequent access checks against the application's user ID.

ALTER OR CONTROL (or user ID is RACF SPECIAL)
Access is permitted with no subsequent access checks made.

For SAF GENCERT and EXPORT requests where the application has READ and UPDATE access, subsequent access checks are performed against the IRR.DIGTCERT.(function) FACILITY profiles. These are identical to the checks made by the RACDCERT TSO command. See *z/OS Security Server RACF Command Language Reference* and *z/OS Security Server RACF Security Administrator's Guide* for more information.

For PKI Services GENCERT, REQCERT, EXPORT, VERIFY, REVOKE, GENRENEW, REQRENEW, RESPOND, SCEPREQ and QRECOVER requests where the application has READ and UPDATE access, subsequent access checks are performed against the IRR.DIGTCERT.*function* FACILITY profiles as follows:

- GENCERT — This function is used to request an auto-approved certificate. The access check user ID needs to have CONTROL access to IRR.DIGTCERT.GENCERT. The access check user ID also needs appropriate access to IRR.DIGTCERT.ADD, UPDATE access if any HostIdMapping information is specified in the certificate request parameter list or the Userid field in the certificate request parameter list indicates that the certificate is being requested for another user other than the caller, otherwise READ access.
- REQCERT — This function is used to request a certificate that must be approved by an administrator before being created. The access check user ID needs to have READ access to IRR.DIGTCERT.REQCERT
- EXPORT — This function is used to retrieve (export) a certificate that was requested previously or the PKI Services RA/CA certificate. The access check user ID needs to have appropriate access to IRR.DIGTCERT.EXPORT, UPDATE access if no pass phrase is specified on the call, READ access if a pass phrase is specified or the Cert ID is "PKICACERT".
- VERIFY — This function is used to confirm that a given user certificate was issued by this CA and if so, return the certificate fields. The access check user ID needs to have READ access to IRR.DIGTCERT.VERIFY. It is assumed that the calling application has already verified that the end user possesses the private key that correlates to the input certificate.
- REVOKE — This function is used to revoke a certificate that was previously issued. The access check user ID needs to have READ access to IRR.DIGTCERT.REVOKE. It is assumed that the calling application has already verified the target certificate using the VERIFY function.
- GENRENEW — This function is used to generate a renewal certificate. The request submitted is automatically approved. The access check user ID needs to have READ access to IRR.DIGTCERT.GENRENEW and CONTROL access to IRR.DIGTCERT.GENCERT. It is assumed that the calling application has already verified the input certificate using the VERIFY function.
- REQRENEW — This function is used to request certificate renewal. The request submitted needs to be approved by the administrator before the certificate is renewed. The access check user ID needs to have READ access to IRR.DIGTCERT.REQRENEW. It is assumed that the calling application has already verified the input certificate using the VERIFY function.
- RESPOND — This function is used to get an Online Certificate Status Protocol (OCSP) response from the PKI Services responder. The access check user ID needs to have READ access to IRR.RPKISERV.RESPOND and IRR.DIGTCERT.RESPOND.
- SCEPREQ — This function is used to request a certificate using SCEP. The access check user ID needs to have READ access to IRR.DIGTCERT.SCEPREQ
- QRECOVER — This function is used to get a list of certificates whose key pairs were generated by PKI Services under a particular email address and pass phrase. The access check user ID needs READ access to IRR.DIGTCERT.QRECOVER.

For the administrative functions, this interface is protected by a single FACILITY class profile (resource), IRR.RPKISERV.PKIADMIN[.*ca-domain*]. If the CA_domain parameter supplied on the R_PKIServ call is not null (has a length greater than 0), the profile is qualified with the CA domain name. If the CA_domain parameter supplied on the R_PKIServ call is null, the qualifier is not used. For example, if the CA_domain parameter is “Customers”, the FACILITY class resource is IRR.RPKISERV.PKIADMIN.CUSTOMER. However, if the CA_domain parameter is null, the FACILITY class resource is IRR.RPKISERV.PKIADMIN.

If the caller is not RACF SPECIAL, the caller will need READ access to perform read operations (QUERYREQS, QUERYCERTS, REQDETAILS, and CERTDETAILS) and UPDATE access for the action operations (PREREGISTER, MODIFYREQS, and MODIFYCERTS). To determine the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.

Format

```
CALL IRRSPX00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               Number_parameters,
               Function_code,
               Attributes,
               Log_string,
               Parm_list_version,
               Function_parmlist,
               CA_domain
               )
```

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 word 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.

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.

Number_parameters
Specifies the name of a fullword that contains the number of parameters that follow in the non-request specific portion of the R_PKIServ callable service invocation. If the CA_domain parameter is specified, Number_parameters must be set to 6. Otherwise, Number_parameters must be set to 5.

Function_code
The name of a 2-byte area containing the function code. The function code has one of the following values:

End user functions:

- X'0001'-Generate a basic X.509V3 certificate using the application-provided data pointed to by the function-specific parameter list (Function name GENCERT).
- X'0002'-Extract certificate by certificate request ID (Function name EXPORT).
- X'0009'-Submit a request for a X.509V3 certificate using the application provided data pointed to by the function-specific parameter list. Similar to

function 1, except that the request would be pending the approval of the PKI Services administrator (Function name REQCERT).

- X'000A'-Verify that certificate was issued by this PKI Services CA and return certificate fields (Function name VERIFY).
- X'000B'-Revoke a PKI Services certificate (Function name REVOKE).
- X'000C'-Generate a renewal PKI Services certificate (Function name GENRENEW).
- X'000D'-Request a renewal certificate from PKI Services (Function name REQRENEW).
- X'000E' -Get an Online Certificate Status Protocol (OCSP) response from the PKI Services responder (Function name RESPOND).
- X'000F' -Submit a request to PKI Services using SCEP (Function name SCEPREQ).
- X'0011'-List certificates whose key pairs were generated by PKI services for a particular requester. The requester is identified by the email address and pass phrase provided when generating the certificates and key pairs. (Function name QRECOVER).

Administrative functions-

- X'0003'-Query PKI Services for certificate requests (Function name QUERYREQS).
- X'0004'-Get detailed information pertaining to one PKI Services certificate request (Function name REQDETAILS).
- X'0005'-Modify PKI Services certificate requests (Function name MODIFYREQS).
- X'0006'-Query PKI Services issued certificates (Function name QUERYCERTS).
- X'0007'-Get detailed information pertaining to one PKI Services issued certificate (Function name CERTDETAILS).
- X'0008'-Modify PKI Services issued certificates (Function name MODIFYCERTS).
- X'0010'-Preregister a user (Function name PREREGISTER).

Attributes
The name of a 4-byte area containing bit settings that direct the function to be performed. This is a reserved field that must be specified. The bit settings are mapped as follows:

- Functions GENCERT (X'0001) and GENRENEW (X'000c')-x'80000000' - Do not return control until the certificate has been generated. The request will be purged if unsuccessful for any reason.

  All other bit positions are reserved and must be set to zero.
- All other functions

  All bit positions are reserved and must be set to zero.

Log_string
The name of an area that consists of a 1-byte length field followed by character data to be included in any audit records that are created as a result of the R_PKIServ invocation. The first eight bytes of the Log_string data specified on a GENCERT and RENEW request is also used as application data(ApplData) to be stored with the certificate. If not specified, the length must equal 0.

Parmlist_version
The name of a 4-byte input value which contains the version number for the

following input field, Function_parmlist. To take full advantage of the support provided by this release, this field should be set to 1 for the EXPORT and MODIFYREQS functions, and should be set to 2 for the MODIFYCERTS function. For all other functions this field must be set to 0.

Function_parmlist
Specifies the name of the function code specific parameter list for the Function_code specified:

Table 94. Function_parmlist for GENCERT

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example 'GENCERT'.
CertPlistLen	4-byte length	In	Describes the length in bytes of the certificate generation plist.
CertPlist	Address of	In	The name of the area which is the CertGen request parameter list. This area maps the specific name, length, address/data values which are used in satisfying the certificate request for the specified user. Also, see Table 95.
Certid	Address of	In/Out	Points to a 57-byte area, in which the first byte will contain the actual length on return of the certificate request ID. The storage address specified must be obtained by the caller, and freed by the caller. The returned certificate request ID is used to extract the completed certificate, if the request has been accepted by RACF.

The GENCERT and REQCERT functions have in essence two connected parameter list areas; the function-specific parameter list as defined above and the CertGen request parameter list (CertPlist) containing specific certificate field information. CertPlist is a list of ordered triplets that consists of name, length, and data value. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks. The length field is a binary 4-byte value, which qualifies the length of the data item. Note that all data values are EBCDIC character data unless otherwise indicated. The following table describes the valid certificate request fields:

Table 95. CertPlist for GENCERT and REQCERT

Field name	Max length	Description
DiagInfo	80 bytes (exactly)	EyeCatcher to identify this request in virtual storage for diagnostic reasons. For certificate generation warnings and errors, RACF will update the field with diagnostic information. The length will be updated as well. Required field must be the first field in the CertPlist.
SerialNumber	64 bytes	Serial number, or registration number, of the subject device. Optional. No default. Only valid with PKI Services requests.
UnstructAddr	64 bytes	Unstructured address of the subject device. Optional. No default. Only valid with PKI Services requests.

Table 95. CertPlist for GENCERT and REQCERT (continued)

Field name	Max length	Description
UnstructName	64 bytes	Unstructured name of the subject device. Optional. No default. Only valid with PKI Services requests.
EmailAddr	64 bytes	Subject's email address for distinguished name EMAIL= attribute. Optional. No default. Only valid with PKI Services requests.
Mail or Email	64 bytes	Subject's email address for distinguished name MAIL= attribute. Optional. No default. Only valid with PKI Services requests. (Field name "Email" is deprecated, use "Mail")
DNQualifier	64 bytes	Subject's Distinguished Name Qualifier. Optional. No default. Only valid with PKI Services requests.
Uid	64 bytes	Subject's login ID. Optional. No default. Only valid with PKI Services requests.
CommonName	64 bytes	Subject's common name. Optional. No default, except in the following situation: If specified with a null value (length 0). RACF will use the PGMRNAME field from the RACF user profile as determined by the user ID field for this request. If PGMRNAME is null, the common name will be of the form of RACF UserID:*(user's-racf-identify)*, for example RACF UserID:JSWEENY
Title	64 bytes	Subject's Title. Optional. No default.
DomainName	64 bytes	Subject's Domain Name containing all the domain components in the form of *domain-component-1.domain-component-2.domain-component-3...domain-component-n*. Optional. No default. Only valid with PKI Services requests.
OrgUnit	64 bytes	Subject's Organizational Unit. Note that this field may be repeated. RACF concatenates in the order of appearance to construct the hierarchy of organizational units. Optional. No default.
Org	64 bytes	Subject's Organization. Optional. No default.
Street	64 bytes	Subject's street address. Optional. No default. Only valid with PKI Services requests.
Locality	64 bytes	Subject's City or Locality. Optional. No default.
StateProv	64 bytes	Subject's State/Providence. Optional. No default.
PostalCode	64 bytes	Subject's postal code or postal code. Optional. No default. Only valid with PKI Services requests.
Country	2 bytes	Subject's Country. Optional. No default.
KeyUsage	20 bytes	One of 'handshake' | 'dataencrypt' | 'certsign' | 'docsign' | 'digitalsignature' | 'digitalsig' | 'nonrepudiation' | 'keyencipherment' | 'keyenciph' | 'keyencrypt' | 'dataencipherment' | 'dataenciph' | 'keyagreement' | 'keyagree' | 'keycertsign' | 'crlsign' (not case sensitive, no quotes). Note that this field may be repeated to request multiple usages. Optional. No default.

Table 95. CertPlist for GENCERT and REQCERT (continued)

Field name	Max length	Description
ExtKeyUsage	20 bytes	One of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' (not case sensitive, no quotes). Note this field may be repeated to request multiple usages. Optional. No default. Only valid with PKI Services requests.
NotBefore	2 bytes	Number of days from today's date that the certificate becomes valid. Range 0-30. Validity checked by RACF. Optional. Default is 0.
NotAfter	4 bytes	Number of days from today's date that the certificate expires. Range 1–9999. The number of days in the validity period is calculated as NotAfter minus NotBefore, therefore the NotAfter value must be greater than the NotBefore value. Validity checked by RACF. Optional. Default is 365.
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternate name extension. Optional. No default. Note that this field may be repeated in the PKI Services requests.
AltURI	255 bytes	Uniform Resource Identifier for subject alternate name extension. Optional. No default. Note that this field may be repeated in the PKI Services requests.
AltEmail	100 bytes	Email address for subject alternate name extension. Optional. No default. Note that this field may be repeated in the PKI Services requests.
AltDomain	100 bytes	Domain Name for subject alternate name extension. Optional. No default. Note that this field may be repeated in the PKI Services requests.
AltOther	255 bytes	Other Name for subject alternate name extension. Optional. No default. Only valid with PKI Services requests.
NotifyEmail	64 bytes	Email address for notification purposes. Its value is copied from the Requestor field when Keysize is specified. Optional. No default. Only valid with PKI Services requests.
PublicKey	65535 bytes	PKCS #10 or Netscape Navigator certificate request or CMP CertReqMsg structure containing the public key to be certified. This is base 64 encoded DER. Required field if KeySize is not specified.
KeySize	4 bytes	Size of the key in bits if key pair is to be generated by PKI services. Range 512–4096 for RSA keys; For NISTECC keys: 192, 224, 256, 384 and 521; for BPECC keys: 160, 192, 224, 256, 320, 384 and 512. Required field if PublicKey is not specified. Only valid with PKI Services requests.

Table 95. CertPlist for GENCERT and REQCERT (continued)

Field name	Max length	Description
KeyAlg	10 bytes	Algorithm of the key if the key pair is to be generated by PKI Services. The acceptable values are 'RSA', 'NISTECC' and 'BPECC'. Optional. Default to RSA. Only valid with PKI Services requests.
SignWith	45 bytes	For SAF this is the label of z/OS certificate authority certificate to sign the completed certificate request. Format: SAF:CERTAUTH/<ca-cert-label> or SAF:/<ca-cert-label>, where ca-cert-label is the certificate label under CERTAUTH or the caller's user ID. May also be used to indicate that PKI Services should process the request rather than SAF. In this case, the format is PKI: (exactly 4 characters). Required field.
HostIdMap	100 bytes	HostIdMapping extension entry in the form of an email address, for example, gumby@plpsc.pok.ibm.com. The rightmost '@' is used to delineate the subjectId from the hostName. Optional. No default. Only valid with PKI Services requests. Note that this field may be repeated.
Requestor	32 bytes	Name of the person submitting the request. If KeySize is specified, this is a required field in the form of an email address. Otherwise, it is an optional field derived from the UnstructuredName if not specified. If UnstructName is not specified, Requestor will be derived from CommonName. If CommonName is also not specified, Requestor will be derived from the first RDN of the subject's name. Only valid with PKI Services requests.
PassPhrase	32 bytes	Value to be used for challenge or response when retrieving the certificate through function EXPORT. If KeySize is specified, this is a required field. Otherwise, it is optional. No default. Only valid with PKI Services requests.
UserId	8 bytes	Subject's RACF UserID. If not specified, the user ID is taken from the ACEE. For SAF requests, this is the User ID that will own the certificate. For SAF requests, this is the User ID that will own the certificate. For PKI requests, the user ID is used only to determine the Common Name when CommonName is specified without a value.
Label	32 bytes	Up to 32 mixed case characters that may be used as the 'handle'. Optional. Default is that one will be generated and added to the user's list of certificates. Only valid with SAF requests.

Table 95. CertPlist for GENCERT and REQCERT (continued)

Field name	Max length	Description
CertPolicies	32 bytes	Blank-separated array of non-repeating policy numbers, range 1 - 99, for the CertificatePolicies extension. These correspond to the policy numbers defined during PKI Services configuration. Each value must be one or two digits with no leading zeros. Optional. No default. Only valid with PKI Services requests.
AuthInfoAcc	255 bytes	A comma-separated two-part string specifying information used to create the AuthorityInfoAccess extension. The two-part string identifies the accessMethod and accessLocation. The accessMethod is one of 'OCSP' | 'IdentrusOCSP' (not case sensitive, no quotes). The accessLocation is a URI in the form URI=access-url or URL=access-url. Optional. No default. Only valid with PKI Services requests. Note that this field may be repeated.
Critical	32 bytes	Name of a certificate extension to be marked critical. One of 'BasicConstraints' | 'KeyUsage' | 'ExtKeyUsage' | 'SubjectAltName' | 'AltEmail' | 'AltIPAddr' | 'AltDomain' | 'AltURI' | 'HostIdMappings' | 'HostIdMap' | 'CertificatePolicies' | 'CertPolicies' (not case sensitive, no quotes). Optional. The BasicConstraints and KeyUsage extensions are always marked critical even if not specified. Only valid with PKI Services requests. Note this field may be repeated.
CustomExt	1024 bytes	Customized extension in the form of a comma-separated four-part string. The first part is the OID of the extension. The second part is the critical flag – 'C'(critical) or 'N'(non-critical), the third part is the encode type – 'INT'(integer), 'IA5'(IA5 string), 'PRT'(printable string), 'BMP'(BMP string) , 'OCT'(Octet string) or 'UTF'(UTF8 string), the last part is the value. The critical flag and the encode type are not case sensitive. Note 'C'(critical) is allowed only if KeySize is not specified. Optional. No default. Only valid with PKI Services requests. This field may be repeated. For more information, see Forming the CustomExt value for CertPlist for the R_PKIServ callable service in *z/OS Cryptographic Services PKI Services Guide and Reference*.
BusinessCat	64 bytes	Subject's business category. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurCountry	2 bytes	Subject's Jurisdiction Country. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.

Table 95. CertPlist for GENCERT and REQCERT (continued)

Field name	Max length	Description
JurStateProv	64 bytes	Subject's Jurisdiction State/Province. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurLocality	64 bytes	Subject's Jurisdiction Locality. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.

Table 96. Function_parmlist for EXPORT

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example 'EXPORT'.
CertAnchorLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the Cert Anchor area on input to EXPORT. RACF will update this value with the actual length of the certificate returned. If the storage area as specified by the cert anchor address is too small, RACF will set a failing return or reason code and update the length field to the size required. The caller must allocate a larger area.
CertAnchor	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the certificate that is specified by the CertID parameter if the service was able to successfully retrieve the completed certificate. If the caller has supplied an area that is too small, (based in the CertAnchorLen), this service fails the request, and updates the CertAnchorLen field to indicate the actual storage required to store the certificate.

Table 96. Function_parmlist for EXPORT (continued)

Field	Attributes	Usage	Description
CertId	Address of	In	Points to a 57-byte area, in which the first byte will contain the actual length of the input certificate request ID or serial number that will be used to locate the certificate to be exported. For PKI Services requests where PassPhrase was specified on the GENCERT, the user-provided pass phrase must be appended to the actual CertId value and included here. The leading length byte must account for the additional length. For PKI Services requests where SCEP is enabled in the PKI Services daemon, the constant value "PKICACERT" can be specified to retrieve the CA certificate or CA/RA certificate pair. No PassPhrase is used in this case. If the KeyID value is specified, a serial number is expected instead of a request ID. The serial number is a 16-byte value in the form of printable EBCDIC (HEX) with leading 0's (for example, 0000000000001A5F).
KeyId	Address of	In	Points to a 41-byte area. The first byte will contain the length of the KeyId, which is a hash of the public key generated by PKI Services. When this field is being used to export a recovery certificate, the first byte is 40. In all the other cases, the first byte should be 0. The minimum Parmlist_Version for the use of this parameter is 1.

Table 97. Function_parmlist for QUERYREQS

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' QUERYREQS '
ResultsListLen	4-byte length	In/Out	4–byte area that is the length of the pre-allocated storage of the Results List area on input to QUERYREQS. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Results List address is too small, RACF will set a failing return or reason code and update the length field to the size required. The caller must allocate a larger area.

Table 97. Function_parmlist for QUERYREQS (continued)

Field	Attributes	Usage	Description
ResultsList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area that is too small, (based in the ResultsListLen), this service fails the request, and updates the ResultsListLen field to indicate the actual storage required to store the data.
CertId	Address of	In	Points to a 57-byte area, in which the first byte will contain the actual length of the input certificate request ID that will be used as a starting point for this query. Only requests located after this request will be returned. If the first byte is zero (x'00'), then the query will start with the first request.
NumEntries	4-byte numeric	In/Out	Input value indicating the maximum number of entries that should be returned in the ResultsList area. Zero indicates no limit. Updated to indicate the number of entries actually returned.
CriteriaStatus	4-byte numeric	In	Value indicating the request status to use as search criteria. • X'00000000' - return all requests, • X'00000001' - return requests pending approval only, • X'00000002' - return requests that have been approved only, • X'00000003' - return completed requests only, • X'00000004 - return all rejected requests only, • X00000005 - return rejected requests in which the client has been notified only, • X'00000006' - return preregistered requests only.
CriteriaDays	4-byte numeric	In	Value indicating the recent activity time period to use as additional search criteria. The time period is the number of days in the past that should be scanned for requests that have been created or modified. If zero (x'00000000'), recent activity will not be used as additional search criteria.

Table 97. Function_parmlist for QUERYREQS (continued)

Field	Attributes	Usage	Description
CriteriaName	Address of	In	Points to a 33-byte area, in which the first byte will contain the actual length of the input requestor's name to be used as additional search criteria. If the first byte is zero (x'00'), then the requestor's name will not be used as additional search criteria.

The QUERYREQS function returns results in the ResultsList area provided by the caller. The ResultsList has the following format:

Table 98. ResultsList for QUERYREQS

Length	Value
1-byte length	Entry 1's certificate request ID, max 56 bytes
1-byte length	Entry 1's requestor's name, max 32 bytes
1-byte length	Entry 1's subject's distinguished name, max 255 bytes
1-byte length	Entry 1's issuer's distinguished name, max 255 bytes
1-byte length	Entry 1's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Zero bytes for requests with a status of “Preregistered”, otherwise exactly 41 bytes.
1-byte length	Entry 1's keyUsage value, max 64 bytes (one or more of 'handshake' | 'dataencrypt' | 'certsign' | 'docsign' | 'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry 1's status, max 32 bytes (one of "Preregistered", "Pending Approval", "Approved", "Completed", "Rejected", or "Rejected, User Notified")
1-byte length	Entry 1's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 1's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 1's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry 1's serial number if certificate has been issued, max 16 bytes
1-byte length	Entry 1's previous serial number if this is a renewal request, max 16 bytes
1-byte length	Entry 1's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified").
1-byte length	Entry 2's certificate request ID, max 56 bytes
1-byte length	Entry 2's requestor's name, max 32 bytes
1-byte length	Entry 2's subject's distinguished name, max 255 bytes
1-byte length	Entry 2's issuer's distinguished name, max 255 bytes

Table 98. ResultsList for QUERYREQS (continued)

Length	Value
1-byte length	Entry 2's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Zero bytes for requests with a status of “Preregistered”, otherwise exactly 41 bytes.
1-byte length	Entry 2's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt |, 'certsign' |'docsign' |'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry 2's status, max 32 bytes (one of "Preregistered", "Pending Approval", "Approved", "Completed", "Rejected", or "Rejected, User Notified")
1-byte length	Entry 2's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 2's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 2's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry 2's serial number if certificate has been issued, max 16 bytes
1-byte length	Entry 2's previous serial number if this is a renewal request, max 16 bytes
1-byte length	Entry 2's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified").

⋮

Length, continued	Value, continued
1-byte length	Entry n's certificate request ID, max 56 bytes
1-byte length	Entry n's requestor's name, max 32 bytes
1-byte length	Entry n's subject's distinguished name, max 255 bytes
1-byte length	Entry n's issuer's distinguished name, max 255 bytes
1-byte length	Entry n's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Zero bytes for requests with a status of “Preregistered”, otherwise exactly 41 bytes.
1-byte length	Entry n's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt' |'certsign' |'docsign' |'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry n's status, max 32 bytes (one of "Preregistered", "Pending Approval", "Approved", "Completed", "Rejected", or "Rejected, User Notified")
1-byte length	Entry n's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry n's last modified date in YYYY/MM/DD form, exactly 10 bytes

Length, continued	Value, continued
1-byte length	Entry n's ApplData value from the GENCERT or REQCERT n's last modified date in YYYY/MM/DD invocation, max 8 bytes
1-byte length	Entry n's serial number if certificate has been issued, max 16 bytes
1-byte length	Entry n's previous serial number if this is a renewal request, max 16 bytes
1-byte length	Entry n's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified").

Table 99. Function_parmlist for REQDETAILS

FIELD	ATTRIBUTES	USAGE	DESCRIPTION
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' REQDTAIL'
SumListLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the Summary List area on input to REQDETAILS. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Summary List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
SumList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the SummaryListLen), this service fails the request, and updates the SummaryListLen field to indicate the actual storage required to store the data. Also, see Table 100 on page 272.
CertPlistLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the certificate generation plist area on input to REQDETAILS. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Results List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
CertPlist	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the CertPlistLen), this service fails the request, and updates the CertPlistLen field to indicate the actual storage required to store the data.. This area maps some of the specific name, length, address/data values which were used when generating the original certificate request on GENCERT. Also, see Table 101 on page 273.

Table 99. Function_parmlist for REQDETAILS (continued)

FIELD	ATTRIBUTES	USAGE	DESCRIPTION
CertId	Address of	In	Points to a 57-byte area, in which the first byte will contain the actual length of the input certificate request ID from which details are to be extracted

The REQDETAILS function returns QUERYREQS style summary data in the SumList area provided by the caller. The SumList has the following format:

Table 100. SumList for REQDETAILS

Length	Value
1-byte length	Entry's certificate request ID, max 56 bytes
1-byte length	Entry's requestor's name, max 32 bytes
1-byte length	Entry's subject's distinguished name, max 255 bytes
1-byte length	Entry's issuer's distinguished name, max 255 bytes
1-byte length	Entry's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Zero bytes for requests with a status of “Preregistered”, otherwise exactly 41 bytes
1-byte length	Entry's keyUsage value, max 64 bytes (one or more of 'handshake'| 'dataencrypt'|, 'certsign'| 'docsign' | 'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry's status, max 32 bytes (one of "Pending Approval", "Approved", "Completed", "Rejected", or "Rejected, User Notified")
1-byte length	Entry's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry's serial number if certificate has been issued, max 16 bytes
1-byte length	Entry's previous serial number if this is a renewal request, max 16 bytes
1-byte length	Entry's last action comment, max 64 bytes
1-byte length	Entry's pass phrase provided when the certificate request was made, max 32 bytes
1-byte length	Entry's notification email address, max 64 bytes
1-byte length	Entry's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' or "not specified").
1-byte length	Entry’s certificate request fingerprints - 40 byte SHA1 hash (printable EBCDIC) followed by 32 byte MD5 hash (printable EBCDIC), Exactly 72 bytes if available. Zero bytes if unavailable.
1–byte length	Entry’s certificate request fingerprints - 64 byte SHA256 hash (printable EBCDIC) followed by 128 byte SHA512 hash (printable EBCDIC). Exactly 192 bytes if available. Zero bytes if unavailable.
1–byte length	Entry’s request signature algorithm, max 32 bytes, one of 'sha-1WithRSAEncryption' , 'sha-224WithRSAEncryption' , 'sha-256WithRSAEncryption' , 'sha-384WithRSAEncryption' , 'sha-512WithRSAEncryption' , 'md-5WithRSAEncryption' , 'md-2WithRSAEncryption' , 'id-dsa-with-sha1' , 'ecdsa-with-sha1' , 'ecdsa-with-sha224' , 'ecdsa-with-sha256' , 'ecdsa-with-sha384' or 'ecdsa-with-sha512'. Zero bytes if unavailable.

