z/OS

OA56851 - RACF ACEE Privilege Escalation Detection

This edition applies to Version 2 Release 3 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: July 18, 2019

CONTENTS

Detecting ACEE modifications

The ACEE (accessor environment element) is a control block that contains a description of a user's security environment, including user ID, current connect group, user attributes, and group authorities. An ACEE is constructed during user identification and verification, using information in the user's RACF® profile, and is anchored in the user's address space. It is referenced by RACF during command and resource authorization processing to determine if the user is allowed to perform a given action or access a given resource.

An application running in an authorized state can make modifications directly to the ACEE in storage after its creation. Such a modification would be outside the involvement of the security product, and the application must take caution to ensure that such updates do not interfere with normal system processing. Some applications might make an ACEE change to temporarily or permanently elevate the privilege of the user. This use is not necessarily malicious, and may be beneficial. For example, consider a reporting application that must access a large number of protected resources on a user's behalf. It could be infeasible to explicitly grant permission to all of the resources, and it might result in excess auditing that is unnecessary.

A security administrator might want to be aware when an application modifies an ACEE, to make sure that any increases in the user's authority complies with the organization's security policy. This can be done using the ACEECHK class.

When the ACEECHK class is active and raclisted, RACF command and authorization processing will detect certain changes made to the ACEE that affect authorization. When a modification is detected, RACF issues message IRR421I to the security console. IRR421I contains information to possibly help you determine whether the modification is expected. See *z/OS Security Server RACF Messages and Codes* for more information about message IRR421I.

If the modification is expected, a program can be added to an exception list in the ACEECHK class such that future IRR421I messages are suppressed. When RACF detects an ACEE modification, it will look in the ACEECHK class for the existence of a 'bypass' profile, indicating that the IRR421I message should be suppressed. Only the existence of the profile matters; the access list, universal access, and so forth are ignored.

To understand the bypass profiles, you must understand the ACEEs and the user IDs that are possibly involved. Every address space has an ACEE that represents the identity of the user or application associated with the address space. This ACEE is pointed to by the address space control block extension (ASXB), which can be located using well-defined system pointers. An address space might be for a single-user. For example, a TSO logon or a batch job.

An authorized application, such as a server, can work on its own behalf, or can run work under the authority of its clients. It can do the latter in one of three ways:

1. By attaching a task (or "thread") and associating an ACEE with it. Such an ACEE can likewise be located using well-defined system pointers to access the task control block (TCB).
2. By creating an ACEE for the end user and explicitly providing it to RACROUTE REQUEST=AUTH. This ACEE may be created for the life of an authorization request, or for a longer interval. The application knows the location of such ACEEs, but RACF cannot locate them using system pointers. Such an ACEE is called a 2nd party ACEE.
3. By specifying a user ID on RACROUTE REQUEST=AUTH. RACF creates the ACEE and anchors it in the caller's ACEE. Such an ACEE is called a 3rd party ACEE.

RACF command and authorization checking will check the task level ACEE if one exists and the address space level ACEE if not. In addition, authorization checking will check a 2nd and 3rd party ACEE. In this situation, it is possible for two IRR421 messages to be issued.

IRR421I displays a "call chain". This describes the flow of control of programs leading up to the currently running program, which is the program that triggered the IRR421I. This information might help in identifying the application responsible for modifying the ACEE, and so it is important to understand the possible program configurations that might exist in the environment. The job step task is typically the "top" program in an address space. It might call other programs, and these programs might call other programs and so forth. In a simple case where a batch job runs a single program that modifies its ACEE and then accesses a resource for which a system authorization facility (SAF) check is made, then that program is the one for which you would consider adding a bypass profile.

However, if this program calls another program, and it is the second program driving a SAF check, this second program is the one that IRR421I identifies as the currently running program. If it is a utility program from some system library, you would not want to create an exception for it because the exception would apply to the utility's use by all programs that call it. Now consider the case of a program or command invoked by a TSO user. As in the previous scenario, it calls a utility program that drives a SAF check. Again, you would not want to add an exception for the utility program. However, in this case, you would also not want to an add an exception for the job step program since that would belong to TSO. Adding such an exception would suppress IRR421I messages for all TSO users.