Table 100. SumList for REQDETAILS (continued)

Length	Value
1–byte length	Entry's Key type, max 16 bytes, one of 'RSA', 'DSA', 'BPECC' or 'NISTECC'. Zero bytes if unavailable.
1–byte length	Entry's Public Key size in bits, expressed as a decimal character string (EBCDIC, for example "1024"). Zero bytes if unavailable

Additionally, the REQDETAILS function returns GENCERT style certificate field name/value pairs in the CertPlist area. This is the list of fields that may be returned. They are also the fields that may be modified by function MODIFYREQS. The fields and their values are conditionally present, depending on the values of the original GENCERT request. Multiple OrgUnits and HostIdMaps are returned in the order they were originally specified. Fields other than OrgUnit and HostIdMap are not returned in any specific order. Like GENCERT, the CertPlist returned is a list of ordered triplets which consists of name, length and data value. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks, the length field is a binary four byte value, which qualifies the length of the data item.

Note: All data values are EBCDIC character data unless otherwise indicated. Also, NotBefore and NotAfter are replaced with StartDate and EndDate.

Table 101. CertPlist for REQDETAILS

Field name	Max length	Description
SerialNumber	64 bytes	Serial number of the subject device or registration number of the subject.
UnstructAddr	64 bytes	Unstructured address of the subject device.
UnstructName	64 bytes	Unstructured name of the subject device.
EmailAddr	64 bytes	Subject's email address for distinguished name EMAIL= attribute.
Mail	64 bytes	Subject's email address for distinguished name MAIL= attribute.
DNQualifier	64 bytes	Subject's Distinguished Name Qualifier.
Uid	64 bytes	Subject's login ID.
CommonName	64 bytes	Subject's common name.
Title	64 bytes	Subject's title.
DomainName	64 bytes	Subject's Domain Name containing all the domain components.
OrgUnit	64 bytes	Subject's Organization Unit. Note this field may be repeated.
Org	64 bytes	Subject's Organization.
Street	64 bytes	Subject's street address.
Locality	64 bytes	Subject's City or Locality.
StateProv	64 bytes	Subject's State or Province.
PostalCode	64 bytes	Subject's zip code or postal code.
Country	2 bytes	Subject's Country.

Table 101. CertPlist for REQDETAILS (continued)

Field name	Max length	Description
KeyUsage	20-bytes	KeyUsage extension entry. One of 'handshake' | 'dataencrypt' | 'certsign' | 'docsign' | 'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign (not case sensitive, no quotes). Note this field may be repeated.
ExtKeyUsage	20 bytes	One of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' |'mssmartcardlogon'. Note this field may be repeated.
StartDate	10 bytes	Data certificate becomes valid in YYYY/MM/DD form.
EndDate	10 bytes	Last date that the certificate is valid in YYYY/MM/DD form.
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternative name extension. Note that this field may be repeated.
AltURI	255 bytes	Uniform Resource Identifier for subject alternative name extension. Note that this field may be repeated.
AltEmail	100 bytes	Email address for subject alternative name extension. Note that this field may be repeated.
AltDomain	100 bytes	Domain Name for subject alternative name extension. Note that this field may be repeated.
AltOther	255 bytes	Other Name for subject alternative name extension. Note this field may be repeated.
HostIdMap	100 bytes	HostIdMapping extension entry. Note this field may be repeated.
AutoRenew	1 byte	Indicates whether the automatic renewal of certificates is enabled. Either 'Y' or 'N' (case sensitive, no quotes).
CustomExt	1024 bytes	Customized extension in the form of a comma-separated four-part string. The first part is the OID of the extension, the second part is the critical flag – 'C'(critical) or 'N'(non-critical) , the third part is the encode type – 'INT'(integer in printable hexadecimal format), 'IA5'(IA5 string), 'PRT'(printable string), 'BMP'(BMP string) , 'OCT'(Octet string) or 'UTF'(UTF8 string), the last part is the value. Note that this field may be repeated.
BusinessCat	64 bytes	Subject's business category. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.

Table 101. CertPlist for REQDETAILS (continued)

Field name	Max length	Description
JurCountry	2 bytes	Subject's Jurisdiction Country. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurStateProv	64 bytes	Subject's Jurisdiction State/Province. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurLocality	64 bytes	Subject's Jurisdiction Locality. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.

Table 102. Function_parmlist for MODIFYREQS

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' MODREQS '
Action	4-byte value	In	4-byte binary value indicating the action to take against the requests. • X'00000001' - Approve with possible modifications as specified below • X'00000002' - Reject • X'00000003' -Delete from request database
Comment	Address of	In	Points to a 65-byte area, in which the first byte will contain the actual length of the comment associated with this action. If the first byte is zero (x'00'), no comment will be recorded.
CertIdsLen	4-byte length	In/Out	Describes the length in bytes of the input certificate request ID list. May be modified by RACF on output if a smaller list is being returned.
CertIds	Address of	In/Out	Points to an area containing 1 or more certificate request Ids that are to be modified by this request. Each certificate request ID occupies a maximum of 57 bytes, in which the first byte will contain the actual length of the certificate request ID. If any requests cannot be modified because their states changed, RACF will return a shortened list containing those request Ids that couldn't be modified .
CertPlistLen	4-byte length	In	Describes the length in bytes of the certificate modification plist. A zero indicates no modification plist.

Table 102. Function_parmlist for MODIFYREQS (continued)

Field	Attributes	Usage	Description
CertPlist	Address of	In	Points to the area which is the certificate modification parameter list. This area maps the specific name, length, address/data values which are used to replace the existing values in the certificate request. The format is the same as the Certificate Request Plist defined under GENCERT (DiagInfo must be the first field.), except that the modifiable fields are those listed below. The certificate modification plist is optional and is valid for the "Approve" action only and only when the CertIds list contains exactly one Certificate ID. For all other cases, it is ignored. If no modification plist is specified for "Approve", the request is approved as is. See usage notes, 23 on page 307, for a description of the processing that performed when a modification plist is specified.
ErrListLen	4-byte length	In/Out	The 4-byte length of the pre-allocated storage for the ErrList parameter. This value may be modified by RACF on output to reflect the total length of the data returned in the ErrList parameter. If the value is zero upon input, the ErrList parameter will be ignored. The minimum Parmlist_Version for the use of this parameter is 1.
ErrList	Address of	In/Out	Points to a pre-allocated storage buffer that RACF will use to return error results for requests that could not be modified. On output, each error result will occupy a maximum of 101 bytes; a 1-byte length field, followed by up to 100 bytes of error description text. Each error result corresponds to a certificate request ID returned in the CertIds parameter. This parameter is ignored if the ErrListLen parameter is set to a value of zero. The minimum Parmlist_version for the use of this parameter is 1

The MODIFYREQS modification plist (CertPlist) Structure. Like GENCERT, the CertPlist is a list of ordered triplets which consists of name, length and data value. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks, the length field is a binary four byte value, which qualifies the length of the data item. Note that all data values are EBCDIC character data unless otherwise indicated. Note that NotBefore and NotAfter are replaced with StartDate and EndDate. See GENCERT for more information about the other individual fields

Table 103. CertPlist for MODIFYREQS

Field name	Max length	Description
DiagInfo	80 bytes (exactly)	Diagnostic information area. Must be first field in the CertPlist. For certificate generation warnings and errors, RACF will update this field with diagnostic information. The length will be updated as well. Required field.
SerialNumber	64 bytes	Serial number of the subject device. Optional. No default.
UnstructAddr	64 bytes	Unstructured address of the subject device. Optional. No default.
UnstructName	64 bytes	Unstructured name of the subject device. Optional. No default.
EmailAddr	64 bytes	Subject's email address for distinguished name EMAIL= attribute. Optional. No default.
Mail or Email	64 bytes	Subject's email address for distinguished name MAIL= attribute. Optional. No default. (Field name "Email" is deprecated, use "Mail")
DNQualifier	64 bytes	Subject's Distinguished Name Qualifier. Optional. No default.
Uid	64 bytes	Subject's login ID. Optional. No default.
CommonName	64 bytes	Subject's common name. Optional.
Title	64 bytes	Subject's title. Optional.
DomainName	64 bytes	Subject's Domain Name containing all the domain components in the form of *domain-component-1.domain-component-2.domain-component-3...domain-component-n*. Optional. No default.
OrgUnit	64 bytes	Subject's Organizational Unit. Note this field may be repeated. Optional.
Org	64 bytes	Subject's Organization. Optional.
Street	64 bytes	Subject's street address. Optional
Locality	64 bytes	Subject's City or Locality. Optional.
StateProv	64 bytes	Subject's State or Province. Optional.
PostalCode	64 bytes	Subject's Zip or postal code. Optional.
Country	2 bytes	Subject's Country. Optional.
KeyUsage	20 bytes	KeyUsage extension entry. One of 'handshake' | 'dataencrypt' | 'certsign' | 'docsign' 'digitalsignature' | 'digitalsig' | 'nonrepudiation' | 'keyencipherment' | 'keyenciph' | 'keyencrypt' | 'dataencipherment' | 'dataenciph' | 'keyagreement' | 'keyagree' | 'keycertsign' | 'crlsign' (not case sensitive, no quotes). Note this field may be repeated. Optional.
ExtKeyUsage	20 bytes	One of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' (not case sensitive, no quotes). Note this field may be repeated to request multiple usages. Optional, no default.

Table 103. CertPlist for MODIFYREQS (continued)

Field name	Max length	Description
StartDate	10 bytes	Date certificate becomes valid in YYYY/MM/DD form. Must be a valid date within the range 1970/01/01 through 9997/12/31. Required.
EndDate	10 bytes	Last date that the certificate is valid in YYYY/MM/DD form. Must be a valid date within the range of today through 9997/12/31 and must not be before StartDate. Required.
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternative name extension. Optional. There is no default. Note that this field may be repeated.
AltURI	255 bytes	Uniform Resource Identifier for subject alternative name extension. Optional. There is no default. Note that this field may be repeated.
AltEmail	100 bytes	Email address for subject alternative name extension. Optional. There is no default. Note that this field may be repeated.
AltDomain	100 bytes	Domain Name for subject alternative name extension. Optional. There is no default. Note that this field may be repeated.
AltOther	255 bytes	Other Name for subject alternative name extension. Optional. There is no default. Note this field may be repeated.
HostIdMap	100 bytes	HostIdMapping extension entry. Note this field may be repeated. Optional.
CertPolicies	32 bytes	Blank separated array of non-repeating policy numbers, range 1 - 99, for the CertificatePolicies extension. These correspond to the policy numbers defined during PKI Services configuration. Each value must be one or two digits with no leading zeros. Optional, no default.
AuthInfoAcc	255 bytes	A comma separated two part string specifying information used to create the AuthorityInfoAccess extension. The two part string identifies the accessMethod and accessLocation. The accessMethod is one of 'OCSP' | 'IdentrusOCSP' (not case sensitive, no quotes). The accessLocation is a URI in the form URI=access-url or URL=access-url. Optional, no default. Note this field may be repeated.
Critical	32 bytes	Name of a certificate extension to be marked critical. One of 'BasicConstraints' | 'KeyUsage' | 'ExtKeyUsage' | 'SubjectAltName' | 'AltEmail' | 'AltIPAddr' | 'AltDomain' | 'AltURI' | 'HostIdMappings' | 'HostIdMap' | 'CertificatePolicies' | 'CertPolicies' (not case sensitive, no quotes). Optional. The BasicConstraints and KeyUsage extensions are always marked critical even if not specified. Note this field may be repeated.
AutoRenew	1 byte	Indicates whether the automatic renewal of certificates is enabled. Either 'Y' or 'N' (not case sensitive, no quotes). Optional. There is no default.

Table 103. CertPlist for MODIFYREQS (continued)

Field name	Max length	Description
CustomExt	1024 bytes	Customized extension in the form of a comma-separated four-part string. The first part is the OID of the extension, the second part is the critical flag – 'C'(critical) or 'N'(non-critical) , the third part is the encode type – 'INT'(integer), 'IA5'(IA5 string), 'PRT'(printable string), 'BMP'(BMP string) , 'OCT'(Octet string) or 'UTF'(UTF8 string), the last part is the value. Note 1: The value specified for the INT type is a string of printable hexadecimal characters. If the number of characters is odd, the high order bit of the first character is propagated in the encoded value. Note 2: The critical flag and the encode type are not case sensitive. 'C'(critical) is allowed only if KeySize is not specified. Optional. No default. Only valid with PKI Services requests. Note that this field may be repeated.

Table 104. Function_parmlist for QUERYCERTS

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' QUERYCTS '
ResultsListLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the Results List area on input to QUERYCERTS. RACF will update this value with the actual length of the data returned. If the storage area, as specified by the Results List address is too small, RACF sets a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
ResultsList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the ResultsListLen), this service fails the request, and updates the ResultsListLen field to indicate the actual storage required to store the data. Also, see Table 105 on page 281.
SerialNum	Address of	In	Points to a 17-byte area, in which the first byte will contain the actual length of the input certificate serial number that will be used as a starting point for this query. Only certificates located after this certificate will be returned. If the first byte is zero (x'00'), the query will start with the first request. The serial number is in printable EBCDIC (HEX) form for example, "01A6",

Table 104. Function_parmlist for QUERYCERTS (continued)

Field	Attributes	Usage	Description
NumEntries	4-byte numeric	In/Out	Input value indicating the maximum number of entries that should be returned in the ResultsList area. Zero indicates no limit. Updated to indicate the number of entries actually returned.
CriteriaStatus	4-byte numeric	In	Value indicating the certificate status to be used as search criteria. • X'00000000' - return all issued certificates, • X'00000001' - return revoked certificates only, • X'00000002' -return expired certificates only, • X'00000003' - return non-expired, non-revoked certificates only, that is, active certificates, • X'00000004' - return non-expired revoked or suspended certificates only, that is, CRL certificates, • X'00000005' - return suspended certificates only, • X'00000006' - return active certificates enabled for auto renewal, • X'00000007' - return active certificates capable for auto renewal but disabled. • X'00000008' - return active certificates that cannot be renewed due to a change of email address.
CriteriaDays	4-byte numeric	In	Value indicating the recent activity time period or the certificate expiry period to be used as additional search criteria. A positive value indicates the number of days in the past that should be scanned for certificates that have been created or modified. A negative value indicates the numbers of days in the future that should be scanned for certificates that will expire within that period. A negative value is valid only if the CriteriaStatus is X'00000003', X'00000006', X'00000007', or X'00000008'. If it is zero (x'00000000'), recent activity and certificate expiry period will not be used as additional search criteria.
CriteriaName	Address of	In	Points to a 33-byte area, in which the first byte will contain the actual length of the input requestor's name to be used as additional search criteria. If the first byte is zero (x'00'), the requestor's name will not be used as additional search

The QUERYCERTS function returns results in the ResultsList area provided by the caller. The ResultsList has the following format:

Table 105. ResultsList for QUERYCERTS

Length	Value
1-byte length	Entry 1's serial number in printable EBCDIC (HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry 1's requestor's name, max 32 bytes
1-byte length	Entry 1's subject's distinguished name, max 255 bytes
1-byte length	Entry 1's issuer's distinguished name, max 255 bytes
1-byte length	Entry 1's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry 1's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt' |'certsign' |'docsign' | 'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry 1's status, max 32 bytes, one of "Active", "Active, AutoRenew", "Active, AutoRenewDisabled", "Expired", "Revoked", "Suspended", "Revoked, Expired", or "Active, NotRenewable"
1-byte length	Entry 1's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 1's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 1's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry 1's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified")
1–byte length	Entry 1's KeyId, exactly 40 bytes (printable EBCDIC)
1-byte length	Entry 2's serial number in printable EBCDIC (HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry 2's requestor's name, max 32 bytes
1-byte length	Entry 2's subject's distinguished name, max 255 bytes
1-byte length	Entry 2's issuer's distinguished name, max 255 bytes
1-byte length	Entry 2's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry 2's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt' | 'certsign' |'docsign' |'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry 2's status, max 32 bytes, one of "Active", "Active, AutoRenew", "Active, AutoRenewDisabled", "Expired", "Revoked", "Suspended", "Revoked, Expired", or "Active, NotRenewable".
1-byte length	Entry 2's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry 2's last modified date in YYYY/MM/DD form, exactly 10 bytes

Table 105. ResultsList for QUERYCERTS (continued)

Length	Value
1-byte length	Entry 2's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry 2's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified")
1–byte length	Entry 2's KeyID, exactly 40 bytes (printable EBCDIC)

⋮

Length, continued	Value, continued
1-byte length	Entry n's serial number in printable EBCDIC(HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry n's requestor's name, max 32 bytes
1-byte length	Entry n's subject's distinguished name, max 255 bytes
1-byte length	Entry n's issuer's distinguished name, max 255 bytes
1-byte length	Entry n's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry n's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt' |'certsign' |'docsign' |'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry n's status, max 32 bytes, one of "Active", "Active, AutoRenew", "Active, AutoRenewDisabled", "Expired", "Revoked", "Suspended", "Revoked, Expired", or "Active, NotRenewable"
1-byte length	Entry n's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry n's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry n's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry n's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified")
1–byte length	Entry n's KeyId, exactly 40 bytes (printable EBCDIC)

Table 106. Function_parmlist for CERTDETAILS

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' CRTDTAIL'

Table 106. Function_parmlist for CERTDETAILS (continued)

Field	Attributes	Usage	Description
SumListLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the Summary List area on input to CERTDETAILS. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Summary List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
SumList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the SummaryListLen), this service fails the request, and updates the SummaryListLen field to indicate the actual storage required to store the data. Also, see Table 107 on page 284.
CertPlistLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the certificate generation plist area on input to CERTDETAILS. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Results List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
CertPlist	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the CertPlistLen), this service fails the request, and updates the CertPlistLen field to indicate the actual storage required to store the data.. This area maps some of the specific name, length, address/data values which were used when generating the original certificate request on GENCERT. Also, see Table 108 on page 285.
SerialNum	Address of	In	Points to a 17-byte area, in which the first byte will contain the actual length of the input certificate serial number from which details are to be extracted. The serial number is in printable EBCDIC (HEX) form for example, "01A6",

The CERTDETAILS function returns QUERYCERTS style summary data in the SumList area provided by the caller. The SumList has the following format:

Table 107. SumList for CERTDETAILS

Length	Value
1-byte length	Entry's serial number in printable EBCDIC(HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry's requestor's name, max 32 bytes
1-byte length	Entry's subject's distinguished name, max 255 bytes
1-byte length	Entry's issuer's distinguished name, max 255 bytes
1-byte length	Entry's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry's keyUsage value, max 64 bytes (one or more of 'handshake' |'dataencrypt' |'certsign' |'docsign' |'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")
1-byte length	Entry's status, max 32 bytes, one of "Active", "Active, AutoRenew", "Active, AutoRenewDisabled", "Expired", "Revoked", "Suspended", "Revoked, Expired", or "Active, NotRenewable"
1-byte length	Entry's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry's last action comment, max 64 bytes
1-byte length	Entry's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified")
1-byte length	Entry's pass phrase provided when the certificate request was made, max 32 bytes
1–byte length	Entry's KeyId, exactly 40 bytes (printable EBCDIC)
1–byte length	Entry’s certificate signature algorithm, max 32 bytes, one of 'sha-1WithRSAEncryption', 'sha-224WithRSAEncryption', 'sha-256WithRSAEncryption', 'sha-384WithRSAEncryption', 'sha-512WithRSAEncryption', 'md-5WithRSAEncryption', 'md-2WithRSAEncryption', 'id-dsa-with-sha1', 'ecdsa-with-sha1', 'ecdsa-with-sha224', 'ecdsa-with-sha256', 'ecdsa-with-sha384' or 'ecdsa-with-sha512'
1–byte length	Entry’s Key type, max 16 bytes, one of 'RSA', 'DSA', 'BPECC' or 'NISTECC'
1–byte length	Entry's Public Key size in bits, expressed as a decimal character string (EBCDIC, for example, "1024"). Zero bytes if unavailable

Additionally, the CERTDETAILS function returns GENCERT style certificate field name/value pairs in the CertPlist area. This is the list of fields that may be returned. The fields and their values are conditionally present, depending on the values of the original GENCERT request. Multiple HostIdMaps are returned in the order they were originally specified. Fields other than HostIdMap are not returned in any specific order. Like GENCERT, the CertPlist returned is a list of ordered triplets which consists of name, length and data value. The field name is a fixed 12-character field, case sensitive, left-aligned,

and padded with blanks, the length field is a binary four byte value, which qualifies the length of the data item. Note that all data values are EBCDIC character data unless otherwise indicated.

Table 108. CertPlist for CERTDETAILS

Field	Max length	Description
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternative name extension. Note that this field may be repeated.
AltURI	255 bytes	Uniform Resource Identifier for subject alternative name extension. Note that this field may be repeated.
AltEmail	100 bytes	Email address for subject alternative name extension. Note that this field may be repeated.
AltDomain	100 bytes	Domain Name for subject alternative name extension. Note that this field may be repeated.
AltOther	255 bytes	Other Name for subject alternative name extension. Note this field may be repeated.
HostIdMap	100 bytes	HostIdMapping extension entry. Note this field may be repeated.
CustomExt	1024 bytes	Customized extension in the form of a comma-separated four-part string. The first part is the OID of the extension. The second part is the critical flag – 'C'(critical) or 'N'(non-critical), the third part is the encode type – 'INT'(integer in printable hexadecimal format), 'IA5'(IA5 string), 'PRT'(printable string), 'BMP'(BMP string) , 'OCT'(Octet string) or 'UTF'(UTF8 string), the last part is the value. Note that this field may be repeated.

Table 109. Function_parmlist for MODIFYCERTS

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, 'MODCERTS'.
Action	4-byte value	In	4-byte binary value indicating the action to take with the certificates. • X'00000002' - Revoke • X'00000003' - Delete from issued certificate database • X'00000004' - Resume suspended certificate • X'00000005' - Disable auto renew • X'00000006' - Enable auto renew • X'00000007' - Change requestor email • X'00000008' - Create CRLs • X'00000009' - Post Certs

Table 109. Function_parmlist for MODIFYCERTS (continued)

Field	Attributes	Usage	Description
Comment	Address of	In	Points to a 65-byte area, in which the first byte will contain the actual length of the comment associated with this action. If the first byte is zero (x'00'), no comment will be recorded. If the action is X'00000008' (Create CRLs), this field is ignored.
SerialNumsLen	4-byte length	In/Out	Describes the length in bytes of the input certificate serial number list. May be modified by RACF on output if a smaller list is being returned. If the action is X'00000008' (Create CRLs), this field is ignored.
SerialNums	Address of	In/Out	Points to an area containing 1 or more certificate serial numbers that are to be modified by this request. Each occupies a maximum of 17 bytes, in which the first byte will contain the actual length of the certificate serial number. The serial number itself is in printable EBCDIC (HEX) form for example, "01A6", If any certificates cannot be modified, RACF will return a shortened list containing those serial numbers that couldn't be modified. The ErrList field will contain the corresponding error description. If the action is X'00000007' - Change requestor email, this field must contain only 1 serial number. If the action is X'00000008' (Create CRLs), this field is ignored.
Reason	4-byte value	In	4-byte binary value indicating the reason for the certificate revocation. • X'00000000' - No Reason • X'00000001' - User key was compromised • X'00000002' - CA key was compromised • X'00000003' - User changed affiliation • X'00000004' - Certificate was superseded • X'00000005' - Original use no longer valid • X'00000006' - Temporarily suspend Ignored for actions other than "Revoke".
RequestorEmail	Address of	In	Points to a 33-byte area in which the first byte will contain the actual length of the email address to be changed. Only valid with action X'00000007' - Change requestor email. The minimum Parmlist_Version for the use of this parameter is 1.

Table 109. Function_parmlist for MODIFYCERTS (continued)

Field	Attributes	Usage	Description
ErrListLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage for the Error List output area. May be modified by RACF on output to reflect the actual length of ErrList. If the action is X'00000008' (Create CRLs) or X'00000009' (Post Certs), this field is ignored. The minimum Parmlist_Version for the use of this parameter is 2.
ErrList	Address of	In/Out	Points to an area containing 1 or more error results when any of the input certificates cannot be modified. Each error result occupies a maximum of 100 bytes, in which the first byte is the actual length of the error description for the corresponding serial number returned in the SerialNums field. If the action is X'00000008' (Create CRLs) or X'00000009' (Post Certs), this field is ignored. The minimum Parmlist_Version for the use of this parameter is 2.

Table 110. Function_parmlist for VERIFY

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' VERIFY '.
SumListLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the Summary List area on input to VERIFY. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Summary List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
SumList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the verify if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the SummaryListLen) this service fails the request, and updates the SummaryListLen field to indicate the actual storage required to store the data. Also, see Table 111 on page 288.

Table 110. Function_parmlist for VERIFY (continued)

Field	Attributes	Usage	Description
CertPlistLen	4-byte length	In/Out	4-byte area which is the length of the pre-allocated storage of the certificate generation plist area on input to VERIFY. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Results List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
CertPlist	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the verify if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the CertPlistLen) this service fails the request, and updates the CertPlistLen field to indicate the actual storage required to store the data. This area maps some of the specific name, length, address/data values which were used when generating the original certificate request on GENCERT. Also, see Table 112 on page 289.
CertLen	4-byte length	In	4-byte area that is the length of the certificate contained in the Cert area on input to VERIFY.
Cert	Address of	In	The address of the storage area containing the X509 certificate or the PKCS #7 certificate chain to verify. This is base64 encoded DER.

The VERIFY function returns CERTDETAILS style summary data in the SumList area provided by the caller. The SumList has the following format:

Table 111. SumList for VERIFY

Length	Value
1-byte length	Entry's serial number in printable EBCDIC(HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry's requestor's name, max 32 bytes
1-byte length	Entry's subject's distinguished name, max 255 bytes
1-byte length	Entry's issuer's distinguished name, max 255 bytes
1-byte length	Entry's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry's keyUsage value, max 64 bytes (one or more of 'handshake' | 'dataencrypt' | 'certsign' | 'docsign' | 'digitalsig' | 'keyencrypt' | 'keyagree' | 'keycertsign' | 'crlsign' or "not specified")

Table 111. SumList for VERIFY (continued)

Length	Value
1-byte length	Entry's status, max 32 bytes, one of "Active", "Active, AutoRenew", "Active, AutoRenewDisabled", "Expired", "Suspended", "Revoked", "Revoked, Expired", or "Active, NotRenewable"
1-byte length	Entry's creation date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's last modified date in YYYY/MM/DD form, exactly 10 bytes
1-byte length	Entry's ApplData value from the GENCERT or REQCERT invocation, max 8 bytes
1-byte length	Entry's ExtKeyUsage value, max 255 bytes (one or more of 'serverauth' | 'clientauth' | 'codesigning' | 'emailprotection' | 'timestamping' | 'ocspsigning' | 'mssmartcardlogon' or "not specified")

Additionally, the VERIFY function returns GENCERT style certificate field name/value pairs in the CertPlist area. The fields and their values are conditionally present, depending on the values actually contained in the certificate. Fields are not returned in any specific order. Like GENCERT, the CertPlist is a list of ordered triplets which consists of name, length and data value. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks, the length field is a binary four byte value, which qualifies the length of the data item. Note that all data values are EBCDIC character data unless otherwise indicated.

Table 112. CertPlist for VERIFY

Field	Max length	Description
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternative name extension. Note that this field may be repeated.
AltURI	255 bytes	Uniform Resource Identifier for subject alternative name extension. Note that this field may be repeated.
AltEmail	100 bytes	Email address for subject alternative name extension. Note that this field may be repeated.
AltDomain	100 bytes	Domain Name for subject alternative name extension. Note that this field may be repeated.
AltOther	255 bytes	Other Name for subject alternative name extension. Note this field may be repeated.
HostIdMap	100 bytes	HostIdMapping extension entry. Note this field may be repeated.
CustomExt	1024 bytes	Customized extension in the form of a comma-separated four-part string. The first part is the OID of the extension. The second part is the critical flag – 'C'(critical) or 'N'(non-critical), the third part is the encode type – 'INT'(integer in printable hexadecimal format), 'IA5'(IA5 string), 'PRT'(printable string), 'BMP'(BMP string) , 'OCT'(Octet string) or 'UTF'(UTF8 string), the last part is the value. Note that this field may be repeated.

Table 113. Function_parmlist for REVOKE

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' REVOKE '
Reason	4-byte value	In	4-byte binary value indicating the reason for the certificate revocation, X'00000000' - No Reason, X'00000001' - User key was compromised, X'00000002' - CA key was compromised, X'00000003' - User changed affiliation, X'00000004' - Certificate was superseded, X'00000005' -Original use no longer valid, X'00000006' - Temporarily suspend
SerialNum	Address of	In	Points to a 17-byte area, in which the first byte will contain the actual length of the input certificate serial number for the certificate that is to be revoked . The serial number is in printable EBCDIC (HEX) form for example, "01A6",

Table 114. Function_parmlist for GENRENEW and REQRENEW

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' RENEW '
CertPlistLen	4-byte length	In	Describes the length in bytes of the certificate generation plist. A zero indicates no modification plist
CertPlist	Address of	In	The name of the area which is the renew request parameter list. This area maps the specific name, length, address/data values which are used in satisfying the certificate request for the specified user. Also, see Table 115 on page 291.
CertId	Address of	In/Out	Points to a 57-byte area, in which the first byte will contain the actual length on return of the certificate request ID. The storage address specified must be obtained by the caller, and freed by the caller. The returned certificate request ID is used to extract the completed certificate, if the request has been accepted.
SerialNum	Address of	In	Points to a 17-byte area, in which the first byte will contain the actual length of the input certificate serial number for the certificate that is to be renewed. The serial number is in printable EBCDIC (HEX) form for example, "01A6",

Here is the layout and supported fields for the RENEW CertPlist. Because most of the certificate information from the old certificate is reused for the new

certificates, very little new information can be specified in the RENEW CertPlist. Like GENCERT, the CertPlist is a list of ordered triplets that consists of name, length and data value. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks. The length field is a binary four byte value, which qualifies the length of the data item. Note that all data values are EBCDIC character data unless otherwise indicated. See GENCERT for more information about the other individual fields

Table 115. CertPlist for GENRENEW and REQRENEW

Field name	Max length	Description
DiagInfo	80 bytes (exactly)	Diagnostic information area. Must be first field in the CertPlist. For certificate generation warnings and errors, RACF will update this field with diagnostic information. The length will be updated as well. Required field.
PassPhrase	32 bytes	Value to be used for challenge/response when retrieving the certificate through function EXPORT. When renewing a certificate whose key pair was generated by PKI Services, the PassPhrase from the original certificate will be used if this field is not specified. Optional.
NotAfter	4 bytes	Number of days from today's date that the certificate expires. Range 1-9999. Validity checked by RACF. Optional. Default is 365. The start date of the validity period is set from the original certificate's start of validity.
NotifyEmail	64 bytes	Email address for notification purposes. When renewing a certificate whose key pair was generated by PKI Services, the specified value must not exceed 32 characters. If this field is not specified, the notification email address of the original certificate will be used. Optional.
CertPolicies	32 bytes	Blank separated array of non-repeating policy numbers. Range 1 - 99, for the CertificatePolicies extension. These correspond to the policy numbers defined during PKI Services configuration. Each value must be one or two digits with no leading zeros. Optional, no default.
AuthInfoAcc	255 bytes	A comma separated two part string specifying information used to create the AuthorityInfoAccess extension. The two part string identifies the accessMethod and accessLocation. The accessMethod is one of 'OCSP' | 'IdentrusOCSP' (not case sensitive, no quotes). The accessLocation is a URI in the form URI=access-url or URL=access-url. Optional, no default. Note this field may be repeated.
Critical	32 bytes	Name of a certificate extension to be marked critical. One of 'CertificatePolicies' | 'CertPolicies' (not case sensitive, no quotes). Optional.

Table 116. Function_parmlist for RESPOND

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example 'RESPOND'
RestLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the OCSP response area on input to RESPOND. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Response address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
Response	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the RESPOND function if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the ResLen), this service fails the request, and updates the ResLen field to indicate the actual storage required to store the data.
ReqLen	4-byte length	In	4-byte area that is the length of the request contained in the Request area on input to RESPOND.
Request	Address of	In	The address of the storage area containing the request to verify.

Table 117. Function_parmlist for SCEPREQ

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' SCEPREQ '
RestLen	4-byte length	In/Out	4-byte area that is the length of the pre-allocated storage of the SCEP response area on input to SCEPREQ. RACF will update this value with the actual length of the data returned. If the storage area, as specified by the Response address is too small, RACF sets a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
Response	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the SCEPREQ function if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the ResLen), this service fails the request, and updates the ResLen field to indicate the actual storage required to store the data.

Table 117. Function_parmlist for SCEPREQ (continued)

Field	Attributes	Usage	Description
ReqLen	4-byte length	In	4-byte area that is the length of the SCEP request contained in the Request area on input to SCEPREQ.
Request	Address of	In	The address of the storage area containing the SCEP request to verify.

Table 118. Function_parmlist for PREREGISTER

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example, ' PREREG '
PreregPlistLen	4-byte length	In	Describes the length in bytes of the preregistration plist
PreregPlist	Address of	In	The name of the area which is the preregistration parameter list. This area maps the specific name, length, address or data values which must match the values specified when the certificate request is received.
CertId	Address of	In/Out	Points to a 57-byte area, in which the first byte will contain the actual length on return of the certificate request ID. The storage address specified must be obtained by the caller, and freed by the caller. The returned certificate request ID is used to query the preregistration record, if the request has been accepted by RACF.

Like GENCERT, the Preregistration CertPlist is a list of ordered triplets which consists of name, length and data value. The data values provided on PREREGISTER must match the values specified when the certificate request is received, otherwise the requestor is considered unauthenticated. The field name is a fixed 12-character field, case sensitive, left-aligned, and padded with blanks, the length field is a binary 4-byte value, which qualifies the length of the data item.

Note: All data values are EBCDIC character data unless otherwise indicated. See GENCERT for more information about the other individual fields.

Table 119. CertPlist for PREREGISTER

Field name	Max length	Description
DiagInfo	80 bytes (exactly)	EyeCatcher to identify this request in virtual storage for diagnostic reasons. For certificate generation warnings and errors RACF will update this field with diagnostic information. The length will be updated as well. Required field. Must be first field in the CertPlist.
ClientName	64 bytes	Name of the person or device that is being preregistered. Must match either the CommonName or the UnstructName when the certificate request is received. Required field.

Table 119. CertPlist for PREREGISTER (continued)

Field name	Max length	Description
PassPhrase	32 bytes	Challenge/response value to be used for authenticating when the certificate request is received. Optional. No default.
SerialNumber	64 bytes	Serial number of the subject device. Optional. No default.
UnstructAddr	64 bytes	Unstructured address of the subject device. Optional. No default.
EmailAddr	64 bytes	Subject's email address for distinguished name EMAIL= attribute. Optional. No default.
Mail	64 bytes	Subject's email address for distinguished name MAIL= attribute. Optional. No default.
DNQualifier	64 bytes	Subject's Distinguished Name Qualifier. Optional. No default.
Uid	64 bytes	Subject's login ID. Optional. No default.
Title	64 bytes	Subject's Title. Optional. No default.
DomainName	64 bytes	Subject's Domain Name containing all the domain components in the form of *domain-component-1.domain-component-2.domain-component-3...domain-component-n*. Optional. No default.
OrgUnit	64 bytes	Subject's Organizational Unit. Note this field may be repeated, RACF concatenates in the order of appearance to construct the hierarchy of organizational units. Optional. No default.
Org	64 bytes	Subject's Organization. Optional. Optional. No default.
Street	64 bytes	Subject's Street Address. Optional. No default.
Locality	64 bytes	Subject's City or Locality. Optional. No default.
StateProv	64 bytes	Subject's State/Province. Optional. No default.
PostalCode	64 bytes	Subject's Zip code or Postal Code. Optional. No default.
Country	2 bytes	Subject's Country. Optional. No default.
AltIPAddr	45 bytes	IP address in IPv4 or IPv6 format for subject alternate name extension. Optional. No default. Note that this field may be repeated.
AltURI	255 bytes	Uniform Resource Identifier for subject alternate name extension. Optional. No default. Note that this field may be repeated.
AltEmail	100 bytes	Email address for subject alternate name extension. Optional. No default. Note that this field may be repeated.
AltDomain	100 bytes	Domain Name for subject alternate name extension.Optional. No default. Note that this field may be repeated.
AltOther	255 bytes	Other Name for subject alternate name extension. Optional. No default. Note this field may be repeated.

Table 119. CertPlist for PREREGISTER (continued)

Field name	Max length	Description
BusinessCat	64 bytes	Subject's business category. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurCountry	2 bytes	Subject's Jurisdiction Country. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurStateProv	64 bytes	Subject's Jurisdiction State/Province. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.
JurLocality	64 bytes	Subject's Jurisdiction Locality. Optional. No default. Only valid with PKI Services requests. **Note:** It is recommended that this field be used only in requests for Extended Verification certificates.

Table 120. Function_parmlist for QRECOVER

Field	Attributes	Usage	Description
Eyecatcher	8 characters	In	Eyecatcher, 8 characters left-aligned blank filled. Actual value set by invoker, for example 'QRECOVER'.
ResultsListLen	4-byte length	In/Out	4-byte area which is the length of the pre-allocated storage of the Results List area on input to QRECOVER. RACF will update this value with the actual length of the data returned. If the storage area as specified by the Results List address is too small, RACF will set a failing return/reason code and update the length field to the size required. The caller must allocate a larger area.
ResultsList	Address of	In/Out	The address of the storage area in which the R_PKIServ service stores the results of the query if the service was able to successfully retrieve the data. If the caller has supplied an area which is too small, (based in the ResultsListLen), this service fails the request, and updates the ResultsListLen field to indicate the actual storage required to store the data.
NumEntries	4-byte numeric	In/Out	Input value indicating the maximum number of entries that should be returned in the ResultsList area. Zero indicates no limit. Updated to indicate the number of entries actually returned.

Table 120. Function_parmlist for QRECOVER (continued)

Field	Attributes	Usage	Description
CriteriaName	Address of	In	Points to a 33-byte area, in which the first byte will contain the actual length of the input requestor's email address to be used as a search criterion.
CriteriaPass	Address of	In	Points to a 33-byte area, in which the first byte will contain the actual length of the input pass phrase to be used as a search criterion.

The QRECOVER function returns results in the ResultsList area provided by the caller. The results list has the following format:

Table 121. ResultsList for QRECOVER

Length	Value
1-byte length	Entry 1's serial number in printable EBCDIC (HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry 1's subject's distinguished name, max 255 bytes
1-byte length	Entry 1's issuer's distinguished name, max 255 bytes
1-byte length	Entry 1's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry 1's Pass phrase provided when the certificate request was made, max 32 bytes
1-byte length	Entry 1's KeyId, exactly 40 bytes (printable EBCDIC)
1-byte length	Entry 2's serial number in printable EBCDIC (HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry 2's subject's distinguished name, max 255 bytes
1-byte length	Entry 2's issuer's distinguished name, max 255 bytes
1-byte length	Entry 2's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry 2's Pass phrase provided when the certificate request was made, max 32 bytes
1-byte length	Entry 2's KeyId, exactly 40 bytes (printable EBCDIC)

⋮

Length, continued	Value, continued
1-byte length	Entry n's serial number in printable EBCDIC (HEX) form for example, "01A6", max 16 bytes
1-byte length	Entry n's subject's distinguished name, max 255 bytes
1-byte length	Entry n's issuer's distinguished name, max 255 bytes
1-byte length	Entry n's validity period in local time. Format YYYY/MM/DD HH:MM:SS - YYYY/MM/DD HH:MM:SS. Exactly 41 bytes
1-byte length	Entry n''s Pass phrase provided when the certificate request was made, max 32 bytes
1-byte length	Entry 1's KeyId, exactly 40 bytes (printable EBCDIC)

CA_domain
The name of an optional 9-byte input area that consists of a 1-byte length field followed by up to 8 characters from the following character set: the alphanumerics (a-z, A-Z, 0-9) and the hyphen ('-'). In addition, the leftmost character must not be a digit or the hyphen. The value is the not case-sensitive domain name of the PKI Services certificate authority instance to be invoked. If Number_parameters is less than 6, CA_domain is null, or the length byte is 0, then the default instance of PKI Services will be invoked. This field is ignored for SAF requests.

Return and reason codes

R_PKIServ may return the following values in the reason and return code parameters:

Table 122. Return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	Successful completion
4	0	0	RACF not installed
8	8	4	A parameter list error has been detected. See Usage Notes for further details
8	8	8	The caller of this service has not been RACF authorized to use this callable service
8	8	12	An internal error has occurred during RACF processing of the requested function
8	8	16	Unable to establish a recovery environment
8	8	20	Function code specified is not defined
8	8	24	Parameter list version specified is not supported
8	8	28	Certificate generation provider not available
8	8	32	Incorrect value specified for CA_domain. Either the length is greater than 8 or the value contains characters that are not valid
8	12	xx	Certificate generation provider internal error. Reason code is the reason code from provider

Reason and return code parameters specific to function GENCERT and REQCERT:

SAF return code	RACF return code	RACF reason code	Explanation
4	4	0	Successful completion. However, notification of the TID through email is unsuccessful.
8	8	40	CertPlist has an incorrect length
8	8	44	CertPlist DiagInfo field missing or has an incorrect length

SAF return code	RACF return code	RACF reason code	Explanation
8	8	48	Incorrect field name specified in CertPlist. The field name is either unknown or not supported by this certificate generation provider
8	8	52	Incorrect field value specified in CertPlist
8	8	56	Required field is missing from the CertPlist
8	8	60	Certificate generation provider input or environment error
8	8	64	PKCS#11 Token Service encountered an error.
8	8	68	Notification form is not set up correctly in the case of key generation.
8	8	76	Conflicting field names specified in CertPlist.

Reason and return code parameters specific to function EXPORT:

SAF return code	RACF return rode	RACF reason code	Explanation
0	0	0	Successful completion. If the “PKICACERT” CertId was specified, then the returned certificate package contains just the X.509 CA certificate
0	0	1	Successful completion. The “PKICACERT” CertId was specified. The returned certificate package is the RA/CA PKCS #7 certificate chain
0	0	2	Successful completion. If serial number was specified in CertId, the returned certificate package contains the PKCS12 package.
8	8	40	CertAnchor area missing
8	8	44	CertAnchor area too small
8	8	48	Incorrect CertID (transaction ID or serial number) specified
8	8	52	Incorrect PassPhrase specified
8	8	56	Request is still pending approval or has yet to be issued
8	8	60	Request has been rejected by the administrator
8	8	64	PKCS#11 Token Service has encountered an error.
8	8	68	Incorrect KeyId specified or private key object in TKDS not found.
8	8	72	“PKICACERT” CertId specified, but SCEP is disabled

Reason and return code parameters specific to function QUERYREQS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Results list area missing.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	44	Results list area too small.
8	8	48	Incorrect CertId specified.
8	8	56	Incorrect status criteria specified.
8	8	60	No requests satisfy the input criteria.

Reason and return code parameters specific to function REQDETAILS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Summary list or CertPlist area missing.
8	8	44	Summary list or CertPlist area too small.
8	8	48	Incorrect CertId specified.
8	8	52	Success, however name fields not returned in CertPlist.
8	8	64	Not authorized to display the details of the request under a specific template.

Reason and return code parameters specific to function MODIFYREQS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	CertPlist has an incorrect length.
8	8	44	CertPlist DiagInfo field missing or has an incorrect length.
8	8	48	Incorrect field name specified in CertPlist.
8	8	52	Incorrect field value specified in CertPlist.
8	8	56	Required field missing from CertPlist.
8	8	60	Certificate generation input or environment error.
8	8	64	CertIds has an incorrect length or value.
8	8	68	Incorrect Action specified.
8	8	72	One or more requests could not be modified because of a state change. CertIds contains the certificate request Ids that could not be modified. ErrList contains the corresponding error description.
8	8	96	One or more requests could not be modified because the user is not authorized to perform the action on the requests under that template. CertIds contains the certificate request Ids that could not be modified. ErrList contains the corresponding error description.

Reason and return code parameters specific to function QUERYCERTS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Results list area missing.
8	8	44	Results list area too small.
8	8	48	Incorrect SerialNum specified.
8	8	56	Incorrect status criteria specified.
8	8	60	No certificates satisfy the input criteria.

Reason and return code parameters specific to function CERTDETAILS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Summary list or CertPlist area missing.
8	8	44	Summary list or CertPlist area too small.
8	8	48	Incorrect SerialNum specified.
8	8	64	Not authorized to display the details of the certificate under a specific template.

Reason and return code parameters specific to function MODIFYCERTS:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Incorrect Reason specified.
8	8	52	Incorrect field value specified in RequestorEmail.
8	8	64	SerialNums has an incorrect length or value.
8	8	68	Incorrect Action specified.
8	8	72	One or more certificates cannot be modified because of a state change. SerialNums contains the certificate serial numbers that cannot be modified. ErrList contains the corresponding error description.
8	8	76	One or more certificates cannot be set up for automatic renewal. SerialNums contains the certificate serial numbers that could not be set up for automatic renewal. ErrList contains the corresponding error description.
8	8	80	RequestorEmail could not be modified because the key of the certificate was not generated by PKI Services.
8	8	84	More than one serial number is requested for RequestorEmail change.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	88	One or more certificates could not be deleted from the TKDS although they were deleted from the ICL. SerialNums contains the certificate serial numbers that could not be deleted from the TKDS. ErrList contains the corresponding error description.
8	8	92	CRL or Certificate posting is not configured.
8	8	96	One or more certificates could not be modified because the calling userid is not authorized to perform the modify action for the given template (if applicable) and CA domain. The SerialNums parameter contains the serial number of the certificates that could not be modified. The ErrList parameter contains the corresponding error description for each of the failing serial numbers.

Reason and return code parameters specific to function VERIFY:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Summary list or CertPlist area missing.
8	8	44	Summary list or CertPlist area too small.
8	8	64	Incorrect certificate specified.

Reason and return code parameters specific to function REVOKE:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Incorrect Reason specified.
8	8	64	SerialNum has an incorrect length or value.
8	8	72	The certificate could not be revoked because of a state change.

Reason and return code parameters specific to functions GENRENEW and REQRENEW:

SAF return code	RACF return code	RACF reason code	Explanation
4	4	0	Successful completion. But notification of the TID through email was unsuccessful.
8	8	40	CertPlist has an incorrect length.
8	8	44	CertPlist DiagInfo field missing or has an incorrect length.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	48	Incorrect field name specified in CertPlist. The field name is either unknown or not supported by this certificate generation provider.
8	8	52	Incorrect field value specified in CertPlist.
8	8	56	Required field missing from CertPlist.
8	8	60	Certificate generation input or environment error.
8	8	64	SerialNum has an incorrect length or value.
8	8	68	Notification form is not set up correctly for key generation.
8	8	72	The certificate could not be renewed because of a state change.
8	8	76	Conflicting field names specified in CertPlist.
8	8	80	The certificate could not be renewed because the requester's email has changed.

Reason and return code parameters specific to function RESPOND:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Response area missing.
8	8	44	Response area too small.
8	8	64	Incorrect request specified.
8	8	72	Responder disabled.

Reason and return code parameters specific to function SCEPREQ:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Response area missing.
8	8	44	Response area too small.
8	8	64	Incorrect request specified.
8	8	72	SCEP disabled.

Reason and return code parameters specific to function PREREGISTER:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	CertPlist has an incorrect length
8	8	44	CertPlist DiagInfo field missing or has an incorrect length

SAF return code	RACF return code	RACF reason code	Explanation
8	8	48	Incorrect field name specified in CertPlist. The field name is either unknown or not supported by this certificate generation provider
8	8	52	Incorrect field value specified in CertPlist
8	8	56	Required field missing from CertPlist
8	8	60	Certificate generation provider input or environment error
8	8	72	Client already preregistered
8	8	76	Not authorized to preregister a client under a specific template.

Reason and return code parameters specific to function QRECOVER:

SAF return code	RACF return code	RACF reason code	Explanation
8	8	40	Results list area missing.
8	8	44	Results list area too small.
8	8	48	Incorrect requester's email address specified.
8	8	52	Incorrect pass phrase specified.
8	8	56	Notify form is not set up correctly.
8	8	60	No certificates satisfy the input criteria.
8	8	64	Notify email cannot be sent.

Usage notes

1. This service is intended for use by z/OS application servers, to programmatically request the fulfillment of an X.509 V.3 certificate request.
2. An audit record will be created as a result of invoking this service which will indicate the success or failure of the attempt.
3. For GENCERT, the certificate generation provider is designated by the first four characters of the CertPlist SignWith value, SAF: for SAF requests, PKI: for PKI Services requests. For EXPORT, the caller designates the certificate generation provider indirectly by providing the Certificate ID returned by the provider. For all other functions, PKI Services is used exclusively. There is no SAF equivalent for these functions.
4. TheCertPlist for the PREREGISTER, GENCERT REQCERT, REQDETAILS, MODIFYREQS, VERIFY, REQRENEW, GENRENEW, and CERTDETAILS functions consists of triplets which consist of field name, field length, and data. The field name is a fixed field, 12 characters in length, and the field name must be left-aligned, and padded with blanks. The data length is also a fixed width field of 4 bytes which contain an integer which represents the length of the following field data.
5. The R_PKIServ service requires the caller to preallocate the 57-byte storage area that will hold the certificate ID returned on a successful GENCERT, REQRENEW, GENRENEW, or REQCERT. When successful, RACF will update

the first byte with the actual length of the CertID. The entire 57 areas must be provided for EXPORT even if the actual CertId is smaller than that.

6. The R_PKIServ service requires the caller to preallocate the storage that will hold the certificate being extracted through the EXPORT function code. On successful certificate retrieval, RACF will update the CertAnchorLen field with the actual length of the certificate. If the storage area is too small to hold the certificate, then RACF will fail the request and update the CertAnchorLen field in the EXPORT request-specific parameter list as supplied by the caller of this service. The caller is responsible for releasing and obtaining a new area of virtual storage that is the size as specified by RACF, and retrying the EXPORT operation.

   Note: The retry might have to be performed more than once.
7. The R_PKIServ service requires the caller to preallocate the storage that will hold the results list being retrieved through the QUERYREQS, QUERYCERTS, and QRECOVER function codes. On success, RACF will update the ResultsListLen field with the actual length of the data returned. If the storage area is too small to hold the data, then RACF will fail the request and update the ResultsListLen field in the request-specific parameter list as supplied by the caller of this service. The caller is responsible for releasing and obtaining a new area of virtual storage that is the size as specified by RACF, and retrying the operation.
8. The R_PKIServ service requires the caller to preallocate the storage that will hold the summary list being retrieved through the VERIFY, REQDETAILS, and CERTDETAILS function codes. On success, RACF will update the SumListLen field with the actual length of the data returned. If the storage area is too small to hold the data, then RACF will fail the request and update the SumListLen field in the request-specific parameter list as supplied by the caller of this service. The caller is responsible for releasing and obtaining a new area of virtual storage that is the size as specified by RACF, and retrying the operation.
9. The R_PKIServ service requires the caller to preallocate the storage that will hold the CertPlist being retrieved through the VERIFY, REQDETAILS, and CERTDETAILS function codes. On success, RACF will update the CertPListLen field with the actual length of the data returned. If the storage area is too small to hold the data, then RACF will fail the request and update the CertPListLen field in the request-specific parameter list as supplied by the caller of this service. The caller is responsible for releasing and obtaining a new area of virtual storage that is the size as specified by RACF, and retrying the operation.
10. The actual values for the eyecatchers in the function-specific parameters lists and the DiagInfo field in the CertPlist for GENCERT, MODIFYREQS, GENRENEW, REQRENEW, and REQCERT invocations are determined by the caller.
11. For PREREGISTER, GENCERT, REQCERT, GENRENEW, REQRENEW, and MODIFYREQS CertPlist field errors (reason codes 48, 52, 56, and 76), RACF will update the DiagInfo field with the name of the field in error. The length will also be updated.
12. For MODIFYREQS and MODIFYCERTS state change errors (reason code 72), RACF will update the CertIds or SerialNums field with the list of Certificate IDs or Serial Numbers that could not be updated. The length field will also be updated to reflect the size of the data being returned.
13. For GENCERT and REQCERT PKI Services requests, the special user IDs that start with lowercase 'irr' may not be specified for the CertPlist field UserId.

14. For PREREGISTER, GENCERT, REQCERT, GENRENEW, REQRENEW, and MODIFYREQS certificate generation errors (reason code 60) RACF will update the DiagInfo field with a product-specific diagnostic message. For SAF requests, the message will have the following format: error-description (message-ID), where message-ID is the RACDCERT error message ID that is closely related to this error:

```
No matching certificate was found for "SignWith" (IRRD107I)
"PublicKey" encoding does not have a valid signature (IRRD112I)
"PublicKey" encoding is not valid (IRRD104I)
"PublicKey" encoding contains an unsupport encryption algorithm (IRRD118I)
"PublicKey" extension not permitted for CERTAUTH certificate (IRRD126I)
"Label" specified is already in use (IRRD111I)
"SignWith" requires a certificate with an associated private key (IRRD128I)
Certificate cannot be added.  Serial number for this CA already in use (IRRD109I)
SignWith key is an ICSF key.  ICSF is not operational (IRRD135I)
Subject's name exceeds the maximum allowed characters, which is 255 (IRRD131I)
```

Additionally, for successful certificate generation (reason code 0), RACF may also update the DiagInfo field with one of the following informational diagnostic messages:

```
Inconsistency detected.  Signing certificate is not trusted (IRRD132I)
Inconsistency detected.  Signing certificate's date range is incorrect (IRRD113I)
```

RACF may also issue its own diagnostic messages when acting as an RA for PKI Services.

For further information see *z/OS Security Server RACF Command Language Reference* and *z/OS Security Server RACF Messages and Codes*. It is expected that other security products that may be installed in place of RACF have their own product-specific diagnostic data.

15. For GENCERT, REQCERT, PREREGISTER, and MODIFYREQS, RACF forms the subject's distinguished name in the following order:
    - SerialNumber
    - UnstructAddr
    - UnstructName
    - EmailAddr
    - Mail (formally Email)
    - DNQualifier
    - Uid
    - CommonName
    - Title
    - DomainName
    - OrgUnits (in the order that they appear in the CertPlist)
    - Business Category
    - Org
    - Jurisdiction Locality
    - Jurisdiction StateProv
    - Jurisdiction Country
    - Street
    - Locality
    - StateProv

- PostalCode
- Country

Except as noted below, only those portions of the name specified in the CertPlist will appear in the certificate. For GENCERT and REQCERT, if no name fields are specified in the CertPlist, the name is taken directly from the PublicKey field. For SCEPREQ, the name is always taken from the initial PKCS #10 request.

16. Different certificate generation providers may produce certificates with different extension values. To determine what certificate extensions will be created by a given provider, see the provider's supporting documentation, *z/OS Security Server RACF Security Administrator's Guide*, and *z/OS Security Server RACF Command Language Reference*, or *z/OS Cryptographic Services PKI Services Guide and Reference*.
17. For GENCERT and REQCERT, if CommonName is specified with a null value (length 0), RACF will use the PGMRNAME field from the RACF user profile as determined by the UserID field for this request. If PGMRNAME is null, the common name will be of the form of RACF User ID:<*user's-racf-identity*>, for example RACF UserID:JOHNDOE. The above formula is also used for SAF requests if none of the subject's distinguished name fields are specified (CommonName, Title, OrgUnit, Org, Locality, StateProv, or Country) and the PublicKey field contains no information.
18. For PREREGISTER, GENCERT, REQCERT, GENRENEW, REQRENEW, and MODIFYREQS, all CertPlist fields specified must have a non-zero length except for CommonName, which may be null for GENCERT and REQCERT only.
19. For PREREGISTER, GENCERT, REQCERT, GENRENEW, REQRENEW, and MODIFYREQS, OrgUnit, HostIdMap, AuthInfoAcc, Critical, KeyUsage, and ExtKeyUsage may be repeated, where applicable. For all other CertPlist fields if multiple occurrences are found, the last one will be used.
20. For GENCERTand REQCERT, the PublicKey must be either a Netscape Navigator key, a Microsoft Internet Explorer key, or a true PKCS #10 certificate request.
21. For successful EXPORTs where the CertId is not "PKICACERT", the certificate returned in the CertAnchor area is either a base64 encoded DER X509 certificate, a base64 encoded DER PKCS #7 certificate chain, or a DER PKCS #12 certificate package. The base64 data is wrapped with the standard "-----BEGIN CERTIFICATE-----" header and "-----END CERTIFICATE-----" footer. For RACF requests, the returned certificate is always X.509. For PKI Services requests, if the key was not generated by PKI Services, the returned certificate will be packaged as a PKCS #7 certificate chain if at least one hierarchy certificate can be located under the CERTAUTH category and either subsequent access checking is not being performed or the access check user ID has CONTROL authority to IRR.DIGTCERT.EXPORT in the FACILITY Class or is RACF SPECIAL. Otherwise, an X.509 certificate is returned. If the key was generated by PKI Services, the returned certificate will be packaged as a PKCS #12 package with its issuer's certificate. The return code will indicate which format is being returned.

    For successful EXPORTs where the CertId is "PKICACERT", the certificate returned in the CertAnchor area is either the DER-encoded X.509 PKI Services CA certificate or a DER-encoded PKCS #7 certificate chain containing the PKI Services CA and RA certificate. The reason code will indicate which is being returned.

22. For PREREGISTER, GENCERT, REQCERT, GENRENEW, REQRENEW, and MODIFYREQS no validity checking is done for the following fields: AltEmail, AltDomain, AltURI. Additionally, AltPAddr, Mail (formerly Email), EmailAddr, NotifyEmail, and HostIdMap are checked form only (AltPAddr must be in dotted decimal form as per IP Version 4. Mail, Emailaddr, NotifyEmail, and HostIdMap must be in <subject-id>@<host-name>form).
23. For MODIFYREQS, if a modification plist (CertPlist) is specified, all the existing extension request values (KeyUsage, ExtKeyUsage, AltIPAddr, AltURI, AltEmail, AltDomain, HostIdMap, CertPolicies, AuthInfoAcc, and Critical) and validity period values are completely replaced by the new values. Thus if the intent is to alter just one field, the unchanged fields must also be provided. The subject's distinguished name fields work slightly differently. If no subject's distinguished name fields are specified in the modification plist, the existing values are retained. If any subject's distinguished name fields are specified, the name is completely replaced by the values specified.
24. For REQDETAILS, the subject name fields are not returned in the CertPlist if the request has a status of “Preregistered” or if the subject distinguished name does not conform to RACF name standards regarding RDN qualifiers and order. See usage note 15 for more information.
25. SAF return code 8, RACF return code 8, RACF reason code 4 indicates a problem with the value specified for either the Number_parameters or Attributes parameter; or the memory required for parameter storage exceeds system application limits.
26. The R_PKIServ callable service creates SMF type 80 records, with event codes of 69, 70, 72, 73, 74, and 89. RACF audits the invocations of this callable service under the following circumstances:
    a. UAUDIT is in effect for the user
    b. The user has SPECIAL authority and SETROPTS SAUDIT is in effect
    c. The request is successful and SETROPTS AUDIT(USER) is in effect
    d. The request fails due to insufficient authorization

    For detailed information about the SMF records produced, see *z/OS Security Server RACF Macros and Interfaces*.
27. For GENCERT and REQCERT, if the CertPlist fields Email and NotifyEmail are specified together, they must have the same value. Otherwise, the service will fail with reason code 76. For GENRENEW and REQRENEW, if the CertPlist field NotifyEmail is specified and the subject's distinguished name being renewed contains the MAIL attribute, the two values must be the same. Otherwise, the service will fail with reason code 76. In either case "NotifyEmail" is returned in the DiagInfo area as the field name in error.
28. For MODIFYREQS, if a modification parameter list (CertPlist) is specified and it contains the field Email and NotifyEmail was specified on the original request, the new Email value will replace the old NotifyEmail value.
29. For PREREGISTER, GENCERT, REQCERT and MODIFYREQS, if the CertPlist field Subject Alternative Name contains multiple otherName entries, each entry should contain a unique object identifier, though R_PKIServ will not check for the duplication.
30. The supported characters in the CertPlist values are:
    a. Printable string:
        - Country
        - SerialNumber
        - DNQualifier

- JurisdictionCountry

b. IA5 string (Basic Latin):
   - DomainName
   - Mail / Email
   - NotifyEmail
   - EmailAddr
   - AltEmail
   - AltDomain
   - AltURI
   - AuthInfoAcc
   - HostIdMap

c. Basic Latin and Latin-1 supplement:
   - CommonName
   - Title
   - OrgUnit
   - Org
   - Street
   - Locality
   - StateProv
   - PostalCode
   - UID
   - UnstructName
   - UnstructAddr
   - EmailAddr
   - BusinessCategory
   - JurisdictionLocality
   - JurisdictionStateProv

R_proxyserv (IRRSPY00): LDAP interface

Function

The **R_proxyserv** SAF callable service invokes the IBM Tivoli Directory Server for z/OS to store or obtain data which resides in an LDAP directory. Invokers are not required to be Language Environment-enabled.

Requirements

Authorization:
For function codes 1 and 2, any PSW key in supervisor or problem state. For function code 3, supervisor state

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have a FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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. Therefore, 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, for function codes 1 and 2, 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 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. Function code 3 of R_proxyserv requires the caller to be in 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 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.

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 which must be in the primary address space and contains the ALET for the remaining parameters.

Function_code
The name of a halfword (2-byte) area containing the function code. The function code has one of the following values:

Value	Description
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 profile update. DATASET profiles are not supported.

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 that identify the URL of the LDAP server that the IBM Tivoli Directory 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 IBM Tivoli Directory Server Administration and Use for z/OS* 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 IBM Tivoli Directory 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 IBM Tivoli Directory Server Administration and Use for z/OS* 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 IBM Tivoli Directory 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 IBM Tivoli Directory Server Administration and Use for z/OS* 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, by using the FREEMAIN or STORAGE RELEASE macro invocation. Offset values in the output area are all relative to the beginning of the output area.

Table 123. Result_entries output area

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	28	ResultArea	ResultArea
0	0	UNSIGNED	4	ResultAreaLen	Length of results area
4	4	UNSIGNED	1	ResultAreaVersion	Format Version
5	5	UNSIGNED	1	ResultAreaSubpool	Storage subpool
6	6	CHARACTER	22	*	Reserved
28	1C	CHARACTER		ResultAreaData	Function code specific result data

When the function code is X'0001', Return Distinguished Name (DN), the function code specific result data (ResultAreaData) will be mapped as follows:

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	8	DNData	DN result data
0	0	UNSIGNED	4	DNNumber	Number of DNs returned
4	4	UNSIGNED	4	DNList@	Offset to start of DN array

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	8	DNList(*)	DN array
0	0	UNSIGNED	4	DNLen	Length of DN
4	4	UNSIGNED	4	DN@	Offset to DN

When the function code is X'0002', Return Policy Director Authorization Services Attributes, the function code specific data (ResultAreaData) will be mapped as follows:

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	52	PrivilegeData	Privilege result data
0	0	UNSIGNED	1	PrivPasswordValid	Boolean

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
1	1	UNSIGNED	1	PrivAccountValid	Boolean
2	2	UNSIGNED	2	*	Reserved
4	4	UNSIGNED	4	PrivDomainNameLen	Length of domain name
8	8	UNSIGNED	4	PrivDomainName@	Offset to domain name
12	C	UNSIGNED	4	PrivLoginNameLen	Length of login name
16	10	UNSIGNED	4	PrivLoginName@	Offset to login name
20	14	UNSIGNED	4	PrivPrincipalNameLen	Length of principal
24	18	UNSIGNED	4	PrivPrincipalName@	Offset to principal name
28	1C	UNSIGNED	4	PrivUserNameLen	Length of user name
32	20	UNSIGNED	4	PrivUserName@	Offset to user name
36	24	UNSIGNED	4	PrivUserUUIDLen	Length of UUID
40	28	UNSIGNED	4	PrivUserUUID@	Offset to UUID
44	2C	UNSIGNED	4	PrivNumberGroups	Number of groups
48	30	UNSIGNED	4	PrivGroups@	Offset to start of group array

OFFSET (DECIMAL)	OFFSET (HEX)	TYPE	LENGTH	NAME	DESCRIPTION
0	0	STRUCTURE	16	PrivGroups(*)	Group array
0	0	UNSIGNED	4	PrivGroupNameLen	Length of group name
4	4	UNSIGNED	4	PrivGroupName@	Offset to group name
8	8	UNSIGNED	4	PrivGroupUUIDLen	Length of group UUID
12	C	UNSIGNED	4	PrivGroupUUID@	Offset to group UUID

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 124. Parameter list for function code 3

Offset	Length	Type	Name	Description
0	0	Structure	PRXY_F3_PLIST	Function-specific plist for function 3

Table 124. Parameter list for function code 3 (continued)

Offset	Length	Type	Name	Description
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.
		.1..	PRXY_F3_PWUPD2	Reserved for use by the security product. Not for application use. This bit should be set to zero by applications using this interface.
		..1.	PRXY_F3_PWUPD3	Reserved for use by the security product. Not for application use. This bit should be set to zero by applications using this interface.
		...1 1111		Reserved for future use. These bits must be set to 0.
2	8	Character	PRXY_F3_CLASS	RACF class name, padded to the right with blanks. The DATASET class is not supported.
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. When PRXY_F3_CLASS is CONNECT, the profile name takes the format of USER.GROUP.
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 yyyymmddhhiiss.uuuuuuZ where yyyy is the year mm is the month dd is the day hh is the hours ii is the minut ss is the seconds uuuuuu is the microseconds 'Z' is constant If this field contains binary zeros, 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 completed 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
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	0	Invalid function code.
8	8	4	Parameter list error.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	8	An internal error was encountered. A record may be written to LOGREC with more diagnostic information.
8	8	12	A recovery environment could not be established.
8	8	16	Not authorized to use this service.
8	8	20	The LDAP Server address space has not been started or is terminating, or the PC interface has not been started.
8	8	24	Unable to decode the data retrieved from LDAP. A record is written to LOGREC with more diagnostic information.
8	8	28	Unable to determine BIND information for LDAP.
8	8	32	Function not supported for problem state caller.
8	12	8	LDAP Server invocation failed-parameter buffer overflow. This indicates an internal error in IRRRPY00.
8	12	12	LDAP Server invocation failed-unable to allocate storage. The region size for the LDAP Server program should be increased.
8	12	16	LDAP Server invocation failed-LDAP PC handling is not enabled.
8	12	20	LDAP Server invocation failed-abend in the PC service routine. The symptom record associated with this abend can be found in LOGREC.
8	12	24	LDAP Server invocation failed-no control area (internal error). Report the problem to the IBM support center and provide an SVC dump of the LDAP Server address space.
8	12	36	LDAP Server invocation failed-the LDAP Server is busy. Retry the operation.
8	12	40	LDAP Server invocation failed-PC request processing was terminated before completion (the PC catcher initiated the termination). This indicates either an internal error in IRRRPY00 or that the LDAP Server is terminating. If the LDAP Server is terminating, restart it and retry the operation.
8	12	44	LDAP Server invocation failed-PC request processing was terminated before completion (the server agent initiated the termination). This indicates either an internal error in IRRRPY00 or that the LDAP Server is terminating. If the LDAP Server is terminating, restart it and retry the operation.
8	12	52	LDAP Server invocation failed-a lock could not be obtained. This indicates either that the LDAP Server is busy or that an internal error is preventing it from processing requests. If the LDAP Server is not busy, report the problem to the IBM support center and provide an SVC dump of the LDAP Server address space.

SAF return code	RACF return code	RACF reason code	Explanation
8	16	nn	The requested function was not successful. The RACF reason code code is set to the return code from LDAP. See *z/OS IBM Tivoli Directory Server Administration and Use for z/OS* for more information about LDAP return codes. For reason codes other than 32, a record is written to LOGREC containing the LDAP reason string. No record is written for reason code 32, which indicates that the requested information does not exist in the LDAP directory.
8	20	40	Invalid BIND password information was found in either the PROXY segment of the invoker's USER class profile, or in the PROXY segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class.
8	20	44	RACF was unable to retrieve or update the master key/master key token in the SSIGNON segment of the LDAP.BINDPW.KEY profile in the KEYSMSTR class.
8	20	52	RACF cannot locate the CCA support routine.
8	20	56	The invocation of the CCA support routine has failed.

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[1]	In[1]	In[1]	In	N/A	Out	N/A	N/A	N/A
Return the Policy Director attributes for the specified base DN	X'0002'	In[1]	In[1]	In[1]	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 that are not running in a Language Environment. It allows z/OS application servers to perform limited LDAP queries that retrieve information from a directory information tree (DIT). Note that Language Environment-enabled applications can also use this service, if they choose to do so.
2. The R_proxyserv service requires an instance of the LDAP Server on each physical z/OS instance (whether in a sysplex data sharing configuration or not) and each of these LDAP Server instances must be configured to support PC call and the extended operations backend. See *z/OS IBM Tivoli Directory Server Administration and Use for z/OS* 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. Therefore, 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. The LDAP_host, Bind_DN, and Bind_PW parameters are all optional. If any of the three parameters are specified, all must be specified, or R_proxyserv will return an error. If all three parameters are omitted, RACF attempts to determine this information from the PROXY segment associated with the RACF user identity of the invoker (that is, the server's address space level ACEE). If the user profile PROXY segment is found, but any of the corresponding segment values (LDAPHOST, BINDDN, or BINDPW) are not defined, R_proxyserv will return an error. If the LDAP_host, Bind_DN, and Bind_PW parameters are omitted and the PROXY segment is not defined for the invoker's user identity, RACF will then look for the IRR.PROXY.DEFAULTS profile in the FACILITY class. If this profile is not found or does not have a PROXY segment or does not have values defined for LDAPHOST, BINDDN, and BINDPW, R_proxyserv will return an error.
5. The format of the Result_entries output area differs, based on the function code specified. Mappings are provided for each format (see Mappings for Result_entries output area). Storage will be obtained in primary in the subpool indicated in the Result_entries output area and it is the responsibility of the invoker to release this storage.

Related services

None

R_ptrace (IRRSPT00): Ptrace authority check

Function

The **R_ptrace** service checks whether the calling process can ptrace the target process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. **R_ptrace** checks whether the caller is a superuser, or whether the caller is the owner of the target process. If the caller is the owner of the target process, R_ptrace verifies that the target process is not running a SETUID or SETGID program. If the caller is a superuser, R_ptrace does not verify that the target precess is not running a SETUID or SETGID program.
2. If the caller is not superuser nor the process owner, an authorization check is performed on the resource name in the UNIXPRIV class shown in Table 125. If the authorization check is successful, the caller is treated as a superuser.

Table 125. UNIXPRIV class resource names used in R_ptrace

Audit function code	Resource name	Access required
N/A	SUPERUSER.PROCESS.PTRACE	READ

3. When the SECLABEL class is active, and the high order bit of the Target_PID is on, R_ptrace checks if the caller's security label is equivalent to the target process's security label, unless the ACEE indicates trusted or privileged authority.

Format

```
CALL IRRSPT00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Target_process_UIDs,
               ALET, Target_process_GIDs,
               ALET, Target_PID
              )
```

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 words 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.

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.

Target_process_UIDs
The address of a 3-word area containing the real, effective, and saved z/OS UNIX user identifiers (UIDs) (in that order) for the target process. IRRSPT00

uses the high-order bit of the Target_PID to indicate that this area is 5 words, containing the real, effective, and saved z/OS UNIX user identifiers, and the 8-byte security label for the target process.

Target_process_GIDs
The address of a 3-word area containing the real, effective, and saved z/OS UNIX group identifiers (GIDs) (in that order) for the target process.

Target_PID
The name of a fullword containing the PID of the target process. The high order bit of the PID is used to indicate that the Target_process_UIDs field is a five word rather than a three-word area.

Return and reason codes

IRRSPT00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The caller is not authorized to ptrace the target process.
8	8	12	An internal error occurred during RACF processing.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. An audit record is optionally written, depending on the audit options in effect for the system.
3. This service uses task-level support when z/OS UNIX has indicated in the task's ACEE that this is a task-level process.

Related services

None

R_setegid (IRRSEG00): Set effective GID, set all GIDs

Function

The **R_setegid** service checks whether the user is authorized to change the GID and, if so, changes the effective GID for the current process.

If the high-order bit of the input GID is on, the real, effective, and saved GIDs are changed for the current process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. If the high-order bit of the input GID is off and if the user is the superuser or if the input GID is equal to the real or saved GID of the calling process, the effective GID of the process is changed to the input GID. The real and saved GIDs are not changed. The new values of the GIDs are returned to the calling process.

Format

```
CALL IRRSEG00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, GID,
               ALET, Output_area
              )
```

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 words 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.

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.

GID
The name of a fullword containing the GID to be set. The GID must be defined to RACF. If the high-order bit is on, the GIDs stored in the output area are stored as the real, effective, and saved GIDs (in that order) for the current process.

Output_area
The name of a 3-word area in which the new real, effective, and saved GIDs (in that order) are returned. If the high-order bit of the GID is on, the real, effective, and saved GIDs in this area are stored as the real, effective, and saved GIDs for the current process.

Return and reason codes

IRRSEG00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The GID is not defined to RACF.
8	8	8	The user is not authorized to change the GID.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. IRRSEG00 only changes the GIDs. The user's current group in the ACEE is not changed. Therefore, IRRSEG00 only affects access to z/OS UNIX files. Access to other MVS files is not changed.
3. An audit record is written.

Related services

None

R_seteuid (IRRSEU00): Set effective UID, set all UIDs

Function

The **R_seteuid** service checks whether the user is authorized to change the z/OS UNIX user identifiers (UIDs) and, if so, changes the effective UID for the current process.

If the high-order bit of the input UID is on, the real, effective, and saved UIDs are changed for the current process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. If the high-order bit of the input z/OS UNIX user identifier (UID) is off and if the user is the superuser or if the input UID is equal to the real or saved UID of the calling process, the effective UID of the process is changed to the input UID. The real and saved UIDs are not changed. The new values of the UIDs are returned to the calling process.

Format

```
CALL IRRSEU00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, UID,
               ALET, Output_area
              )
```

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 words 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.

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.

UID
The name of a fullword containing the z/OS UNIX user identifier (UID) to be set. The UID must be defined to RACF. If the high-order bit is on, the UIDs stored in the output area are stored as the real, effective, and saved UIDs (in that order) for the current process.

Output_area
The name of a 3-word area in which the new real, effective, and saved z/OS UNIX user identifiers (UIDs) (in that order) are returned. If the high-order bit of the UID is on, the real, effective, and saved UIDs in this area are stored as the real, effective, and saved UIDs for the current process.

Return and reason codes

IRRSEU00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The z/OS UNIX user identifier (UID) is not defined to RACF.
8	8	8	The user is not authorized to change the z/OS UNIX user identifier (UID).
8	8	12	An internal error occurred during RACF processing
8	8	16	Recovery could not be established.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. For additional security-related information, see the description of the **seteuid** callable service in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*
3. An audit record is written.

Related services

None

R_setfacl (IRRSCL00):Unix access control lists

Function

The **R_setfacl** callable service is used to maintain the access lists for a UNIX file or directory

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. To change the ACL, the user must be a superuser or must be the owner of the file. To be considered a superuser, the user must either have a UID value of 0, or must be permitted with at least READ access to the resource named SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class.
2. If the SECLABEL class is active and the file or directory has a security label, then the current security label of the process must be greater than or equal to the security label of the resource or the security label of the resource must be greater than or equal to the current security label of the process, that is, the security labels are not disjoint. If MLFSOBJ is active, a failure will occur if the resource does not have a security label. Security label checking is bypassed if the ACEE indicates trusted or privileged authority or if the service has passed a system CRED.

Format

```
CALL IRRSCL00  (Work_area,
                ALET, SAF_return_code,
                ALET, RACF_return_code,
                ALET, RACF_reason_code,
                ALET, ACL_Update,
                ALET, ACL_Update_length,
                ALET, FSP,
                ALET, File_identifier,
                ALET, CRED,
       )
```

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 words 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.

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.

ACL_Update
The name of an area containing the type of ACL being updated, the operation being requested, and an ACL structure which contains entries to be added, updated, or removed. See the RACL_Edit structure in the IRRPCOMP macro for the mapping of this area.

The ACL structure is mapped by IRRPFACL. If the operation is add and entries are specified in this ACL mapping, then the current ACL will be replaced with the entries specified in this structure.

ACL_Update_Length
The name of a fullword containing the length of the ACL_Update buffer.

FSP
The name of the IFSP for the file whose mode bits are to be changed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas* for more information. The CRED contains a pointer to the ACL being modified/deleted, or contains the address of a buffer in which to create a new ACL.

Return and reason codes

IRRSCL00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The user is not authorized to change the ACL.
8	8	8	The maximum of 1024 entries would be exceeded by this request.
8	8	12	An internal error occurred during RACF processing.
8	8	16	An error was encountered in the ACL passed in the ACL_Update parameter. FACL_ErrOff contains the offset to the header field or the ACL entry in error. See usage notes for possible error conditions.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	20	ACL buffer provided by caller is not large enough to contain a valid ACL
8	8	24	Input parameter list error.
8	0	32	The CRED user type is not supported.

Usage notes

1. This service is intended only for use by a z/OS UNIX System Services file system and by z/OS UNIX System Services servers. The service contains support for z/OS UNIX System Services servers, but cannot be directly invoked by an z/OS UNIX System Services server.
2. An access list may contain a maximum of 1024 entries.
3. R_setfacl will manage the bit in the File Security Packet (FSP) which indicates the presence of an ACL of a given type. That is, when an ACL is successfully added (by using either the add or modify operation), R_setfacl will turn on the appropriate bit in IFSP_FLAG2 (either IFSP_Access_Acl, IFSP_File_Model_Acl, or IFSP_Dir_Model_Acl). For a delete operation, or an add or modify operation which results in an empty ACL, RACF will turn off the appropriate bit in IFSP_FLAG2 .
4. When a modify operation is specified, requests to delete ACL entries are processed before requests to add or modify entries.
5. If a modify operation is specified and an ACL does not exist, it will be created. Likewise, if a modify request for a specific ACL entry is specified, and that entry does not exist, it will be created.
6. If a delete request is specified, but an ACL does not exist, the request will be ignored. Likewise, if a delete request for a specific ACL entry is specified, and that entry does not exist, it will be ignored.
7. If an add request is specified, and an ACL already exists, it will be replaced in accordance with the contents of the RACL_Edit structure pointed to by the ACL_Update parameter. If there is no RACL_Edit in this context, the existing ACL will be deleted.
8. If a delete request is specified, and a RACL_Edit structure is also contained within the structure pointed to by the ACL_Update parameter, then the RACL_Edit is ignored and the ACL is deleted.
9. An audit record (or records) is optionally written, depending on the audit options in effect for the system.
10. The parameter list passed to this service is a variable-length (VL) parameter list. The high-order bit of the last field must be set to mark the end of the parameter list.
11. The caller must pass in the length and address of a buffer which contains the ACL being modified, or in which a new ACL is to be created. The buffer must be large enough to contain the maximum size ACL. The length and address fields are contained within the CRED, and different field names are used depending on which ACL is being created, modified, or deleted. For an access ACL, use CredAccAcl and CredAccAclLen. For a directory model ACL, use CredDirModelAcl and CredDirModelAclLen. For a file model ACL, use CredFileModelAcl and CredFileModelAclLen.
12. R_setfacl will perform validation on the ACL passed into the service as part of the RACL_Edit parameter of IRRPCOMP. An error in this ACL will result in a SAF return code 8, RACF return code 8, and RACF reason code 16 (decimal).

If an error is detected, the FACL_ErrOff field within this ACL mapping will be updated with the offset (from the start of the header) to the header field or ACL entry in error. Some of the items validated are: eye catcher = "FACL", version = 1, length is large enough to contain the number of entries specified in FACL_Num_Entry, the ACL contains at least one entry, ACL entry type is 1 or 2, and UID/GID value is greater than or equal to 0.

13. An error with the input parameter list will result in a SAF return code 8, RACF return code 8, and RACF reason code 24 (decimal). Some of the items validated are: all addresses in the parameter list are non-zero, the variable-length parameter list bit is set, the ACL_Update_Length parameter specifies a length which is large enough to contain the ACL_Update area, the operation type and ACL type specified in the ACL_Update area are valid, and the pointers in the CRED which point to ACL buffers are non-zero and point to an area which is large enough to contain the ACL.

Related services

R_chmod, R_chown, ck_access

R_setfsecl (IRRSSB00): Security label

Function

The **R_setfsecl** service changes the security label in the FSP to the value specified in the CRED, or if no value is specified, the security label of the address space level ACEE.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

This function is available only to supervisor state callers passing a system CRED, or, if no security label is currently assigned, a user running with SPECIAL authority.

Format

```
CALL IRRSSB00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, FSP,
               ALET, File_Identifier,
               ALET, CRED
              )
```

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 words 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.

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.

FSP
The name of the IFSP for the file whose security label is to be changed.

File_Identifier
The name of a 16-byte area containing a unique identifier of the file.

CRED
The name of the CRED structure for the current file system syscall. See *z/OS Security Server RACF Data Areas*.

Return and reason codes

IRRSSB00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
0	0	4	The service was successful but no processing was done because the SECLABEL class is not active.
4	0	0	RACF is not installed.
8	8	0	Caller not authorized.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	12	An internal error occurred during RACF processing.
8	8	24	Input parameter list error.

Usage notes

1. This service is intended only for use by the z/OS UNIX System Services file system and by z/OS UNIX System Services file servers.
2. This service requires a system CRED if the FSP already contains a security label.
3. If the FSP does not already contain a security label, this service requires a system CRED, or a user CRED and an ACEE that indicates the user has the SPECIAL attribute.
4. If the security label is not specified, the security label in the FSP will be set to the value from the address space level ACEE.
5. An audit record is optionally written, depending on the audit options in effect for the system.
6. Callers receiving a SAF RC0, RACF RC0, RACF RS4 indicating that the SECLABEL class is not active, may choose to listen for the ENF event that indicates the SECLABEL class has been RACLISTed before calling this service again. See *z/OS MVS Programming: Authorized Assembler Services Guide* for more information.

Related services

makeFSP, make_root_FSP

R_setgid (IRRSSG00): Set group name

Function

The **R_setgid** service checks whether the user is authorized to change the GIDs and, if so, changes the real, saved, or effective GID (or some combination of these) for the current process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. If the calling process is a superuser, the real, saved, and effective GIDs are changed. If the calling process is not a superuser but the input GID is equal to the real or saved GID, the effective GID of the process is changed. If neither condition is met, the GIDs of the process are not changed, and an error return code and an error reason code are returned.

Format

```
CALL IRRSSG00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, GID,
               ALET, Output_area
              )
```

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 words 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.

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.

GID
The name of a fullword containing the GID to be set. The GID must be defined to RACF.

Output_area
The name of a 3-word area in which the new real, effective, and saved GIDs (in that order) are returned.

Return and reason codes

IRRSSG00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The GID is not defined to RACF.
8	8	8	The user is not authorized to change the GID.
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. IRRSSG00 changes only the GIDs. No change is made to the user's current group in the ACEE. Therefore, IRRSSG00 only affects access to z/OS UNIX files. Access to other MVS files is unchanged.
3. An audit record is written.

Related services

None

R_setuid (IRRSSU00): Set z/OS UNIX user identifier (UID)

Function

The **R_setuid** service checks whether the user is authorized to change the z/OS UNIX user identifiers (UIDs) and, if so, changes the real, saved, or effective UID (or some combination of these) for the current process.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

1. If the calling process is a superuser, the real, saved, and effective z/OS UNIX user identifiers (UIDs) are changed. If the calling process is not a superuser, but the input UID is equal to the real or saved UID, the effective UID of the process is changed. If neither condition is met, the UIDs of the process are not changed, and an error return code and and an error reason code are returned.

Format

```
CALL IRRSSU00 (Work_area,
               ALET,SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, UID,
               ALET, Output_area
              )
```

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 words 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.

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.

UID
The name of a fullword containing the z/OS UNIX user identifier (UID) to be set. The UID must be defined to RACF.

Output_area
The name of a 3-word area in which the new real, effective, and saved z/OS UNIX user identifiers (UIDs) (in that order) are returned.

Return and reason codes

IRRSSU00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	The z/OS UNIX user identifier (UID) is not defined to RACF.
8	8	8	The user is not authorized to change the z/OS UNIX user identifier (UID).
8	8	12	An internal error occurred during RACF processing.
8	8	16	Recovery could not be established.

Usage notes

1. This service is intended only for use by the MVS BCP.
2. For additional security-related information, see the description of the **setuid** callable service in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*.
3. An audit record is optionally written, depending on the audit options in effect for the system.

Related services

None

R_ticketserv (IRRSPK00): Parse or extract

Function

The **R_ticketserv** callable service allows callers to read GSS-API context tokens and also provides RACF PassTicket services. **R_ticketserv** is similar to the Extract client principal (function code 1) of the **R_GenSec** callable service (See "Extract client principal functions (Function code 1):" on page 207 for more information). The main difference is that **R_ticketserv** cannot be invoked from AMODE 64.

R_ticketserv can enable z/OS application servers to parse or extract principal names from a GSS-API context token, which when the request includes a GSS-API context token and the intended recipient is the z/OS application server, enables a z/OS application server to determine the client principal who originated an application-specific request. See *z/OS Integrated Security Services Network Authentication Service Administration* and *z/OS Integrated Security Services Network Authentication Service Programming* for more information about the GSS-API services supported on z/OS.

R_ticketserv also allows callers to generate and evaluate PassTickets.

R_ticketserv can be used by an application, which does not have access to a C runtime environment providing the GSS-API functions required for authentication, through decrypting the service ticket in the provided token. Decrypting the service

ticket allows **R_ticketserv** to determine if a principle has access to a service. See *z/OS Integrated Security Services Network Authentication Service Programming* for more information.

The **R_ticketserv** service, like other SAF calls that use Kerberos for authentication, requires the SKRBKDC started task to be running in the same address space.

R_ticketserv checks that the server principal in the ticket maps to the current user ID, before the ticket is accepted. If tickets for different service principals are to be accepted, a KERBLINK mapping must exist for the service principal, and the current user ID must be granted at least READ authority to the KERBLINK profile. For example, if the server principal in the ticket is princ2/fully_qualified_hostname and is associated with user2, and the current user is user1 who is defined to RACF, you can use the PERMIT command to grant user1 READ authority to the KERBLINK profile of the server principal. With READ authority, user1 can decrypt the tickets of user2. However, before using the PERMIT command to grant this READ authority, you must activate RACF protection for the KERBLINK profile (if not already active). If the server principal does not have a KERBLINK profile, you must create one. The following procedure shows how to activate RACF protection, create a KERBLINK profile, and grant READ authority:

1. Activating RACF protection for the KERBLINK class using the SETROPTS command:

   ```
   SETROPTS CLASSACT(KERBLINK)
   ```

   See *z/OS Security Server RACF Command Language Reference* for more information about the SETROPTS command.
2. Creating a KERBLINK profile for the server principal (princ2/fully_qualified_hostname) in the ticket using the RDEFINE command:

   ```
   RDEFINE KERBLINK princ2/fully_qualified_hostname
   ```

   See *z/OS Security Server RACF Security Administrator's Guide* for more information about KERBLINK profiles and automatic local principal name mapping.
3. Using the PERMIT command to grant user1 READ authority to the KERBLINK profile of the server principal (princ2/fully_qualified_hostname):

   ```
   PERMIT princ2/fully_qualified_hostname CLASS(KERBLINK) ID(user1) ACCESS(READ)
   ```

   See *z/OS Security Server RACF Security Administrator's Guide* for more information about the PERMIT command.

Requirements

Authorization:
Any PSW key in supervisor state or problem state

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have an FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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. Therefore, the last word in the parameter list must have a 1 in the high-order (sign) bit.

RACF authorization

For servers not running in system key or supervisor state, the use of R_ticketserv service to manipulate GSS-API context tokens is authorized by the resource IRR.RTICKETSERV in the FACILITY class. The application server must be running with a RACF user or group that has at least READ authority to this resource. If the class is inactive, or the resource is not defined, only servers running system key or supervisor state may use the R_ticketserv service.

For all callers, the use of R_ticketserv service to use PassTicket services is authorized by resources in the PTKTDATA class which correspond to the application ID and target userid used in the PassTicket operation. The application server must be running with a RACF user or group that has the authority specified in the table below. If the PTKTDATA class is inactive, or the resource is not defined, the request will fail due to insufficient authority. All callers, regardless of PSW key or state, must pass the authorization check. Generic profiles may be used for authorization.

Operation	Profile name	Required access
Generate PassTicket	IRRPTAUTH.*application.target-userid*	UPDATE
Evaluate PassTicket	IRRPTAUTH.*application.target-userid*	READ

See *z/OS Security Server RACF Security Administrator's Guide* for more information about configuring RACF to use PassTicket services.

The PassTicket evaluation function is meant to be used to evaluate PassTicket for users who do not exist in RACF, for example temporary or generated userids, however it can be used with RACF defined users. There is no revocation of users due to failed password attempts, so care must be taken in granting access to the PassTicket evaluation function.

The PassTicket evaluation service only evaluates that a PassTicket is computationally valid for a given userid and application. It does not actually log the user in to the system or create any kind of z/OS security context for that user.

To log in a user using a PassTicket, use a standard z/OS function such as __login() or RACROUTE REQUEST=VERIFY.

Format

```
CALL IRRSPK00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Function_code,
                        Option_word,
                        Ticket_area,
                        Ticket_options,
                        Ticket_principal_userid
                        Application_Id
              )
```

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 words containing the ALETs must be in the primary address space.

SAF_return_code
The name of a full word in which the SAF router returns the SAF return code.

RACF_return_code
The name of a full word in which the service routine stores the return code.

RACF_reason_code
The name of a full word in which the service routine stores the reason code.

ALET
The name of a word that must be in the primary address space and contains the ALET for the following fields:

- Function_code
- Option_word
- Ticket_area
- Ticket_options
- Ticket_principal_userid
- Application_Id

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'
Parse-specified ticket and return Network Authentication Service principal name.

X'0003'
PassTicket operation

Option_word
The name of a fullword containing binary zeros. The area pointed to by this parameter is reserved for future use.

Ticket_area

For function code X'0001', this is the name of an area that consists of a 2-byte field set to the length of the contents of the GSS-API context token, followed by the contents of the GSS-API context token. The GSS-API context token is assumed by SAF to have been created exclusively by the Network Authentication Service Kerberos mechanism.

If the SKRBKDC started task is running with the -NOKDC option (a KDC is not available in the same address space), the keytab file is used to locate the service key that is needed to decrypt the server ticket. The following keytab file is used by the **R_ticketserv** service and must only be readable by the SKRBKDC started task:

/etc/skrb/home/kdc/tickserv.ktf

To decrypt the ticket, the server key in the keytab entry must match that used by the Key Distribution Centre (KDC) for the given server, version, and encryption type used when the KDC issued the service ticket contained in the token.

Note: To decrypt the ticket, you need to know the password used to create the server entry in the KDC.

If the SKRBKDC started task is running with the -KDC option (a KDC is available on the same image), the service key in the KDC is used to decrypt the ticket in the token, and the keytab file is not used.

For function code X'0003', this is the name of a 10-byte area that consists of a 2-byte length field, followed by an 8-byte PassTicket field. If ticket_options indicates that a PassTicket is to be generated, the newly generated PassTicket will be returned in this area. The caller must specify a length of at least 8 to indicate that an acceptable buffer has been supplied for the generated PassTicket.

If ticket_options indicates that a PassTicket is to be evaluated, this contains the PassTicket to be evaluated.

Ticket_options

The name of a fullword containing the address of a binary bit string that identifies the ticket-specific processing to be performed. This parameter is unused when a function code of X'0001' is specified.

When function code X'0003' is specified, the bit string is used as an integer to specify which PassTicket operation to perform.

```
X'00000001' - Generate a PassTicket
X'00000002' - Evaluate a PassTicket
```

Ticket_principal_userid

For function code X'0001' this is the name of a 242-byte area that consists of a 2-byte length field followed by the name of the Ticket principal user ID.

Note: Fully qualified Network Authentication Service names will be returned, using a case-sensitive, DCE-like naming convention: /.../realm_name/principal_name

When function code X'0001' is specified, this parameter must have its high order bit on to signal that it is the last parameter in the parameter list.

For function code X'0003', this is the name of a 10-byte area that consists of a 2-byte length field followed by the userid id for whom a PassTicket operation is to be performed.

Application_Id

The name of an area that consists of a 2-byte length field followed by up to 8 bytes containing the name of the PassTicket application ID. If the function code is x'0001', this parameter is not needed and can be omitted. Refer to *z/OS Security Server RACF Security Administrator's Guide* for information about what to specify for Application_Id.

When function code X'0003' is specified, this parameter must have its high order bit on to signal that it is the last parameter in the parameter list.

Return and reason codes

IRRSPK00 may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	0	Invalid function code.
8	8	4	Parameter list error.
8	8	8	An internal error was encountered.
8	8	12	A recovery environment could not be established.
8	8	16	Not authorized to use this service.
8	8	20	High order bit was not set to indicate last parameter.
8	12	8	Invocation of the Security Server Network Authentication Service Program Call (PC) interface failed with a 'parameter buffer overflow' return code. This indicates an internal error in IRRSPK00.
8	12	12	Invocation of the Security Server Network Authentication Service Program Call (PC) interface failed with an 'unable to allocate storage' return code. The region size for the Security Server Network Authentication Service started task (SKRBKDC) should be increased.
8	12	16	Invocation of the Security Server Network Authentication Service Program Call (PC) interface failed with a 'local services are not available' return code. This indicates that the Security Server Network Authentication Service started task (SKRBKDC) address space has not been started or is terminating.

SAF return code	RACF return code	RACF reason code	Explanation
8	12	20	Invocation of the Security Server Network Authentication Service Program Call (PC) interface failed with an 'abend in the PC service routine' return code. The symptom record associated with this abend can be found in the logrec data set.
8	12	24	Invocation of the Security Server Network Authentication Service Program Call (PC) interface failed with an 'unable to obtain control lock' return code. This can occur if the task holding the lock is not being dispatched (for example, a dump is in progress).
8	16	X'nnnnnnnn'	The Security Server Network Authentication Service was not able to successfully extract the client principal name from the supplied Kerberos V5 ticket. X'nnnnnnnn' is the Kerberos return code. Refer to the Security Server Network Authentication Service documentation for more information.
8	16	28	Unable to generate PassTicket.
8	16	32	PassTicket evaluation failure. Possible reasons include: • PassTicket to be evaluated is not a successful PassTicket • The PassTicket to be evaluated was already evaluated before and replay protection is in effect. • No PTKTDATA profile exists to match the specified application • An internal error occurred.

Usage notes

Function code=X'0001':

1. This service is intended for use by z/OS application servers. The service allows application servers with a GSS-API context token (created with the Kerberos V5 mechanism) to determine the Kerberos client principal associated with the token.
2. This service requires that the Security Server Network Authentication Service be installed and running. Otherwise, SAF return code 8, RACF return code 12, and RACF reason code 16 will be returned to the invoker.
3. In a datasharing sysplex, there must be an Security Server Network Authentication Service instance running on each system in the sysplex. The Security Server Network Authentication Service instances must all be in the same realm and share the same RACF database (if they do not share the same database, then they cannot be in the same realm).
4. An ALET must be specified for the SAF_return_code, RACF_return_code, and RACF_reason_code parameters, and a single ALET specified for all of the remaining paramenters.

5. 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. The first parameter that can have the high-order bit on, ending the parameter list, is the Ticket_principal_userid parameter.
6. A SAF return code 8 and a RACF return code 16 indicates that the Security Server Network Authentication Service was unable to process the input GSS-API token. The return code is passed back to the invoker as the RACF reason code. The following list shows some common return codes:
   - X'861B6D04' (G_BUFFER_ALLOC)=storage not available for GSS-API control block.
   - X'861B6D06' (G_WRONG_SIZE)=client principal name is too long for result buffer.
   - X'861B6D0B' (G_BAD_TOK_HEADER)=the GSS-API token header is incorrect.
   - X'861B6D58' (G_UNEXPECTED_TOKEN)=the GSS-API token was not created by the gss_init_sec_context() function.
   - X'861B6D60' (G_UNSUPPORTED_MECHANISM)=unsupported GSS-API security mechanism.
   - X'96C73A07'(KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN)=the current RACF userid is not associated with a Kerberos principal.
   - X'96C73A20'(KRB5KDC_AP_ERR_TKT_EXPIRED)=Kerberos ticket is expired.
   - X'96C73A25'(KRB5KDC_AP_ERR_SKEW)=Client and server clocks are not synchronized or authenticator is expired.
   - X'96C73A90'(KRB5KDC_AP_WRONG_PRINC)=the server principal in the GSS-API security token does not match the principal associated with the current RACF userid.
   - X'96C73C02'(KRB5_NOMEM)=storage not available for Kerberos control block.

Function code=X'0003': 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. Only the Application_Id parameter must have it's high order bit set when the function_code =X'0003'.

Parameter usage

Parameter	Function code X'0001' and X'0003'
SAF_return_code	Output
RACF_return_code	Output
RACF_reason_code	Output
Function _code	Input
Option_word	Reserved
Ticket_area	Input for function code X'0001' Input or output for function code X'0003'
Ticket_options	Input

Parameter	Function code X'0001' and X'0003'
Ticket_principal_userid	Output for function code X'0001' Input for function code X'0003'
Application_Id	Input

Related services

R_kerbinfo, R_usermap

R_umask (IRRSMM00): Set file mode creation mask

Function

The **R_umask** service sets the file mode creation mask for the current process to the permission bits specified in the input mode parameter. It returns the permission bits that were in the file mode creation map in the mode parameter.

Requirements

Authorization:
Any PSW key in supervisor state

Dispatchable unit mode:
Task of z/OS UNIX user

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
None

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area. The words containing the ALETs must be in the primary address space.

RACF authorization

None

Format

```
CALL IRRSMM00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Mode
              )
```

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 words 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.

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.

Mode
The name of a word containing the mode bits to be set in file mode creation mask of the current process. Only the file permission bits in the mode parameter are used. Other defined bits are ignored.

On output, the mode word is zeroed and the permission bits that were in the file mode creation mask are set in the mode word.

See “File type and file mode values” on page 4 for a definition of the security bits in the mode parameter.

Return and reason codes

IRRSMM00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.

Usage note

- This service is intended only for use by the MVS BCP and by z/OS UNIX servers. The service contains support for z/OS UNIX servers, but cannot be directly invoked by a z/OS UNIX server.

Related services

chmod, makeFSP

R_usermap (IRRSIM00): Map application user

Function

The **R_usermap** service enables z/OS application servers to determine the application user identity associated with a RACF user ID, or to determine the RACF user ID associated with an application user identity or digital certificate, except for Identity Propagation in which case a user's Distinguished Name and a Registry/Realm Name will be used to determine the associated RACF user ID, but not the reverse. Examples of applications supported are RACF user ID, application user identity, application, Lotus Notes® for z/OS and Novell Directory Services (NDS).

This service can only map application user identities which have already been defined to RACF:

- For Lotus Notes for z/OS, the RACF USER profile must have an LNOTES segment containing a short name. This can be added with the ADDUSER or ALTUSER command, or the R_admin callable service.
- For NDS for z/OS, the RACF USER profile must have an NDS segment containing a user name. This can be added with the ADDUSER or ALTUSER command, or the R_admin callable service.
- For digital certificates, the certificate must be associated with a RACF user ID through automatic registration or with the RACDCERT command.
- For Security Server Network Authentication Service, local Kerberos principals require a RACF USER profile with a KERB segment containing a principal name. Foreign Kerberos principals must be defined to RACF using KERBLINK profiles.
- For Identity Propagation, the distributed identity (user's Distinguished Name) must be associated with a RACF user ID. Use the RACMAP command to create the association between the distributed identity and a RACF defined user ID (this association is also known as a 'filter').

Requirements

Authorization:
Any PSW key in supervisor or problem state

Dispatchable unit mode:
Task of user

Cross memory mode:
PASN = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
ESTAE. Caller cannot have a FRR active.

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
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

Function codes X'0001' through X'0006' only: The use of the R_usermap service is authorized by the resource **IRR.RUSERMAP** in the **FACILITY** class for servers not running in system key or supervisor state. The application server must be running with a RACF user or group that has at least **READ** authority to this resource. Only servers running in system key or supervisor state may use the R_usermap service if the class is inactive or the resource is not defined.

Function codes X'0008' only: The use of the R_usermap service is authorized by the resource **IRR.IDIDMAP.QUERY** in the **FACILITY** class for servers not running in system key or supervisor state. The application server must be running with a RACF user or group that has at least **READ** authority to this resource. Only servers running in system key or supervisor state may use the R_usermap service if the class is inactive or the resource is not defined.

Format

```
CALL IRRSIM00 (Work_area,
               ALET, SAF_return_code,
               ALET, RACF_return_code,
               ALET, RACF_reason_code,
               ALET, Function_code,
                        Option_word,
                        RACF_userid,
                        Certificate,
                        Application_userid
                        Distinguished_Name,
                        Registry_Name                    )
```

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 and must be on a doubleword boundary.

ALET
The name of a word containing the ALET for the following parameter. Each ALET can be different. The last ALET in the parameter list will be used for the remainder of the parameters. The words 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.

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.

Function_code
The name of a halfword containing the function code. The function code has one of the following values:

X'0001'
Return the Lotus Notes for z/OS application user identity associated with the supplied RACF user ID.

X'0002'
Return the RACF user ID associated with the supplied Lotus Notes for z/OS application user identity or digital certificate.

X'0003'
Return the NDS for z/OS application user identity associated with the supplied RACF user ID.

X'0004'
Return the RACF user ID associated with the supplied NDS for z/OS application user identity or digital certificate.

X'0005'
Return the Network Authentication Service application user identity associated with the supplied RACF user ID.

Note: This functions only with local Network Authentication Service principals.

X'0006'
Return the RACF user ID associated with the supplied Network Authentication Service application user identity or digital certificate.

X'0008'
Return the RACF user ID associated with the supplied user's Distinguished Name and Registry/Realm Name.

Option_word
The name of a fullword containing binary zeros. The area pointed to by this parameter is reserved for future use.

RACF_userid
The name of a 9-byte area that consists of a 1-byte length field followed by up to 8 characters. It must be specified in uppercase. If not specified, the length must equal 0.

Certificate
The name of an area that consists of a 4-byte length field followed by a digital certificate. The certificate must be a single BER encoded X.509 certificate. If not specified, the length must equal 0.

Application_userid
The name of a 248-byte area that consists of a 2-byte length field followed by the name of the application user identity. If not specified, the length must equal 0.

Distinguished_Name
The name of an area that consists of a 2-byte length field followed by the distinguished name (distributed user ID), in UTF-8 format, of up to the maximum length allowed by the RCVT field RCVTDNL (currently 246). If not

specified, the length must equal 0. For a non-zero length, the field cannot be all blanks (x'20'), all nulls (x'00'), or a combination of blanks and nulls.

Note:

1. The following operations are performed on a copy of the data. The original data is not modified.
   - All leading and trailing blanks (x'20'), nulls (x'00'), or combination of blanks and null characters will be removed from the string and the length will be appropriately adjusted.
   - If the distributed-identity-user-name (user name) is in X.500 format the name will be normalized before it is used to find the matching RACF user ID that is associated with the distributed identity filter.
2. The normalization rules are described in detail under RACMAP MAP.

Registry_Name
The name of an area that consists of a 2-byte length field followed by the registry/realm name, in UTF-8 format, of up to the maximum length allowed by the RCVT field RCVTRL (currently 255). If not specified, the length must equal 0. For a non-zero length, the field cannot be all blanks (x'20'), all nulls (x'00'), or a combination of blanks and nulls.

Note: All leading and trailing blanks (x'20'), nulls (x'00'), or combination of blanks and null characters will be removed from the string and the length will be appropriately adjusted. This operation is performed on a copy of the data. The original data is not modified.

Return and reason codes

R_usermap may return the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful.
4	0	0	RACF is not installed.
8	8	4	Parameter list error occurred.
8	8	8	An internal error occurred during RACF processing.
8	8	12	Recovery environment could not be established.
8	8	16	There is no mapping between RACF and an application. For function codes 1 and 3, and 5, the RACF user ID exists but there is either no SNAME in the LNOTES segment or no LNOTES segment, or there is no UNAME in the NDS segment or no NDS segment, or no KERBNAME in the KERB segment, or no KERB segment. For function codes 2, 4, and 6, there is no mapping to a RACF user ID for the application identity provided.
8	8	20	Not authorized to use this service.
8	8	24	The specified RACF user ID does not exist.
8	8	28	Certificate is not valid.

SAF return code	RACF return code	RACF reason code	Explanation
8	8	32	Either no RACF user ID is defined for this certificate, or the certificate status is NOTRUST.
8	8	36	High order bit was not set to indicate last parameter.
8	8	40	The Distinguished Name length is not valid, or the Distinguished Name string is all blanks (x'20'), all nulls (x'00'), or a combination of blanks and nulls.
8	8	44	The Registry Name length is not valid, or the Registry Name string is all blanks (x'20'), all nulls (x'00'), or a combination of blanks and nulls.
8	8	48	There is no distributed identity filter mapping the supplied distributed identity to a RACF user ID, or The IDIDMAP RACF general resource class is not active or not RACLISTed

Parameter usage

Table 126. Parameter usage

Parameter	Function code 1 (RACF to notes)	Function code 2 (Notes to RACF)	Function code 3 (RACF to NDS)	Function code 4 (NDS to RACF)	Function code 5 (RACF to KERB)	Function code 6 (KERB to RACF)	Function code 8 (DID to RACF)
SAF_return_code	Output	Output	Output	Output	Output	Output	Output
RACF_return_code	Output	Output	Output	Output	Output	Output	Output
RACF_reason_code	Output	Output	Output	Output	Output	Output	Output
Function_code	Input	Input	Input	Input	Input	Input	Input
Option_word	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
RACF_userid	Input	Output	Input	Output	Input	Output	Output
Certificate	N/A	Input	N/A	Input	N/A	Input	N/A
Application_userid	Output	Input	Output	Input	Output	Input	N/A
Distinguished_Name	N/A	N/A	N/A	N/A	N/A	N/A	Input
Registry_Name	N/A	N/A	N/A	N/A	N/A	N/A	Input

Usage notes

1. This service is intended for use by z/OS application servers. It allows them to map between supported application user identities and the corresponding RACF user ID, or to determine the RACF user ID by supplying the corresponding application user identity or digital certificate.
2. An **ALET** must be specified for the SAF_return_code, RACF_return_code, RACF_reason_code parameters, and a single **ALET** specified for all of the remaining parameters.
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 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 1-6, the first parameter that can have the high-order bit on, ending the parameter list, is the Application_userid parameter.
   - For function code 8, the first parameter that can have the high-order bit on, ending the parameter list, is the Registry_Name parameter.
4. If the Function_code indicates that an application identity is to be returned, and no RACF_userid is supplied, the caller receives a parameter list error.
5. For function codes 1-6:
   - The caller receives a parameter list error if the Function_code indicates that a RACF user ID is to be returned, and no Application_userid or Certificate is supplied.
   - Specification of a RACF_userid with a length greater than 8 or an Application_userid with a length greater than 246 will result in a parameter list error.
   - If the Function_code specifies that a RACF user ID is to be returned and the length supplied for the Application_userid is greater than the maximum allowed, such as greater than 64 for a Lotus Notes for z/OS user identity, or greater than 240 for Security Server Network Authentication Service user identity, the caller receives the "no mapping between RACF and an application" error.
   - If the Function_code indicates that a RACF user ID is to be returned, and both an Application_userid and a Certificate are supplied, the Application_userid will be used.
6. For function code 8:
   - The length of the Distinguished_Name must be greater than 0 and less than or equal to the maximum length allowed by the RCVT field RCVTDNL (currently 246). A length of 0 or greater than the maximum allowed will result in the, Distinguished Name length not valid, error (**SAF Return Code = 8, RACF Return Code = 8, RACF Reason Code = 40**).
   - The length of the Registry_Name must be greater than 0 and less than or equal to the maximum length allowed by the RCVT field RCVTRL (currently 255). A length of 0 or greater than the maximum allowed will result in the, Registry Name length not valid, error (**SAF Return Code = 8, RACF Return Code = 8, RACF Reason Code = 44**).
   - The Distinguished_Name and Registry_Name must be in UTF-8 format. If they are not in UTF-8 format the associated RACF user ID will not be found (**SAF Return Code = 8, RACF Return Code = 8, RACF Reason Code = 48**).
7. Specification of an unknown function code will result in a parameter list error.
8. If the Function_code indicates that an application user identity is to be returned, the caller is expected to supply a 248-byte area for the Application_userid parameter. If an application user identity is defined for the RACF_userid specified, R_usermap will update this area with the length and value of the application user identity. This storage must be accessible in the caller's key.
9. If the Function_code indicates that a RACF user ID is to be returned, the caller is expected to supply a 9-byte area for the RACF_userid parameter. If a RACF user ID is associated with the Application_identity specified, R_usermap will update this area with the length and value of the RACF user ID. This storage must be accessible in the caller's key.

10. The conversion between the supported application user identities (or certificates) and RACF user IDs is dependent on the definition of the application-specific segments associated with the RACF USER profile.
    - To convert between Lotus Notes for z/OS user identity and a RACF user ID, the RACF USER profile must have an LNOTES segment containing the SNAME field.
    - To convert between an NDS for z/OS user identity and a RACF user ID, the RACF USER profile must have an NDS segment containing the UNAME field.
    - To convert between a certificate and a RACF user ID, the RACF USER profile must be associated with the certificate.

    Note: In the case of Security Server Network Authentication Service identities, local Security Server Network Authentication Service principals are similar to the above: to convert between a Security Server Network Authentication Service local principal identity and a RACF user ID, the RACF USER profile must have a KERB segment containing the KERBNAME field. However, in the case of foreign Security Server Network Authentication Service principals, a KERBLINK class profile must be defined to map the foreign Security Server Network Authentication Service principal to a RACF user identity. The RACF user identity associated with a foreign Security Server Network Authentication Service principal (or multiple foreign Security Server Network Authentication Service principals), does not require a KERB segment.
11. The certificate supplied by the certificate parameter is used only to identify a RACF user ID. It is expected that the certificate was previously verified. Note the following additional details regarding certificate processing:
    a. All fields as defined for X.509 version 1 certificates must be present and non-null.
    b. X.509 certificates with version numbers greater than 3 are not supported.
    c. Version 3 certificates with critical extensions are not supported. Noncritical extensions are ignored.
    d. Subject and issuer names can contain only the following string types:
        - T61STRING - TAG 20
        - PRINTABLESTRING - TAG 19
        - IA5STRING - TAG 22
        - VISIBLESTRING - TAG 26
        - GENERALSTRING - TAG 27
    e. The length of the serial number plus the length of the issuer's name cannot exceed 245.
    f. No date validity check is performed on the certificate.
    g. No signature check is performed on the certificate.
12. If the certificate supplied by the caller is defined to RACF with a status of NOTRUST, R_usermap will return a RACF return code 8, RACF reason code 32, indicating that no user ID is defined to use this certificate.
13. For function_codes 5 and 6, the application user identity is the Security Server Network Authentication Service principal name. Local principal name is case-sensitive, foreign principal is not (the RDEFINE of a KERBLINK profile will fold the name to uppercase). R_usermap will accept mixed case input of foreign profile names, but will fold to uppercase before attempting to locate the appropriate KERBLINK identity mapping profile. R_usermap output of foreign principal names will always be uppercase. Additionally, while local

principal names may be supplied fully qualified with the name of the local realm, R_usermap output of local principal names will always be unqualified. Realm qualified names follow a DCE-like convention of /.../realm_name/principal_name.

14. The Distinguished Name value can be in any of the following formats:
    a. As a simple character string, such as a user ID defined in a non-LDAP registry.

       Typically, special characters do not appear in user names stored within a registry. However, if you need to specify a user name value that includes certain characters, they must be preceded by the backslash (\) escape character.

       These characters include the plus sign (+), semicolon (;), comma (,), quotation mark ("), backslash (\), less than symbol (<), greater than symbol (>) and the equal sign (=).

    b. As a character string that represents an X.500 distinguished name (DN).

       A Distinguished Name (DN) consists of one or more relative distinguished names (RDNs). Each RDN consists of an attribute type and attribute value, separated by an equal sign (=). RDNs are separated by a comma (,).

       Like the RACMAP command, R_usermap does not perform validity checking of the X.500 name.
15. The following rules are used to define a user name as Distinguished Name when the RACMAP command is used to define the mappings:
    - Specify the user name value in its canonical form, as it is defined within the registry, with any special characters preceded by the backslash (\) escape character. You must specify the RDNs in their correct sequence.

      For example, for users of WebSphere® Application Server applications, the canonical form of the user name must match the value returned by the WSCredential interface method called getUniqueSecurityName().
    - Typically, special characters do not appear in user names stored within a registry. However, if you need to specify a user name value that includes certain characters, including LDAP special characters, they must be preceded by the backslash (\) escape character.

      These characters include the plus sign (+), semicolon (;), comma (,), quotation mark ("), backslash (\), less than symbol (<), greater than symbol (>), and the equal sign (=).

      Exception: Do not escape the equal sign (=), semicolon (;), or comma (,) when you specify them as delimiters of an RDN.
    - Do not specify blank characters immediately preceding or following the equal sign (=) when using the equal sign as a delimiter of an attribute type or RDN.
16. Check if any logrec entry has been created to ensure R_usermap service was being run successfully and also refer to *z/OS Security Server RACF Diagnosis Guide* for detailed logrec information.

Related services

R_dceinfo, R_dceruid

R_writepriv (IRRSWP00): Write-down privilege

Function

The **R_writepriv** service sets, resets, or queries the setting of the write-down privilege in the ACEE. Only callers in system key or supervisor state can change the value in an ACEE other than the current address space level ACEE.

Requirements

Authorization:
Any PSW key in any state

Dispatchable unit mode:
Task

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
31

RMODE:
Any

ASC mode:
Primary or AR mode

Recovery mode:
SETFRR

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area must be in the primary address space. ALETs must be passed for all parameters except the work area and function code. The words containing the ALETs must be in the primary address space.

RACF authorization

To specify an ACEE value, the calling program must be running in a system key or in supervisor state. If the ACEE is not specified, the address space level ACEE is used. In either case, the ability to enable the write-down privilege is determined using the IRR.WRITEDOWN.BYUSER profile in the FACILITY class based on the user ID in the ACEE. The FACILITY class must be active and RACLISTed, the IRR.WRITEDOWN.BYUSER profile must exist in the RACLISTed profiles, and the SETR MLS option must be active.

Format

```
CALL IRRSWP00 (Work_area,
     ALET, SAF_return_code,
     ALET, RACF_return_code,
     ALET, RACF_reason_code,
     Function_Code,
     ALET, ACEE
    )
```

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 ALET can be different. The words 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.

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.

Function_Code
The name of a 1-byte area containing the function code.

X'00' query the current setting of the write-down privilege

X'01' activate the write-down privilege

X'02' inactivate the write-down privilege

X'03' reset the write-down privilege to its default value

ACEE
The name of an area containing an ACEE or a fullword of zeros if ACEE is not specified. If not specified, the home address space level ACEE is to be used.

Return and reason codes

IRRSWP00 returns the following values in the reason and return code parameters:

SAF return code	RACF return code	RACF reason code	Explanation
0	0	0	The service was successful, and the write-down privilege is active for this ACEE.
0	0	4	The service was successful, and the write-down privilege is inactive for this ACEE.
4	0	0	RACF is not installed.
4	4	0	Write-down by user is not active on this system.
8	8	0	The user is not authorized.
8	8	4	Input parameter list error.
8	8	8	Function not supported for problem state caller.
8	8	12	An internal error occurred during RACF processing.

Usage notes

1. If the SAF and RACF return codes are zero, the RACF reason code will indicate the current setting of the write-down privilege when the callable service completes.
2. An audit record is written if the write-down privilege is set successfully, or if a request has been made to change the write-down privilege and the user is not authorized to IRR.WRITEDOWN.BYUSER.

Related services

None

Chapter 3. Installation exits

IRRSXT00 installation exit, for 24 or 31 bit callers, uses the IRRSXT00 module as described in this chapter. IRRSXT0X installation exit, for 64 bit callers, uses IRRSXT0X module.

Function

As described in Chapter 1, "Using the RACF callable services," on page 1, Linkage Conventions for the Callable Services, IRRSXT00 is invoked by the SAF callable services router before and after RACF is called. It receives as input, a function code indicating which callable service is being called, and the parameter list that will be passed to RACF. The first parameter in the parameter list points to a work area. The exit can use the first 152 bytes of this work area. The first word of the work area is set to zero before the pre-RACF call to the exit. the exit should set another value in this word to indicate to the post-RACF exit call that it is the second call. The first four words of the work area are passed unchanged from the pre-RACF to the post-RACF exit.

The pre-RACF exit can change the content of the parameter list that will be passed to the external security product. It can also indicate with return codes that the external security product should be bypassed and control returned to the caller. The SAF return code is set based on the exit return code. If the external security product is bypassed, the exit routine must provide all of the output including RACF-compatible return and reason codes that the invokers of the services expect.

The post-RACF exit can look at or change the output from RACF including the RACF return and reason codes. No exit return codes are defined from this exit call. SAF return codes are set based on the RACF return codes, not on an exit return code.

Requirements

Authorization:
*State and key of the user calling the security function

Dispatchable unit mode:
Task of user calling security function

Cross memory mode:
PASN = HASN or PASN not = HASN

AMODE:
Any *or* 31 *or* 64 for IRRSXT0X

RMODE:
Any *or* 24

ASC mode:
AR mode

Serialization:
Enabled for interrupts

Locks: No locks held

Control parameters:
The parameter list and the work area are in the primary address space. ALETs are passed for all parameters except the work area. The words containing the ALETs are in the primary address space.

Note: *Most callable services will be supervisor state and key 0. For services that can be invoked unauthorized, it can be problem state and any key.

Interface registers

Table 127. Input registers

Register	AR content	GR content
0	Any	Function code of service. Refer to the description of IRRPFC in *z/OS Security Server RACF Data Areas*.
1	0	Address of parameter list
2 - 13	Any	Undefined
14	0	Return address
15	0	Entry point address

Table 128. Output registers

Register	AR content	GR content
0 - 14	Same as input	Same as input
15	Undefined	Return code

Input

The parameter list is the same as the list that will be passed to the external security product (for example, RACF). The content of the list varies depending on the service requested. See *z/OS Security Server RACF Data Areas* for descriptions of the parameter list, IRRPCOMP, as well as detailed descriptions of structures such as the FSP and CRED, which are passed as parameters on a number of the callable services. See Chapter 2, "Callable services descriptions," on page 9 for descriptions of the possible parameter lists. IRRSXT0X uses IRRPCOMX for parameter mapping.

The first 152 bytes of the work area pointed to by the parameter list can be used by the exit. The rest of the work area is reserved for SAF and RACF. The first four words of the 152-byte area can be used to pass data from the pre-RACF call of the exit to the post-RACF call to the exit. These words are not used by SAF or RACF. The other 136 bytes may be used by RACF services or the SAF router between the calls to the pre-RACF exit and the post-RACF exit.

Output

Returned data: If the exit indicates that RACF is not to be called, the exit is responsible for setting the RACF return and reason codes and providing any other output data expected by the caller of the requested service.

Refer to Chapter 2, “Callable services descriptions,” on page 9 for information on which callable services return output to their callers, as well as the format of the output.

Note: The return and reason codes are not stored in the parameter list as they are for SAF exit ICHRTX00. For IRRSXT00, the parameter list contains the addresses of two words in which the return and reason codes must be stored.

The pre-RACF exit routine must restore all registers on return except register 15, which must contain a return code.

The post-RACF exit must also restore all registers except 15. No return codes are defined for the post-RACF exit.

Return codes: The pre-RACF exit can return one of the following return codes:

Return code	Explanation
0	Exit complete - continue processing and call RACF for further security processing. The exit routine may change the content of the parameter list that will be passed to RACF
200	Exit complete - access authorized. The SAF callable services router sets SAF router return code 0 and returns to the caller of the service, bypassing any further security processing.
204	Exit complete - no decision. The SAF callable services router sets SAF return code 4 and returns to the caller of the service, bypassing any further security processing.
208	Exit complete - access not authorized. The SAF callable services router sets SAF return code 8 and returns to the caller of the service, bypassing any further security processing.
Other	Exit complete - the SAF callable services router sets the SAF return code to the exit-supplied value and returns to the caller of the service, bypassing any further security processing.

Usage notes

1. The exit must be reentrant.
2. The exit can receive control in cross memory mode.
3. The exit must use BAKR to save registers. No save area is provided on entry. It is called in AR mode and must save both general and access registers.
4. To install the SAF callable services router installation exit, create the load module, name it IRRSXT00, and load it into the link pack area (LPA).
5. The exit must provide its own recovery routine. If the exit routine terminates abnormally, its recovery routine gets control. If a recovery routine is not provided or if the recovery routine percolates, the recovery routine of the caller of the service stub will get control.

6. To determine the caller of the SAF callable service, use the function code in register 0 to determine which callable service is being called. Refer to Table 1 on page 9, in Chapter 2, for the list of components and products that are possible callers of each service.

   The usage notes that follow the callable service descriptions are also helpful in determining the caller.

   Many services receive the CRED as a parameter. The CRED contains an audit function code, defined in macro IRRPAFC, which identifies the z/OS UNIX function calling the service. You can find additional information on which callable services are called by z/OS UNIX functions in the z/OS Security Server (RACF) Auditor's Guide, under Classes that Control Auditing for z/OS UNIX System Services.

Appendix A. R_admin reference information

Segment and field entry mappings

For an overview of how segment and field entries are used, see "R_admin update functions" on page 83.

For the **R_admin** update functions, the caller provides segment and field entries to identify the segment(s) and field(s) which should be added, altered, deleted, or listed. For SETROPTS extract, RACF returns a BASE segment entry and field entries in the same way.

A segment entry is mapped as follows:

Table 129. Segment entry mapping

Offset	Length	Description
0	8	Profile segment name (left-justified, uppercase, and padded with blanks). The segment name must be one of the documented segments for the profile type in question.
8	1	Flag byte. See Table 130 for information on flag usage.
9	2	Number of field entries within the segment.
11	*	First field entry for the segment.

For user and group delete functions, no segment data is expected. The number of segments should be zero. If non-zero, any segment data present is ignored.

For list functions, the caller must provide a segment entry for each segment which is to be listed (including the BASE segment). The command output for the segments is returned in the order in which the segment entries were supplied, except that the BASE segment is always returned first.

The following table describes the flag byte usage:

Table 130. Flag byte usage

Flag byte value	Description
"Y"	• For add functions, create the segment • For alter functions, create or modify the segment • For list functions, display the segment
"N"	• For add functions, do not create the segment (This usage is optional. You can simply omit the segment entry.) • For alter functions, delete the segment
"I"	• For add and alter functions, ignore this segment entry and any field entries within it • For resource list functions, ignore this segment entry (do not display the segment)

Note:

- For a value of "N", and for user and group list functions, no field data is expected, and the number of fields specified in the segment entry should be zero. If non-zero, any field data present is ignored.
- The flag byte is ignored for the BASE segment.
- The flag byte is ignored for all delete functions.
- For some uses of "Y", you are not necessarily creating or altering a segment, but are simply providing the segment entry as a means to contain field entries. For example, the resource functions require a BASE segment entry containing, at a minimum, a field entry for the PROFILE field to identify the target profile name. This usage is highlighted on the resource delete and list functions. Here, some fields correspond to command keywords but not necessarily to database fields (For example, the GENERIC "field" for data sets, or the AUTHUSER "field" for general resources and data sets).

Each field entry is mapped as in Table 131.

Table 131. Field entry mapping

Offset	Length	Description
0	8	Field name as defined in the tables for each profile type. All field names must be left-justified, entered in all uppercase, and padded with blanks.
8	1	Flag byte. See Table 132 for flag usage.
9	2	Length of field data.
11	*	Field data.

Note: Repeating data is specified as blank or comma delimited character strings.

The following table describes the flag byte usage:

Table 132. Flag byte usage

Flag byte value	Boolean field descriptions	Text field descriptions	Repeating text field descriptions
"Y"	Set the value to TRUE	Create or modify the field with the specified data	Create or replace the field with the specified data
"A"	N/A	N/A	Add the specified data to the existing data, if any
"D"	N/A	N/A	Delete the specified data from the existing data
"N"	Set the value to FALSE	For alter functions, delete the field. Otherwise, N/A.	Remove all data from the field
"I"	Ignore the field entry		

Note:

- For boolean fields, no field data is allowed. Coding field data results in an "Invalid Request" error being returned to the caller.
- For flag byte 'N', any field data specified is ignored.

- For flag byte 'A', field data must be specified. If field data is not specified, an "Invalid Request" error is returned to the caller.

Reference documentation tables

The following are **R_admin** tables by section:

User administration

The following tables define field names and their usage. All field names relate directly to the ADDUSER and ALTUSER keywords. Although the fields are alphabetized in the following tables, there is no defined order in which the fields are returned when using the extract functions. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean and list fields are identified in the field name column. Unless otherwise noted, a field is a character field by default. For list fields, the list header field returned by the extract function is also specified. Unless otherwise noted, all list fields are 1-dimensional arrays.

Table 133. BASE segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword Reference, or LISTUSER heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ADSP (boolean)	'Y'	ADSP	Yes	Yes	Yes
	'N'	NOADSP	Yes	Yes	Yes
AUDITOR (boolean)	'Y'	AUDITOR	Yes	Yes	Yes
	'N'	NOAUDITOR	Yes	Yes	Yes
AUTH	'Y'	AUTHORITY (xx)	Yes	Yes	No
CATEGORY (list NUMCTGY)	'Y'	ADDCATEGORY(xx ...)	Yes	No	Yes
	'A'	ADDCATEGORY(xx ...)	No	Yes	Yes
	'D'	DELCATEGORY(xx ...)	No	Yes	Yes
NOTES: • To remove unknown categories from the profile, specify the 'D' flag and a field length of zero. • On output, if a category cannot be mapped back to its external name, the string "-UNKNOWN-" is returned for the category value.					
CLAUTH (list CLCNT)	'Y'	CLAUTH(xx...)	Yes	No	Yes
	'A'	CLAUTH(xx ...)	No	Yes	Yes
	'D'	NOCLAUTH(xx ...)	No	Yes	Yes
	'N'	NOCLAUTH	Yes	No	Yes
CONNECTS	N/A	**Note:** This is the list header field for the 15-dimensional array consisting of the following fields	No	No	Yes
CADSP (boolean)	N/A	N/A. These fields correspond to CONNECTS command keywords	No	No	Yes
CAUDITOR	N/A	N/A	No	No	Yes
CAUTHDA	N/A	N/A	No	No	Yes

Table 133. BASE segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword Reference, or LISTUSER heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CGROUP	N/A	N/A	No	No	Yes
CGRPACC	N/A	N/A	No	No	Yes
CINITCT	N/A	N/A	No	No	Yes
CLJDATE	N/A	N/A	No	No	Yes
CLJTIME	N/A	N/A	No	No	Yes
COPER	N/A	N/A	No	No	Yes
COWNER	N/A	N/A	No	No	Yes
CRESUME	N/A	N/A	No	No	Yes
CREVOKE (This is the revoke date. See CREVOKFL field for boolean output.)	N/A	N/A	No	No	Yes
CREVOKFL (boolean)	N/A	N/A	No	No	Yes
CSPECIAL (boolean)	N/A	N/A	No	No	Yes
CUACC	N/A	N/A	No	No	Yes
CREATDAT	N/A	CREATED=	No	No	Yes
DATA	'Y'	DATA (xx)	Yes	Yes	Yes
	'N'	NODATA	No	Yes	Yes
DFLTGRP	'Y'	DFLTGRP (xx)	Yes	Yes	Yes
EXPIRED (boolean)	'Y'	EXPIRED	No	Yes	No
	'N'	NOEXPIRED	No	Yes	No
GROUP	'Y'	GROUP (xx)	No	Yes	No
GRPACC (boolean)	'Y'	GRPACC	Yes	Yes	Yes
	'N'	NOGRPACC	Yes	Yes	Yes
HASPHRAS (boolean)	N/A	ATTRIBUTES=	No	No	Yes
HASPWD (boolean)	N/A	ATTRIBUTES=	No	No	Yes
LASTDATE	N/A	LAST-ACCESS= (left side of "/")	No	No	Yes
LASTTIME	N/A	LAST-ACCESS= (right side of "/")	No	No	Yes
MODEL	'Y'	MODEL (xx)	Yes	Yes	Yes
	'N'	NOMODEL	No	Yes	Yes
NAME	'Y'	NAME (xx)	Yes	Yes	Yes
	'N'	NAME	No	Yes	Yes

Table 133. BASE segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword Reference, or LISTUSER heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
OIDCARD (boolean)	'Y'	OIDCARD	No	No	No
	'N'	NOOIDCARD	Yes	Yes	No
OPER (boolean)	'Y'	OPERATIONS	Yes	Yes	Yes
	'N'	NOOPERATIONS	Yes	Yes	Yes
OWNER	'Y'	OWNER(xx)	Yes	Yes	Yes
PASSDATE	N/A	PASSDATE=	No	No	Yes
PASSINT	N/A	PASS-INTERVAL=	No	No	Yes
PASSWORD	'Y'	PASSWORD (xx)	Yes	Yes	No
	'N'	NOPASSWORD	Yes	Yes	No
PHRASE	'Y'	PHRASE (xx)	Yes	Yes	No
	'N'	NOPHRASE	No	Yes	No
PHRDATE	N/A	PHRASEDATE=	No	No	Yes
PPHENV (boolean)	N/A	PHRASE ENVELOPED=	No	No	Yes
PROTECTD (boolean)	N/A	ATTRIBUTES=	No	No	Yes
PWDENV (boolean)	N/A	PASSWORD ENVELOPED=	No	No	Yes
REST (boolean)	'Y'	RESTRICTED	Yes	Yes	Yes
	'N'	NORESTRICTED	Yes	Yes	Yes
RESUME (on output, this is the resume date)	'Y'	RESUME(xx)	No	Yes	Yes
	'N'	NORESUME	No	Yes	Yes
REVOKE (on output, this is the revoke date. See REVOKEFL for boolean 'revoked' bit)	'Y'	REVOKE(xx)	No	Yes	Yes
	'N'	NOREVOKE	No	Yes	Yes
REVOKEFL (boolean)	N/A	ATTRIBUTES=	No	No	Yes
NOTE: The value of the REVOKEFL field is consistent with the behavior of the LISTUSER command in that it takes revoke and resume dates into account when determining if the user is revoked.					
SECLABEL	'Y'	SECLABEL (xx)	Yes	Yes	Yes
	'N'	NOSECLABEL	No	Yes	Yes
SECLEVEL	'Y'	SECLEVEL (xx)	Yes	Yes	Yes
	'N'	NOSECLEVEL	No	Yes	Yes
SPECIAL (boolean)	'Y'	SPECIAL	Yes	Yes	Yes
	'N'	NOSPECIAL	Yes	Yes	Yes
UACC	'Y'	UACC (xx)	Yes	Yes	No
UAUDIT (boolean)	'Y'	UAUDIT	No	Yes	Yes
	'N'	NOUAUDIT	No	Yes	Yes

Table 133. BASE segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword Reference, or LISTUSER heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
WHENDAYS (list WHENDYCT)	'Y'	WHEN(DAYS (xx))	Yes	Yes	Yes
WHENTIME	'Y'	WHEN(TIME (xx))	Yes	Yes	Yes

Table 134. CICS segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
OPCLASS (list OPCLASSN)	'Y'	CICS(OPCLASS (xx ...))	Yes	Yes	Yes
	'A'	CICS(ADDOPCLASS (xx ...))	No	Yes	Yes
	'D'	CICS(DELOPCLASS (xx ...))	No	Yes	Yes
	'N'	CICS(NOOPCLASS)	No	Yes	Yes
OPIDENT	'Y'	CICS(OPIDENT (xx))	Yes	Yes	Yes
	'N'	CICS(NOOPIDENT)	No	Yes	Yes
OPPRTY	'Y'	CICS(OOPPRTY (xx))	Yes	Yes	Yes
	'N'	CICS(NOOPPRTY)	No	Yes	Yes
RSLKEY (list RSLKEYN)	'Y'	CICS(RSLKEY(xx...))	Yes	Yes	Yes
	'N'	CICS(NORSLKEY)	No	Yes	Yes
TIMEOUT	'Y'	CICS(TIMEOUT (xx))	Yes	Yes	Yes
	'N'	CICS(NOTIMEOUT)	No	Yes	Yes
TSLKEY (list TSLKEYN)	'Y'	CICS(TSLKEY(xx...))	Yes	Yes	Yes
	'N'	CICS(NOTSLKEY)	No	Yes	Yes
XRFSOFF	'Y'	CICS(XRFSOFF (xx))	Yes	Yes	Yes
	'N'	CICS(NOXRFSOFF)	No	Yes	Yes

Table 135. CSDATA segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
custom-keyword	'Y'	CSDATA (custom-keyword(xxx))	Yes	Yes	Yes
	'N'	CSDATA (NOcustom-keyword)	No	Yes	
A custom field (keyword and value type) is installation specific. On extract requests the field descriptor mapping will contain the *custom-keyword* and the associated data value.					

Table 136. DCE segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
AUTOLOG (boolean)	'Y'	DCE(AUTOLOGIN (YES))	Yes	Yes	Yes
	'N'	DCE(AUTOLOGIN(NO))	Yes	Yes	

Table 136. DCE segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DCENAME	'Y'	DCE(DCENAME(xx))	Yes	Yes	Yes
	'N'	DCE(DCENAME)	No	Yes	
HOMECELL	'Y'	DCE(HOMECELL (xx))	Yes	Yes	Yes
	'N'	DCE(NOHOMECELL)	No	Yes	
HOMEUUID	'Y'	DCE(HOMEUUID (xx))	Yes	Yes	Yes
	'N'	DCE(NOHOMEUUID)	No	Yes	
UUID	'Y'	DCE(UUID(xx))	Yes	Yes	Yes
	'N'	DCE(NOUUID)	No	Yes	

Table 137. DFP segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DATAAPPL	'Y'	DFP(DATAAPPL(xx))	Yes	Yes	Yes
	'N'	DFP(NODATAAPPL)	No	Yes	
DATACLAS	'Y'	DFP(DATACLAS(xx))	Yes	Yes	Yes
	'N'	DFP(NODATACLAS)	No	Yes	
MGMTCLAS	'Y'	DFP(MGMTCLAS(xx))	Yes	Yes	Yes
	'N'	DFP(NOMGMTCLAS)	No	Yes	
STORCLAS	'Y'	DFP(STORCLAS(XX))	Yes	Yes	Yes
	'N'	DFP(NOSTORCLAS)	No	Yes	

Table 138. EIM segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
LDAPPROF	'Y'	EIM(LDAPPROF(xx))	Yes	Yes	Yes
	'N'	EIM(NOLDAPPROF)	No	Yes	

Table 139. KERB segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference, or LISTUSER heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ENCRYPT (list ENCRYPTN)	'Y'	KERB(ENCRYPT(xx))	Yes	Yes	Yes
	'N'	KERB(NOENCRYPT)	No	Yes	
KERBNAME	'Y'	KERB(KERBNAME(xx))	Yes	Yes	Yes
	'N'	KERB(NOKERBNAME)	No	Yes	
KEYFROM	N/A	KEY FROM=	No	No	Yes
KEYVERS	N/A	KEY VERSION=	No	No	Yes
MAXTKTLF	'Y'	KERB(MAXTKTLFE(xx))	Yes	Yes	Yes
	'N'	KERB(NOMAXTKTLFE)	No	Yes	

Table 140. Language segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
PRIMARY	'Y'	LANGUAGE (PRIMARY(xx))	Yes	Yes	Yes
	'N'	LANGUAGE(NOPRIMARY)	No	Yes	
SECOND	'Y'	LANGUAGE(SECONDARY(xx))	Yes	Yes	Yes
	'N'	LANGUAGE(NOSECONDARY)	No	Yes	

Table 141. LNOTES segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
SNAME	'Y'	LNOTES(SNAME (xx))	Yes	Yes	Yes
	'N'	LNOTES(NOSNAME)	No	Yes	

Table 142. NDS segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
UNAME	'Y'	NDS(UNAME (xx))	Yes	Yes	Yes
	'N'	NDS(NOUNAME)	No	Yes	

Table 143. NetView® segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CONSNAME	'Y'	NETVIEW(CONSNAME (xx))	Yes	Yes	Yes
	'N'	NETVIEW(NOCONSNAME)	No	Yes	
CTL	'Y'	NETVIEW(CTL (xx))	Yes	Yes	Yes
	'N'	NETVIEW(NOCTL)	No	Yes	
DOMAINS (list DOMAINSN)	'Y'	NETVIEW(DOMAINS (xx ...))	Yes	Yes	Yes
	'A'	NETVIEW(ADDDOMAINS (xx ...))	No	Yes	
	'D'	NETVIEW(ADDDOMAINS (xx ...))	No	Yes	
	'N'	NETVIEW(ADDDOMAINS (xx ...))	No	Yes	
IC	'Y'	NETVIEW(IC (xx))	Yes	Yes	Yes
	'N'	NETVIEW(NOIC)	No	Yes	
MSGRECVR (boolean)	'Y'	NETVIEW(MSGRECVR (YES))	Yes	Yes	Yes
	'N'	NETVIEW(MSGRECVR (NO))	Yes	Yes	
NGMFADMN (boolean)	'Y'	NETVIEW(NGMFADMN(YES))	Yes	Yes	Yes
	'N'	NETVIEW(NGMFADMN (NO))	No	Yes	
NGMFVSPN	'Y'	NETVIEW(NGMFVSPN (xx))	Yes	Yes	Yes
	'N'	NETVIEW(NONGMFVSPN)	No	Yes	

Table 143. NetView® segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
OPCLASS (list OPCLASSN)	'Y'	NETVIEW(OPCLASS (xx ...))	Yes	Yes	Yes
	'A'	NETVIEW(ADDOPCLASS (xx ...))	No	Yes	
	'D'	NETVIEW(DELOPCLASS (xx ...))	No	Yes	
	'N'	NETVIEW(NOOPCLASS)	No	Yes	

Table 144. OMVS segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ASSIZE	'Y'	OMVS(ASSIZEMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOASSIZEMAX)	No	Yes	
AUTOUID	'Y'	OMVS(AUTOUID)	Yes	Yes	No
CPUTIME	'Y'	OMVS(CPUTIMEMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOCPUTIMEMAX)	No	Yes	
FILEPROC	'Y'	OMVS(FILEPROCMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOFILEPROCMAX)	No	Yes	
HOME	'Y'	OMVS(HOME (xx))	Yes	Yes	Yes
	'N'	OMVS(NOHOME)	No	Yes	
MEMLIMIT	'Y'	OMVS(MEMLIMIT(xx))	Yes	Yes	Yes
	'N'	OMVS(NOMEMLIMIT))	No	Yes	
MMAPAREA	'Y'	OMVS(MMAPAREAMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOMMAPAREAMAX)	No	Yes	
PROCUSER	'Y'	OMVS(PROCUSERMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOPROCUSERMAX)	No	Yes	
PROGRAM	'Y'	OMVS(PROGRAM (xx))	Yes	Yes	Yes
	'N'	OMVS(NOPROGRAM)	No	Yes	
SHARED	'Y'	OMVS(SHARED)	Yes	Yes	No
SHMEMMAX	'Y'	OMVS(SHMEMMAX(xx))	Yes	Yes	Yes
	'N'	OMVS(NOSHMEMMAX))	No	Yes	
THREADS	'Y'	OMVS(THREADSMAX (xx))	Yes	Yes	Yes
	'N'	OMVS(NOTHREADSMAX)	No	Yes	
UID	'Y'	OMVS(UID (xx))	Yes	Yes	Yes
	'N'	OMVS(NOUID)	No	Yes	

Table 145. OPERPARM segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ALTGRP	'Y'	OPERPARM (ALTGRP(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOALTGRP)	No	Yes	

Table 145. OPERPARM segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
AUTO	'Y'	OPERPARM(AUTO(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOAUTO))	No	Yes	
CMDSYS	'Y'	OPERPARM(CMDSYS (xx))	Yes	Yes	Yes
	'N'	OPERPARM (NOCMDSYS)	No	Yes	
DOM	'Y'	OPERPARM(DOM xx))	Yes	Yes	Yes
	'N'	OPERPARM (NODOM)	No	Yes	
HC	'Y'	OPERPARM(HC(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOHC)	No	Yes	
INTIDS	'Y'	OPERPARM(INTIDS(xx))	Yes	No	Yes
	'N'	OPERPARM(NOINTIDS)	No	Yes	
KEY	'Y'	OPERPARM(KEY(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOKEY)	No	Yes	
LEVEL (list LEVELN)	'Y'	OPERPARM(LEVEL (xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOLEVEL)	No	Yes	
LOGCMD	'Y'	OPERPARM(LOGCMDRESP(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOLOGCMDRESP)	No	Yes	
MFORM (list MFORMN)	'Y'	OPERPARM(MFORM(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOMFORM)	No	Yes	
MIGID	'Y'	OPERPARM(MIGID(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOMIGID)	No	Yes	
MONITOR (list MONITORN)	'Y'	OPERPARM(MONITOR(xx...))	Yes	Yes	Yes
	'N'	OPERPARM(NOMONITOR)	No	Yes	
MSCOPE (list MSCOPEN)	'Y'	OPERPARM(MSCOPE(xx ...))	Yes	Yes	Yes
	'A'	OPERPARM(ADDMSCOPE(xx ...))	No	Yes	
	'D'	OPERPARM(DELMSCOPE(xx ...))	No	Yes	
	'N'	OPERPARM(NOMSCOPE)	No	Yes	
On extract requests, if the value in the ROUTCODE field is "ALL", then the list header field will describe one entry with the value "ALL", as opposed to an enumeration of all 128 route code values.					
OPERAUTH (list OPERAUTN)	'Y'	OPERPARM(AUTH(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOAUTH)	No	Yes	
ROUTCODE (list ROUTCODN)	'Y'	OPERPARM(ROUTCODE(xx ...)	Yes	Yes	Yes
	'N'	OPERPARM(NOROUTCODE)	No	Yes	
STORAGE	'Y'	OPERPARM(STORAGE(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOSTORAGE)	No	Yes	
UD	'Y'	OPERPARM(UD (xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOUD)	No	Yes	
UNKNIDS	'Y'	OPERPARM(UNKNIDS(xx))	Yes	Yes	Yes
	'N'	OPERPARM(NOUNKNIDS)	No	Yes	

Table 146. OVM segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
FSROOT	'Y'	OVM(FSROOT (xx))	Yes	Yes	Yes
	'N'	OVM(NOFSROOT)	No	Yes	
VHOME	'Y'	OVM(HOME(xx))	Yes	Yes	Yes
	'N'	OVM(NOHOME)	No	Yes	
VPROGRAM	'Y'	OVM(PROGRAM(xx))	Yes	Yes	Yes
	'N'	OVM(NOPROGRAM)	No	Yes	
VUID	'Y'	OVM(UID(xx))	Yes	Yes	Yes
	'N'	OVM(NOUID)	No	Yes	

Table 147. PROXY segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract Requests
BINDDN	'Y'	PROXY(BINDDN(xx))	Yes	Yes	Yes
	'N'	PROXY(NOBINDDN)	No	Yes	
BINDPW	'Y'	PROXY(BINDPW(xx))	Yes	Yes	No
	'N'	PROXY(NOBINDPW)	No	Yes	
LDAPHOST	'Y'	PROXY(LDAPHOST(xx))	Yes	Yes	Yes
	'N'	PROXY(NOLDAPHOST)	No	Yes	

Table 148. TSO segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ACCTNUM	'Y'	TSO(ACCTNUM (xx))	Yes	Yes	Yes
	'N'	TSO (NOACCTNUM)	No	Yes	
COMMAND	'Y'	TSO(COMMAND (xx))	Yes	Yes	Yes
	'N'	TSO(NOCOMMAND)	No	Yes	
DEST	'Y'	TSO (DEST (xx))	Yes	Yes	Yes
	'N'	TSO(NODEST)	No	Yes	
HLDCLASS	'Y'	TSO(HOLDCLASS (xx))	Yes	Yes	Yes
	'N'	TSO(NOHOLDCLASS)	No	Yes	
JOBCLASS	'Y'	TSO(JOBCLASS (xx))	Yes	Yes	Yes
	'N'	TSO(NOJOBCLASS)	No	Yes	
MAXSIZE	'Y'	TSO(MAXSIZE (xx))	Yes	Yes	Yes
	'N'	TSO(NOMAXSIZE)	No	Yes	
MSGCLASS	'Y'	TSO(MSGCLASS (xx))	Yes	Yes	Yes
	'N'	TSO(NOMSGCLASS)	No	Yes	
PROC	'Y'	TSO(PROC(xx))	Yes	Yes	Yes
	'N'	TSO(NOPROC)	No	Yes	

Table 148. TSO segment fields (continued)

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
SECLABEL	'Y'	TSO(SECLABEL (xx))	Yes	Yes	Yes
	'N'	TSO(NOSECLABEL)	No	Yes	
SIZE	'Y'	TSO(SIZE (xx))	Yes	Yes	Yes
	'N'	TSO(NOSIZE)	No	Yes	
SYSOUTCL	'Y'	TSO(SYSOUTCL (xx))	Yes	Yes	Yes
	'N'	TSO(NOSYSOUTCL)	No	Yes	
UNIT	'Y'	TSO(UNIT (xx))	Yes	Yes	Yes
	'N'	TSO(NOUNIT)	No	Yes	
USERDATA	'Y'	TSO(USERDATA (xx))	Yes	Yes	Yes
	'N'	TSO(NOUSERDATA)	No	Yes	

Table 149. WORKATTR segment fields

Field name	Flag byte values	ADDUSER/ALTUSER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
WAACCNT	'Y'	WORKATTR(WAACCNT (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWAACCNT)	No	Yes	
WAADDR1	'Y'	WORKATTR(WAADDR1 (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWADDR1)	No	Yes	
WAADDR2	'Y'	WORKATTR(WAADDR2 (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWADDR2)	No	Yes	
WAADDR3	'Y'	WORKATTR(WAADDR3 (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWADDR3)	No	Yes	
WAADDR4	'Y'	WORKATTR(WAADDR4 (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWADDR4)	No	Yes	
WABLDG	'Y'	WORKATTR(WABLDG(xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWABLDG)	No	Yes	
WADEPT	'Y'	WORKATTR(WADEPT (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWADEPT)	No	Yes	
WANAME	'Y'	WORKATTR(WANAME (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWANAME(xx))	No	Yes	
WAROOM	'Y'	WORKATTR(WAROOM (xx))	Yes	Yes	Yes
	'N'	WORKATTR(NOWAROOM)	No	Yes	

Group administration

The following tables define field names and their usage. All field names relate directly to the ADDGROUP and ALTGROUP keywords. Although the fields are alphabetized in the following tables, there is no defined order in which fields are returned when using the extract functions. See *z/OS Security Server RACF Command*

Language Reference for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean and list fields are identified in the field name column. Unless otherwise noted, a field is a character field by default. For list fields, the list header field returned by the extract function is also specified. Unless otherwise noted, all list fields are 1-dimensional.

Table 150. BASE segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference, or LISTGROUP heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CONNECTS	N/A	**Note:** This is the list header field for the 2-dimensional array consisting of the following two fields.	No	No	Yes
GAUTH	N/A	ACCESS=	No	No	Yes
GUSERID	N/A	USER(S)=	No	No	Yes
CREATDAT	N/A	CREATED=	No	No	Yes
DATA	'Y'	DATA(xx)	Yes	Yes	Yes
	'N'	NODATA	No	Yes	
MODEL	'Y'	MODEL (xx)	Yes	Yes	Yes
	'N'	NOMODEL	No	Yes	
OWNER	'Y'	OWNER(xx)	Yes	Yes	Yes
SUBGROUP (list SUBGRPCT)	N/A	SUBGROUP(S)=	No	No	Yes
SUPGROUP	'Y'	SUPGROUP(xx)	Yes	Yes	Yes
TERMUACC (boolean)	'Y'	TERMUACC	Yes	Yes	Yes
	'N'	NOTERMUACC	Yes	Yes	
UNIVERSL (boolean in output)	'Y'	UNIVERSAL	Yes	No	Yes

Table 151. CSDATA segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
custom-keyword	'Y'	CSDATA (custom-keyword(xxx))	Yes	Yes	Yes
	'N'	CSDATA (NOcustom-keyword)	No	Yes	
A custom field (keyword and value type) is installation specific. On extract requests the field descriptor mapping will contain the *custom-keyword* and the associated data value.					

Table 152. DFP segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DATAAPPL	'Y'	DFP(DATAAPPL(xx))	Yes	Yes	Yes
	'N'	DFP(NODATAAPPL)	No	Yes	Yes

Table 152. DFP segment fields (continued)

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DATACLAS	'Y'	DFP(DATACLAS(xx))	Yes	Yes	Yes
	'N'	DFP(NODATACLAS)	No	Yes	Yes
MGMTCLAS	'Y'	DFP(MGMTCLAS(xx))	Yes	Yes	Yes
	'N'	DFP(NOMGMTCLAS)	No	Yes	Yes
STORCLAS	'Y'	DFP(STORCLAS(xx))	Yes	Yes	Yes
	'N'	DFP(NOSTORCLAS)	No	Yes	Yes

Table 153. OMVS segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
AUTOGID	'Y'	OMVS(AUTOGID)	Yes	Yes	No
GID	'Y'	OMVS(GID(xx))	Yes	Yes	Yes
	'N'	OMVS(NOGID)	No	Yes	
SHARED	'Y'	OMVS(SHARED)	Yes	Yes	No

Table 154. OVM segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
GID	'Y'	OVM(GID(xx))	Yes	Yes	Yes
	'N'	OVM(NOGID)	No	Yes	

Table 155. TME segment fields

Field name	Flag byte values	ADDGROUP/ALTGROUP keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ROLES	'Y'	TME(ROLES(xx ...))	Yes	Yes	Yes
	'A'	TME(ADDROLES(xx ...))	No	Yes	
	'D'	TME(DELROLES (xx ...))	No	Yes	
	'N'	TME(NOROLES)	No	Yes	

Group connection administration

The following tables define field names and their usage. All field names relate directly to the CONNECT and REMOVE keywords. Although the fields are alphabetized in the following tables, there is no defined order in which the fields are returned when using the extract function. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean fields are identified in the field name column. Unless otherwise noted, a field is a character field by default.

Table 156. Base segment fields

Field name	Flag byte values	CONNECT and REMOVE keyword reference , or LISTUSER heading (for output-only fields)	Allowed on CONNECT requests	Allowed on REMOVE requests	Returned on extract requests
ADSP (boolean)	'Y'	ADSP	Yes	No	Yes
	'N'	NOADSP	Yes	No	
AUDITOR (boolean)	'Y'	AUDITOR	Yes	No	Yes
	'N'	NOAUDITOR	Yes	No	
AUTH	'Y'	AUTHORITY(xx)	Yes	No	Yes
CGAUTHDA	N/A	CONNECT-DATE=	No	No	Yes
CGINITCT	N/A	CONNECTS=	No	No	Yes
CGLJDATE	N/A	LAST-CONNECT= (left side of "/")	No	No	Yes
CGLJTIME	N/A	LAST-CONNECT= (right side of "/")	No	No	Yes
GROUP	'Y'	GROUP(xx)	Yes	Yes	No
GRPACC (boolean)	'Y'	GRPACC	Yes	No	Yes
	'N'	NOGRPACC	Yes	No	
OPER (boolean)	'Y'	OPERATIONS	Yes	No	Yes
	'N'	NOOPERATIONS	Yes	No	
OWNER	'Y'	OWNER (xx)	Yes	Yes	Yes
RESUME (on input, this can be used as a boolean or a date. On output, this is the resume date.)	'Y'	RESUME (xx)	Yes	No	Yes
	'N'	NORESUME	Yes	No	
REVOKE (on input, this can be used as a boolean or a date. On output, this is the revoke date. See REVOKEFL field for boolean output.)	'Y'	REVOKE (xx)	Yes	No	Yes
	'N'	NOREVOKE	Yes	No	
REVOKEFL (boolean)	N/A	N/A	No	No	Yes
NOTE: The value of the REVOKEFL field is consistent with the behavior of the LISTUSER command in that it takes revoke and resume dates into account when determining if the user is revoked.					
SPECIAL (boolean)	'Y'	SPECIAL	Yes	No	Yes
	'N'	NOSPECIAL	Yes	No	
UACC	'Y'	UACC (xx)	Yes	No	Yes

General resource administration

The following tables define field names and their usage. All field names relate directly to the RDEFINE, RALTER, and RLIST keywords. Although the fields are alphabetized in the following tables, there is no defined order in which the fields are returned when using the extract functions. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean fields are identified in the field name column. Unless otherwise noted, a field is a character field by default. For list fields, the list header field returned by the extract function is also specified. Unless otherwise noted, all list fields are 1–dimensional arrays.

Table 157. BASE segment fields

Field name	Flag byte value	RDEFINE/RALTER/RLIST/RDELETE keyword reference, or RLIST heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Returned on extract requests	Allowed on delete requests
ACLCNT	N/A	This is the list header field for the standard access list, which is a 3-dimensional array consisting of the ACLID, ACLACS, and ACLACNT fields. For generic profiles, the ACLACNT field does not apply, and the array will be 2-dimensional.	No	No	No	Yes	No
ACLACNT	N/A	ACCESS COUNT	No	No	No	Yes	No
ACLACS	N/A	ACCESS	No	No	No	Yes	No
ACLID	N/A	ID	No	No	No	Yes	No
ACL2CNT	N/A	This is the list header field for the conditional access list, which is a 5-dimensional array consisting of the ACL2ID, ACL2ACS, ACL2ACNT, ACL2COND, and ACL2ENT fields. For generic profiles, the ACL2ACNT field does not apply, and the array will be 4-dimensional.	No	No	No	Yes	No
ACL2ACNT	N/A	ACCESS COUNT	No	No	No	Yes	No
ACL2ACS	N/A	ACCESS	No	No	No	Yes	No
ACL2COND	N/A	CLASS	No	No	No	Yes	No
ACL2ENT	N/A	ENTITY NAME	No	No	No	Yes	No
ACL2ID	N/A	ID	No	No	No	Yes	No
ACSALTR	N/A	ALTER COUNT	No	No	No	Yes	No
ACSCNTL	N/A	CONTROL COUNT	No	No	No	Yes	No
ACSREAD	N/A	READ COUNT	No	No	No	Yes	No
ACSUPDT	N/A	UPDATE COUNT	No	No	No	Yes	No
ALL (boolean)	'Y'	ALL	No	No	Yes	No	No
APPLDATA	'Y'	APPLDATA(xx)	Yes	Yes	No	Yes	No
	'N'	NOAPPLDATA	No	Yes	No		No
AUDALTR	'Y'	AUDIT(xx (ALTER))	Yes	Yes	No	No	No
AUDCNTL	'Y'	AUDIT(xx (CONTROL))	Yes	Yes	No	No	No
AUDNONE	'Y'	AUDIT(NONE)	Yes	Yes	No	No	No
AUDREAD	'Y'	AUDIT(xx (READ))	Yes	Yes	No	No	No
AUDUPDT	'Y'	AUDIT(xx (UPDATE))	Yes	Yes	No	No	No

Table 157. BASE segment fields (continued)

Field name	Flag byte value	RDEFINE/RALTER/RLIST/RDELETE keyword reference, or RLIST heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Returned on extract requests	Allowed on delete requests
AUTHUSER (boolean)	'Y'	AUTHUSER	No	No	Yes	No	No
AUTOMATC (boolean)	N/A	AUTOMATIC	No	No	No	Yes	No
CATEGORY (list NUMCTGY)	'Y'	ADDCATEGORY(xx ...)	Yes	No	No	Yes	No
	'A'	ADDCATEGORY(xx ...)	No	Yes	No		No
	'D'	DELCATEGORY(xx ...)	No	Yes	No		No
NOTE: To remove unknown categories from the profile, specify the 'D' flag and a field length of zero.							
CREATDAT	N/A	CREATION DATE	No	No	No	Yes	No
DATA	'Y'	DATA(xx)	Yes	Yes	No	Yes	No
	'N'	NODATA	No	Yes	No		No
FCLASS	'Y'	FCLASS(xx)	Yes	No	No	No	No
FGENERIC (boolean)	'Y'	FGENERIC	Yes	No	No	No	No
FPROFILE	'Y'	FROM(xx)	Yes	No	No	No	No
FVOLUME	'Y'	FVOLUME(xx)	Yes	No	No	No	No
GAUDALTR	'Y'	GLOBALAUDIT(xx (ALTER))	No	Yes	No	No	No
GAUDCNTL	'Y'	GLOBALAUDIT(xx (CONTROL))	No	Yes	No	No	No
GAUDNONE	'Y'	GLOBALAUDIT (NONE)	No	Yes	No	No	No
GAUDREAD	'Y'	GLOBALAUDIT(xx (READ))	No	Yes	No	No	No
GAUDUPDT	'Y'	GLOBALAUDIT(xx (UPDATE))	No	Yes	No	No	No
GENERIC	'Y'	GENERIC	No	No	Yes	No	No
	'N'	NOGENERIC	No	No	Yes		Yes
HISTORY (boolean)	'Y'	HISTORY	No	No	Yes	No	No
LCHGDAT	N/A	LAST CHANGE DATE	No	No	No	Yes	No
LEVEL	'Y'	LEVEL(xx)	Yes	Yes	No	Yes	No
LREFDAT	N/A	LAST REFERENCE DATE	No	No	No	Yes	No
MEMBER (list MEMCNT)	'Y'	ADDMEM(xx ...)	Yes	No	No	Yes	No
	'A'	ADDMEM(xx ...)	No	Yes	No		No
	'D'	DELMEM(xx ...)	No	Yes	No		No
NORACF (boolean)	'Y'	NORACF	No	No	Yes	No	No
NOTIFY	'Y'	NOTIFY(xx)	Yes	Yes	No	Yes	No
	'N'	NONOTIFY	No	Yes	No		No
NOYOURAC (boolean)	'Y'	NOYOURACC	No	No	Yes	No	No
OWNER	'Y'	OWNER(xx)	Yes	Yes	No	Yes	No
PROFILE	'Y'	N/A (see note)	Yes	Yes	Yes	No	No
RAUDIT	N/A	AUDITING	No	No	No	Yes	No
RESGROUP	'Y'	RESGROUP	No	No	Yes	No	No
RGAUDIT	N/A	GLOBALAUDIT	No	No	No	Yes	No
SECLABEL	'Y'	SECLABEL (xx)	Yes	Yes	No	Yes	No
	'N'	NOSECLABEL	No	Yes	No		No

Table 157. BASE segment fields (continued)

Field name	Flag byte value	RDEFINE/RALTER/RLIST/RDELETE keyword reference, or RLIST heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Returned on extract requests	Allowed on delete requests
SECLEVEL	'Y'	SECLEVEL(xx)	Yes	Yes	No	Yes	No
	'N'	NOSECLEVEL	No	Yes	No		No
SINGLDSN (boolean)	'Y'	SINGLEDSN	Yes	Yes	No	Yes	No
	'N'	NOSINGLEDSN	No	Yes	No		No
STATS (boolean)	'Y'	STATISTICS	No	No	Yes	No	No
TIMEZONE	'Y'	TIMEZONE(xx)	Yes	Yes	No	Yes	No
	'N'	NOTIMEZONE	No	Yes	No		No
TVTOC (boolean)	'Y'	TVTOC	Yes	Yes	Yes	Yes	No
	'N'	NOTVTOC	No	Yes	No		No
UACC	'Y'	UACC(xx)	Yes	Yes	No	Yes	No
VOLUME (list VOLCNT)	'A'	ADDVOL(xx ...)	No	Yes	No	Yes	No
	'D'	DELVOL(xx ...)	No	Yes	No		No
WARNING (boolean)	'Y'	WARNING	Yes	Yes	No	Yes	No
	'N'	NOWARNING	No	Yes	No		No
WHENDAYS (list WHENDYCT)	'Y'	WHEN(DAYS(xx))	Yes	Yes	No	Yes	No
WHENTIME	'Y'	WHEN(TIME (xx))	Yes	Yes	No	Yes	No

Table 158. CDTINFO segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CDTCASE	'Y'	CDTINFO(CASE(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOCASE)	No	Yes	
CDTDFTRC	'Y'	CDTINFO(DEFAULTRC(x))	Yes	Yes	Yes
	'N'	CDTINFO(NODEFAULTRC)	No	Yes	
CDTFIRST (list CDTFIRN)	'Y'	CDTINFO(FIRST(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOFIRST)	No	Yes	
CDTGEN	'Y'	CDTINFO (GENERIC(xx))	Yes	Yes	Yes
	'N'	CDTINFO (NOGENERIC)	No	Yes	
CDTGENL	'Y'	CDTINFO(GENLIST(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOGENLIST)	No	Yes	
CDTGROUP	'Y'	CDTINFO(GROUP(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOGROUP)	No	Yes	
CDTKEYQL	'Y'	CDTINFO(KEYQUALIFIERS(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOKEYQUALIFIERS)	No	Yes	
CDTMAC	'Y'	CDTINFO(MACPROCESSING(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOMACPROCESSING)	No	Yes	
CDTMAXLN	'Y'	CDTINFO(MAXLENGTH(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOMAXLENGTH)	No	Yes	

Table 158. CDTINFO segment fields (continued)

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CDTMAXLX	'Y'	CDTINFO(MAXLENX(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOMAXLENX)	No	Yes	
CDTMEMBR	'Y'	CDTINFO(MEMBER(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOMEMBER)	No	Yes	
CDTOPER	'Y'	CDTINFO(OPERATIONS(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOOPERATIONS)	No	Yes	
CDTOTHER (list CDTOTHN)	'Y'	CDTINFO(OTHER(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOOTHER))	No	Yes	
CDTPOSIT	'Y'	CDTINFO(POSIT(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOPOSIT)	No	Yes	
CDTPRFAL	'Y'	CDTINFO(PROFILESALLOWED(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOPROFILESALLOWED)	No	Yes	
CDTRACL	'Y'	CDTINFO(RACLIST(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NORACLIST)	No	Yes	
CDTSIGL	'Y'	CDTINFO(SIGNAL(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOSIGNAL)	No	Yes	
CDTSLREQ	'Y'	CDTINFO(SECLABELSREQUIRED(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NOSECLABELSREQUIRED)	No	Yes	
CDTUACC	'Y'	CDTINFO(DEFAULTUACC(xx))	Yes	Yes	Yes
	'N'	CDTINFO(NODEFAULTUACC)	No	Yes	

Table 159. CFDEF segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CFDTYPE	'Y'	CFDEF (TYPE(xx))	Yes	No	Yes
CFFIRST	'Y'	CFDEF (FIRST(xx))	Yes	Yes	Yes
CFHELP	'Y'	CFDEF (HELP(xx))	Yes	Yes	Yes
CFLIST	'Y'	CFDEF (LISTHEAD(xx))	Yes	Yes	Yes
CFMIXED	'Y'	CFDEF (MIXED(xx))	Yes	Yes	Yes
CFMNVAL	'Y'	CFDEF (MINVALUE(xx))	Yes	Yes	Yes
	'N'	CFDEF (NOMINVALUE)	No	Yes	
CFMXLEN	'Y'	CFDEF (MAXLENGTH(xx))	Yes	Yes	Yes
CFMXVAL	'Y'	CFDEF (MAXVALUE(xx))	Yes	Yes	Yes
	'N'	CFDEF (NOMAXVALUE)	No	Yes	
CFOTHER	'Y'	CFDEF (OTHER(xx))	Yes	Yes	Yes

Table 160. DLFDATA segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
JOBNAME (list JOBNMCNT)	'Y'	DLFDATA (JOBNAMES (xx ...))	Yes	Yes	Yes
	'A'	DLFDATA(ADDJOBNAMES(xx ...))	No	Yes	
	'D'	DLFDATA(DELJOBNAMES(xx ...))	No	Yes	
	'N'	DLFDATA(NOJOBNAMES)	No	Yes	
RETAIN (boolean)	'Y'	DLFDATA(RETAIN(YES))	Yes	Yes	Yes
	'N'	DLFDATA (RETAIN(NO))	Yes	Yes	

Table 161. EIM segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DOMAINDN	'Y'	EIM(DOMAINDN(xx))	Yes	Yes	Yes
	'N'	EIM(NODOMAINDN)	No	Yes	
KERBREG	'Y'	EIM(KERBREGISTRY(xx))	Yes	Yes	Yes
	'N'	EIM(NOKERBREGISTRY)	No	Yes	
LOCALREG	'Y'	EIM(LOCALREGISTRY(xx))	Yes	Yes	Yes
	'N'	EIM(NOLOCALREGISTRY)	No	Yes	
OPTIONS	'Y'	EIM(OPTIONS(xx))	Yes	Yes	Yes
	'N'	EIM(NOOPTIONS)	No	Yes	
X509REG	'Y'	EIM(X509REGISTRY(xx))	Yes	Yes	Yes
	'N'	EIM(NOX509REGISTRY)	No	Yes	

Table 162. KERB segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference, or RLIST heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CHKADDRS	'Y'	KERB(CHECKADDRS(Y))	Yes	Yes	Yes
	'N'	KERB(CHECKADDRS(N))	Yes	Yes	
DEFTKTLF	'Y'	KERB(DEFTKTLFE(xx))	Yes	Yes	Yes
	'N'	KERB(NODEFTKTLFE)	No	Yes	
ENCRYPT (list ENCRYPTN)	'Y'	KERB(ENCRYPT(xx))	Yes	Yes	Yes
	'N'	KERB(NOENCRYPT)	No	Yes	
KERBNAME	'Y'	KERB(KERBNAME(xx))	Yes	Yes	Yes
	'N'	KERB(NOKERBNAME)	No	Yes	
KEYVERS	N/A	KEY VERSION=	No	No	Yes
MAXTKTLF	'Y'	KERB(MAXTKTLFE(xx))	Yes	Yes	Yes
	'N'	KERB(NOMAXTKTLFE)	No	Yes	
MINTKTLF	'Y'	KERB(MINTKTLFE(xx))	Yes	Yes	Yes
	'N'	KERB(NOMINTKTLFE)	No	Yes	

Table 162. KERB segment fields (continued)

Field name	Flag byte values	RDEFINE/RALTER keyword reference, or RLIST heading (for output-only fields)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
PASSWORD	'Y'	KERB(PASSWORD(xx))	Yes	Yes	No
	'N'	KERB(NOPASSWORD)	No	Yes	

Table 163. ICSF segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CRTLBLS (list CRTLBLCT)	'Y'	ICSF(SYMEXPORTCERTS(xx ...))	Yes	Yes	Yes
	'A'	ICSF(ADDSYMEXPORTCERTS (xx ...))	No	Yes	
	'D'	ICSF(DELSYMEXPORTCERTS (xx ...))	No	Yes	
	'N'	ICSF(NOSYMEXPORTCERTS)	No	Yes	
EXPORT	'Y'	ICSF(SYMEXPORTABLE(xx))	Yes	Yes	Yes
	'N'	ICSF(NOSYMEXPORTABLE)	No	Yes	
KEYLBLS (list KEYLBLCT)	'Y'	ICSF(SYMEXPORTKEYS(xx ...))	Yes	Yes	Yes
	'A'	ICSF(ADDSYMEXPORTKEYS (xx ...))	No	Yes	
	'D'	ICSF(DELSYMEXPORTKEYS (xx ...))	No	Yes	
	'N'	ICSF(NOSYMEXPORTKEYS)	No	Yes	
SCPWRAP	'Y'	ICSF(SYMCPACFWRAP(xx))	Yes	Yes	Yes
USAGE (list USAGECT)	'Y'	ICSF(ASYMUSAGE(xx ...))	Yes	Yes	Yes
	'N'	ICSF(NOASYMUSAGE)	No	Yes	

Table 164. ICTX segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
DOMAP (Boolean)	'Y'	ICTX(DOMAP(YES))	Yes	Yes	Yes
	'N'	ICTX(DOMAP(NO))	Yes	Yes	
MAPREQ (Boolean)	'Y'	ICTX(MAPREQUIRED(YES))	Yes	Yes	Yes
	'N'	ICTX(MAPREQUIRED(NO))	Yes	Yes	
MAPTIMEO	'Y'	ICTX(MAPPINGTIMEOUT(xx))	Yes	Yes	Yes
	'N'	ICTX(NOMAPPINGTIMEOUT)	No	Yes	
USEMAP (Boolean)	'Y'	ICTX(USEMAP(YES))	Yes	Yes	Yes
	'N'	ICTX(USEMAP(NO))	Yes	Yes	

Table 165. PROXY segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
BINDDN	'Y'	PROXY(BINDDN(xx))	Yes	Yes	Yes
	'N'	PROXY(NOBINDDN)	No	Yes	

Table 165. PROXY segment fields (continued)

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
BINDPW	'Y'	PROXY(BINDPW(xx))	Yes	Yes	No
	'N'	PROXY(NOBINDPW)	No	Yes	
LDAPHOST	'Y'	PROXY(LDAPHOST(xx))	Yes	Yes	Yes
	'N'	PROXY(NOLDAPHOST)	No	Yes	

Table 166. SESSION segment fields

Field name	Flag byte value	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CONVSEC	'Y'	SESSION (CONVSEC(xx))	Yes	Yes	Yes
	'N'	SESSION (NOCONVSEC))	No	Yes	
INTERVAL	'Y'	SESSION(INTERVAL(xx))	Yes	Yes	Yes
	'N'	SESSION (NOINTERVAL))	No	Yes	
LOCK (boolean)	'Y'	SESSION (LOCK)	Yes	Yes	Yes
	'N'	SESSION (NOLOCK)	No	Yes	
SESSKEY	'Y'	SESSION (SESSKEY(xx))	Yes	Yes	Yes
	'N'	SESSION(NOSESSKEY))	No	Yes	

Table 167. SIGVER segment fields

Field name	Flag byte value	RDEFINE/RALTER keyword reference (R_admin keyword table has a 12 character limit)	Allowed on add requests	Allowed on alter requests	Returned on extract requests
FAILLOAD	'Y'	SIGVER(FAILLOAD(xx))	Yes	Yes	Yes
	'N'	SIGVER(NOFAILLOAD)	No	Yes	
SIGAUDIT	'Y'	SIGVER(SIGAUDIT(xx))	Yes	Yes	Yes
	'N'	SIGVER(NOSIGAUDIT)	No	Yes	
SIGREQD (boolean)	'Y'	SIGVER(SIGREQUIRED(YES))	Yes	Yes	Yes
	'N'	SIGVER(SIGREQUIRED(NO))	Yes	Yes	

Table 168. SSIGNON segment fields

Field name	Flag byte values	REDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
KEYCRYPT	'Y'	SSIGNON (KEYENCRYPT (xx))	Yes	Yes	No
	'N'	SSIGNON(NOKEYENCRYPT)	No	Yes	
KEYMASK	'Y'	SSIGNON (KEYMASK(xx))	Yes	Yes	No
	'N'	SSIGNON (NOKEYMASK)	No	Yes	

Table 169. STDATA segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
GROUP	'Y'	STDATA(GROUP(xx))	Yes	Yes	Yes
	'N'	STDATA(NOGROUP)	No	Yes	
PRIVLEGE (boolean)	'Y'	STDATA(PRIVILEGED(YES))	Yes	Yes	Yes
	'N'	STDATA(PRIVILEGED(NO))	Yes	Yes	
TRACE (boolean)	'Y'	STDATA(TRACE(YES))	Yes	Yes	Yes
	'N'	STDATA(TRACE(NO))	Yes	Yes	
TRUSTED (boolean)	'Y'	STDATA(TRUSTED(YES))	Yes	Yes	Yes
	'N'	STDATA(TRUSTED(NO))	Yes	Yes	
USER	'Y'	STDATA(USER(xx))	Yes	Yes	Yes
	'N'	STDATA(NOUSER)	No	Yes	

Table 170. SVFMR segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
PARMNAME	'Y'	SVFMR(PARMNAME(xx))	Yes	Yes	Yes
	'N'	SVFMR(NOPARMNAME)	No	Yes	
SCRIPT	'Y'	SVFMR(SCRIPTNAME(xx))	Yes	Yes	Yes
	'N'	SVFMR(NOSCRIPTNAME)	No	Yes	

Table 171. TME segment fields

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
CHILDREN (list CHILDN)	'Y'	TME(CHILDREN(xx ...))	Yes	Yes	Yes
	'A'	TME(ADDCHILDREN(xx ...))	No	Yes	
	'D'	TME(DELCHILDREN(xx ...))	No	Yes	
	'N'	TME(NOCHILDREN)	No	Yes	
GROUPS (list GROUPN)	'Y'	TME(GROUPS(xx ...))	Yes	Yes	Yes
	'A'	TME(ADDGROUPS(xx ...))	No	Yes	
	'D'	TME(DELGROUPS(xx ...))	No	Yes	
	'N'	TME(NOGROUPS)	No	Yes	
PARENT	'Y'	TME(PARENT(xx))	Yes	Yes	Yes
	'N'	TME(NOPARENT)	No	Yes	
RESOURCE (list RESN)	'Y'	TME(RESOURCE(xx ...))	Yes	Yes	Yes
	'A'	TME(ADDRESOURCE(xx ...))	No	Yes	
	'D'	TME(DELRESOURCE(xx ...))	No	Yes	
	'N'	TME(NORESOURCE)	No	Yes	

Table 171. TME segment fields (continued)

Field name	Flag byte values	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
ROLES (list ROLEN)	'Y'	TME(ROLES(xx ...))	Yes	Yes	Yes
	'A'	TME(ADDROLES(xx ...))	No	Yes	
	'D'	TME(DELROLES(xx ...))	No	Yes	
	'N'	TME(NOROLES)	No	Yes	

Data set administration

The following tables define the DATASET field names and their usage. All field names relate directly to the ADDSD, ALTDSD, DELDSD, and LISTDSD keywords. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean fields are identified in the field name column. Unless otherwise noted, a field is a character field by default.

Table 172. BASE segment fields

Field name	Flag byte value	ADDSD, ALTDSD, DELDSD, LISTDSD keyword reference	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Allowed on delete requests
ALL (boolean)	'Y'	ALL	No	No	Yes	No
ALTVOL	'Y'	ALTVOL (xx)	No	Yes	No	No
AUDALTR	'Y'	AUDIT(xx(ALTER))	Yes	Yes	No	No
AUDCNTL	'Y'	AUDIT (xx (CONTROL))	Yes	Yes	No	No
AUDNONE (boolean)	'Y'	AUDIT (NONE)	Yes	Yes	No	No
AUDREAD	'Y'	AUDIT (xx (READ))	Yes	Yes	No	No
AUDUPDT	'Y'	AUDIT (xx (UPDATE))	Yes	Yes	No	No
AUTHUSER (boolean)	'Y'	AUTHUSER	No	No	Yes	No
CATEGORY	'Y'	ADDCATEGORY (xx ...)	Yes	No	No	No
	'A'	ADDCATEGORY (xx ...)	No	Yes	No	No
	'D'	DELCATEGORY (xx ...)	No	Yes	No	No
NOTE: To remove unknown categories from the profile, specify the 'D' flag and a field length of zero.						
DATA	'Y'	DATA (xx)	Yes	Yes	No	No
	'N'	NODATA	No	Yes	No	No
DSNS (boolean)	'Y'	DSNS	No	No	Yes	No
ERASE (boolean)	'Y'	ERASE	Yes	Yes	No	No
	'N'	NOERASE	No	Yes	No	No
FCLASS	'Y'	FCLASS (xx)	Yes	No	No	No
FGENERIC (boolean)	'Y'	FGENERIC	Yes	No	No	No

Table 172. BASE segment fields (continued)

Field name	Flag byte value	ADDSD, ALTDSD, DELDSD, LISTDSD keyword reference	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Allowed on delete requests
FILESEQ	'Y'	FILESEQ (xx)	Yes	No	No	No
FROM	'Y'	FROM (xx)	Yes	No	No	No
FVOLUME	'Y'	FVOLUME (xx)	Yes	No	No	No
GAUDALTR	'Y'	GLOBALAUDIT (xx (ALTER))	No	Yes	No	No
GAUDCNTR	'Y'	GLOBALAUDIT (xx (CONTROL))	No	Yes	No	No
GAUDNONE (boolean)	'Y'	GLOBALAUDIT (NONE)	No	Yes	No	No
GAUDREAD	'Y'	GLOBALAUDIT (xx (READ))	No	Yes	No	No
GAUDUPDT	'Y'	GLOBALAUDIT (xx UPDATE))	No	Yes	No	No
GENERIC (boolean)	'Y'	GENERIC	Yes	Yes	Yes	Yes
	'N'	NOGENERIC	No	No	Yes	No
HISTORY	'Y'	HISTORY	No	No	Yes	No
LEVEL	'Y'	LEVEL (xx)	Yes	Yes	No	No
MODEL (boolean)	'Y'	MODEL	Yes	No	No	No
NORACF (boolean)	'Y'	NORACF	No	No	Yes	No
NOTIFY	'Y'	NOTIFY (xx)	Yes	Yes	No	No
	'N'	NONOTIFY	No	Yes	No	No
OWNER	'Y'	OWNER (xx)	Yes	Yes	No	No
PREFIX	'Y'	PREFIX (xx)	No	No	Yes	No
PROFILE	'Y'	for list, DATASET (xx...)	Yes	Yes	Yes	Yes
NOTE: For add, alter, and delete, this field is required. There is no associated command keyword because it is a positional parameter. For list, this field is optional and is used in the DATASET(xx...) keyword. For add and alter, if working with a password-protected data set, append "/password? to the data set profile name, and include this additional data in the length field for the data set profile name.						
RETPD	'Y'	RETPD (xx)	Yes	Yes	No	No
SECLABEL	'Y'	SECLABEL (xx)	Yes	Yes	No	No
	'N'	NOSECLABEL	No	Yes	No	No
SECLEVEL	'Y'	SECLEVEL (xx)	Yes	Yes	No	No
	'N'	NOSECLEVEL	No	Yes	No	No
SET (boolean)	'Y'	SET	Yes	Yes	No	Yes
SETONLY (boolean)	'Y'	SETONLY	Yes	No	No	No
STATS (boolean)	'Y'	STATISTICS	No	No	Yes	No
TAPE (boolean)	'Y'	TAPE	Yes	No	No	No
UACC	'Y'	UACC	Yes	Yes	No	No
UNIT	'Y'	UNIT (xx)	Yes	Yes	No	No

Table 172. BASE segment fields (continued)

Field name	Flag byte value	ADDSD, ALTDSD, DELDSD, LISTDSD keyword reference	Allowed on add requests	Allowed on alter requests	Allowed on list requests	Allowed on delete requests
VOLUME	'Y'	VOLUME (xx ...)	Yes	Yes	Yes	Yes
	'A'	ADDVOL (xx ...)	No	Yes	No	No
	'D'	DELVOL (xx ...)	No	Yes	No	No
WARNING (boolean)	'Y'	WARNING	Yes	Yes	No	No
	'N'	NOWARNING	No	Yes	No	No

Table 173. DFP segment fields

Field name	Flag byte value	ADDSD and ALTDSD keyword reference	Allowed on add requests	Allowed on alter requests
RESOWNER	'Y'	DFP(RESOWNER(xx))	Yes	Yes
	'N'	DFP(NORESOWNER)	No	Yes

Table 174. TME segment fields

Field name	Flag byte value	ADDSD and ALTDSD keyword reference	Allowed on add requests	Allowed on alter requests
ROLES	'Y'	TME(ROLES(xx...))	Yes	Yes
	'A'	TME(ADDROLES(xx...))	No	Yes
	'D'	TME(DELROLES(xx...))	No	Yes
	'N'	TME(NOROLES)	No	Yes

Access list administration

The following tables define the permit field names and their usage. All field names relate directly to PERMIT keywords. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean fields are identified in the field name column. Unless otherwise noted, a field is a character field by default.

Table 175. Base segment fields

Field name	Flag byte values	PERMIT keyword reference
ACCESS	'Y'	ACCESS (xx)
DELETE (boolean)	'Y'	DELETE
FCLASS	'Y'	FCLASS (xx)
FPROFILE	'Y'	FROM (xx)
FGENERIC (boolean)	'Y'	FGENERIC
FVOLUME	'Y'	FVOLUME (xx)
GENERIC (boolean)	'Y'	GENERIC
ID	'Y'	ID (xx)

Table 175. Base segment fields (continued)

Field name	Flag byte values	PERMIT keyword reference
PROFILE	'Y'	N/A (See note.)
NOTE: This field is required; there is no associated command keyword because it is a positional parameter.		
RESET	'Y'	RESET (xx)
VOLUME	'Y'	VOLUME (xx)
WHENAPPC	'Y'	WHEN (APPCPORT (...))
WHENCONS	'Y'	WHEN (CONSOLE (...))
WHENJES	'Y'	WHEN (JESINPUT (...))
WHENPROG	'Y'	WHEN (PROGRAM (...))
WHENSERV	'Y'	WHEN(SERVAUTH(...))
WHENSQLR	'Y'	WHEN(CRIT(SQLROLE(...)))
WHENSYS	'Y'	WHEN (SYSID (...))
WHENTERM	'Y'	WHEN (TERMINAL (...))

SETROPTS administration

The following tables define SETROPTS field names and their usage. All field names relate directly to SETROPTS keywords. Although the fields are alphabetized in the following table, there is no defined order in which the fields are returned when using the extract function. See *z/OS Security Server RACF Command Language Reference* for questions pertaining to field usage and data. Note that within the command image generated internally, RACF truncates long keywords to 12 characters.

Boolean fields are identified in the field name column. Unless otherwise noted, a field is a character field by default.

When SETROPTS extract returns the fields which contain a list of classes (CLASSACT, CLASSTAT, GENCMD, GENERIC, GENLIST, GLOBAL, RACLIST, AUDIT, LOGALWYS, LOGNEVER, LOGSUCC, LOGFAIL, AND LOGDEFLT), each class name (including the final one) will be padded with blanks to eight characters, and followed by a single blank. Therefore, you can determine the number of classes by dividing the total field length by nine.

Table 176. BASE segment field names

Field name	Flag byte value	SETROPTS keyword reference
ADDCREAT (boolean)	'Y'	ADDCREATOR
	'N'	NOADDCREATOR
ADSP (boolean)	'Y'	ADSP
	'N'	NOADSP
APPLAUDT (boolean)	'Y'	APPLAUDIT
	'N'	NOAPPLAUDIT
AUDIT	'A'	AUDIT (xx ...)
	'D'	NOAUDIT (xx ...)

Table 176. BASE segment field names (continued)

Field name	Flag byte value	SETROPTS keyword reference
CATDSNS	'Y'	CATDSNS (xx)
	'N'	NOCATDSNS
CLASSACT	'A'	CLASSACT (xx ...)
	'D'	NOCLASSACT (xx ...)
CLASSTAT	'A'	STATISTICS (xx ...)
	'D'	NOSTATISTICS (xx ...)
CMDVIOL (boolean)	'Y'	CMDVIOL
	'N'	NOCMDVIOL
COMPMODE (boolean)	'Y'	COMPATMODE
	'N'	NOCOMPATMODE
EGN (boolean)	'Y'	EGN
	'N'	NOEGN
ERASE (boolean)	'Y'	ERASE
	'N'	NOERASE
ERASEALL (boolean)	'Y'	ERASE (ALL)
ERASESEC	'Y'	ERASE (SECLEVEL (xx))
	'N'	ERASE (NOSECLEVEL)
GENCMD	'A'	GENCMD (xx ...)
	'D'	NOGENCMD (xx ...)
GENERIC	'A'	GENERIC (xx ...)
	'D'	NOGENERIC (xx ...)
GENLIST	'A'	GENLIST (xx ...)
	'D'	NOGENLIST (xx ...)
GENOWNER (boolean)	'Y'	GENERICOWNER
	'N'	NOGENERICOWNER
GLOBAL	'A'	GLOBAL (xx ...)
	'D'	NOGLOBAL (xx ...)
GRPLIST (boolean)	'Y'	GRPLIST
	'N'	NOGRPLIST
HISTORY	'Y'	PASSWORD (HISTORY (xx))
	'N'	PASSWORD (NOHISTORY)
INACTIVE	'Y'	INACTIVE (xx)
	'N'	NOINACTIVE (xx)
INITSTAT (boolean)	'Y'	INITSTATS
	'N'	NOINITSTATS
INTERVAL	'Y'	PASSWORD (INTERVAL (xx))
JESBATCH (boolean)	'Y'	JES (BATCHALLRACF)
	'N'	JES (NOBATCHALLRACF)

Table 176. BASE segment field names (continued)

Field name	Flag byte value	SETROPTS keyword reference
JESEARLY (boolean)	'Y'	JES (EARLYVERIFY)
	'N'	JES (NOEARLYVERIFY)
JESNJE	'Y'	JES (NJEUSERID(xx))
JESUNDEF	'Y'	JES (UNDEFINEDUSER(xx))
JESXBM (boolean)	'Y'	JES (XBMALLRACF)
	'N'	JES (NOXBMALLRACF)
KERBLVL	'Y'	KERBLVL(xx)
LIST (boolean)	'Y'	LIST
NOTE: The LIST field is not returned by ADMN_UNL_SETR or ADMN_XTR_SETR.		
LOGALWYS	'Y'	LOGOPTIONS (ALWAYS (xx ...))
LOGDEFLT	'Y'	LOGOPTIONS (DEFAULT (xx ...))
LOGFAIL	'Y'	LOGOPTIONS (FAILURES (xx ...))
LOGNEVER	'Y'	LOGOPTIONS (NEVER (xx ...))
LOGSUCC	'Y'	LOGOPTIONS (SUCCESSES (xx ...))
MINCHANG	'Y'	PASSWORD (MINCHANG(xx))
MIXDCASE (boolean)	'Y'	PASSWORD (MIXEDCASE)
	'N'	PASSWORD (NOMIXEDCASE)
MLACTIVE	'Y'	MLACTIVE (xx)
	'N'	NOMLACTIVE
MLFS	'Y'	MLFSOBJ(xx)
MLIPC	'Y'	MLIPCOBJ(xx)
MLNAMES (boolean)	'Y'	MLNAMES
	'N'	NOMLNAMES
MLQUIET (boolean)	'Y'	MLQUIET
	'N'	NOMLQUIET
MLS	'Y'	MLS (xx)
	'N'	NOMLS
MLSTABLE (boolean)	'Y'	MLSTABLE
	'N'	NOMLSTABLE
MODEL (boolean)	'N'	NOMODEL
MODGDG (boolean)	'Y'	MODEL (GDG)
	'N'	MODEL (NOGDG)
MODGROUP (boolean)	'Y'	MODEL (GROUP)
	'N'	MODEL (NOGROUP)
MODUSER (boolean)	'Y'	MODEL (USER)
	'N'	MODEL (NOUSER)
OPERAUDT (boolean)	'Y'	OPERAUDT
	'N'	NOOPERAUDT

Table 176. BASE segment field names (continued)

Field name	Flag byte value	SETROPTS keyword reference
PREFIX	'Y'	PREFIX (xx)
	'N'	NOPREFIX
PRIMLANG	'Y'	LANGUAGE (PRIMARY (xx))
PROTALL	'Y'	PROTECTALL (xx)
	'N'	NOPROTECTALL
RACLIST	'A'	RACLIST (xx ...)
	'D'	NORACLIST (xx ...)
REALDSN (boolean)	'Y'	REALDSN
	'N'	NOREALDSN
REFRESH (boolean)	'Y'	REFRESH
NOTE: The REFRESH field is not returned by ADMN_UNL_SETR or ADMN_XTR_SETR.		
RETPD	'Y'	RETPD (xx)
REVOKE	'Y'	PASSWORD (REVOKE (xx))
	'N'	PASSWORD (NOREVOKE)
RULES (boolean)	'N'	PASSWORD (NORULES)
NOTE: Specifying RULES with the 'N' flag results in the cancellation of all password syntax rules, regardless of any RULEn fields also specified.		
RULE1	'Y'	PASSWORD (RULE1 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE1)
RULE2	'Y'	PASSWORD (RULE2 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE2)
RULE3	'Y'	PASSWORD (RULE3 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE3)
RULE4	'Y'	PASSWORD (RULE4 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE4)
RULE5	'Y'	PASSWORD (RULE5 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE5)
RULE6	'Y'	PASSWORD (RULE6 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE6)
RULE7	'Y'	PASSWORD (RULE7 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE7)
RULE8	'Y'	PASSWORD (RULE8 (LENGTH (m1:m2) content-keyword (position)))
	'N'	PASSWORD (NORULE8)

Table 176. BASE segment field names (continued)

Field name	Flag byte value	SETROPTS keyword reference
NOTE: When specifying the 'Y' flag, the date supplied in the RULEn field consists of a length field and a character sequence, separated by a blank. The length field can be either a single numeric value, or two numeric values separated by a colon (:) to denote a minimum and maximum length. The character sequence conforms to the format of the output of the SETROPTS LIST command. It is a string of 1 to 8 characters, where each position of the string contains a character that indicates the valid characters that can occupy that position: • A - Alphabetic • C - Consonant • c - Mixed consonant • L - Alphanumeric • m - Mixed numeric • N - Numeric • V - Vowel • v - Mixed vowel • W - Non-vowel • * - Any character • $ - National For example, if the RULE1 field is specified with field data of "3:6 A*NV*A", the resulting SETROPTS PASSWORD keyword would be RULE1(LENGTH(3:6) ALPHA(1 6) NUMERIC(3) VOWEL(4)). See the *z/OS Security Server RACF Command Language Reference* for details on SETROPTS.		
RVARSWPW	'Y'	RVARY (SWITCH (xx))
NOTE: For ADMN_XTR_SETR, the value returned for this field is not the actual password, but one of two predefined values. A value of "DEFAULT" indicates that the default password in in effect, while a value of "INSTLN" indicates that an installation-defined password is in effect.		
RVARSTPW	'Y'	RVARY (STATUS (xx))
NOTE: For ADMN_XTR_SETR, the value returned for this field is not the actual password, but one of two predefined values. A value of "DEFAULT" indicates that the default password in in effect, while a value of "INSTLN" indicates that an installation-defined password is in effect.		
SAUDIT (boolean)	'Y'	SAUDIT
	'N'	NOSAUDIT
SECLABCT (boolean)	'Y'	SECLABELCONTROL
	'N'	NOSECLABELCONTROL
SECLANG	'Y'	LANGUAGE (SECONDARY (xx))
SESSINT	'Y'	SESSIONINTERVAL (xx)
	'N'	NOSESSIONINTERVAL
SLABAUDT (boolean)	'Y'	SECLABELAUDIT
	'N'	NOSECLABELAUDIT
SLBYSYS (boolean)	'Y'	SECLBYSYSTEM
	'N'	NOSECLBYSYSTEM

Table 176. BASE segment field names (continued)

Field name	Flag byte value	SETROPTS keyword reference
SLEVAUDT	'Y'	SECLEVELAUDIT (xx)
	'N'	NOSECLEVELAUDIT
TAPEDSN (boolean)	'Y'	TAPEDSN
	'N'	NOTAPEDSN
TERMINAL	'Y'	TERMINAL(xx)
WARNING	'Y'	PASSWORD (WARNING (xx))
	'N'	PASSWORD (NOWARNING)
WHENPROG (boolean)	'Y'	WHEN (PROGRAM)
	'N'	NOWHEN (PROGRAM)

Appendix B. Accessibility

Accessible publications for this product are offered through the z/OS Information Center, which is available at www.ibm.com/systems/z/os/zos/bkserv/.

If you experience difficulty with the accessibility of any z/OS information, please send a detailed message to mhvrcfs@us.ibm.com or to the following mailing address:

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

Accessibility features

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

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

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the z/OS Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually

exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

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

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

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

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

The following words and symbols are used next to the dotted decimal numbers:

- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1!

(KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Note:

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

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

Notices

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

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

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

Minimum supported hardware

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

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

Programming interface information

This manual is intended to describe the RACF callable services. This publication primarily documents intended Programming Interfaces that allow the customer to write programs to obtain the services of RACF.

RSA Secure code

This product contains code licensed from RSA Data Security Incorporated.

Trademarks

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

Index

Special characters

A

B

C

D

E

F

G

I

K

M

N

O

P

Q

R

S

T

U

W

Product Number: 5650-ZOS

Printed in USA

SA23-2293-00