Note: It is entirely possible that the program that modified the ACEE is not in the call chain. For example, imagine if someone implemented an SVC in an authorized library to turn on a bit in the ACEE and also wrote a TSO command processor to call it. If someone issued the TSO command, then ran a different program which triggered an IRR421I message, the TSO command program name would not be identified in the call chain. As another example, the program might actually have been in the call chain, but was not the first program of a parent task with respect to the currently running program (only the first program of ancestor tasks is displayed in the call chain in IRR421I).

RACF exception checking is performed for every program in the displayed call chain, starting at the currently running program and ending with the job step program, until a matching exception is located. Therefore, it is important to understand the application environment before adding an exception.

A bypass profile name is of the format:

IRR.EXCLUDE.program-name[.user1[.user2]]

Where:

- "IRR.EXCLUDE." is a constant prefix
- program-name is the name of a program in the chain of execution
- user1 is the user ID from the address space ACEE
- If user1 is a server running work on behalf of a different user, then user2 is the user ID from the end user ACEE that is being checked (the task-level, 2nd or 3rd party ACEE). This case applies only to RACROUTE REQUEST=AUTH processing (ACEEs cannot be passed to RACF commands. The command issuer's ACEE will always be located environmentally; from either the TCB or ASXB).

RACF will look for a matching profile in the following order.

If the ACEE being checked is the address space ACEE:

1. IRR.EXCLUDE.program-name.user1
2. IRR.EXCLUDE.program-name

If the ACEE being checked is not the address space ACEE:

1. IRR.EXCLUDE.program-name.user1.user2
2. IRR.EXCLUDE.program-name.user1
3. IRR.EXCLUDE.program-name.user2
4. IRR.EXCLUDE.program-name

Note: For authorization checking, if both a task level and 2nd or 3rd party ACEE are found to have been modified, the list is checked for each user ID separately.

Examples:

1. The installation knows that program ABCXYZ modifies ACEEs in a safe manner and wants to suppress IRR421I messages when any user is running the program. Defining a profile named IRR.EXCLUDE.ABCXYZ accomplishes this.
2. A started task is designed to safely alter its own ACEE. It runs a program named STCPROG1. The administrator has set up the started task to run under user ID STCUSR1. The message is to be suppressed when STCUSR1 is running STCPROG1, but not when any other user is running STCPROG1. Defining a profile named IRR.EXCLUDE.STCPROG1.STCUSR1 accomplishes this.

Notes:

- It is recommended to audit changes to the ACEECHK class using SETROPTS AUDIT(ACEECHK)
- ACEECHK supports generic profile names. However, "backstop" profiles "*" and "**" are ignored since they essentially disable the ACEECHK class, by causing every program to be excluded from privilege escalation detection.
- When RACF program control is active, the exception list will not be checked if the environment is uncontrolled ("dirty"). IBM® recommends activating program control when activating the ACEECHK class so that any program added to the exception list is known to come from a library defined to program control. See Protecting programs in *z/OS Security Server RACF Security Administrator's Guide* for more information
- If the modification is detected in the user ID field of either of the possible ACEEs involved, the profile qualifier corresponding to the modified ACEE is not considered when looking for a match. For example, if the user ID was modified in a 2nd-party ACEE, RACF will not check for IRR.EXCLUDE.program- name.user1.user2 or IRR.EXCLUDE.program-name.user2.
- A unix program cannot be added to the exception list. If you wish to suppress IRR421I messages for a unix file, it will first need to be moved into an MVS library.

Failure option

You can request that RACF abend the running program when a modified ACEE is detected. This will result in a 4C6 abend with reason code X'ACE'. The abend is issued only if an exception was not located for a program in the call chain. To request

this failure option, define the IRR.ABEND.ON.FAILURE resource profile in the ACEECHK class and REFRESH the class. As with the exception profiles described above:

- only the existence of the profile matters.
- "backstop" profiles (* or **) are ignored.

Careful thought and planning should precede the decision to enable the failure option. You should only do so after running in production under normal workloads for a long enough interval that you are confident that you have identified all programs that are known to modify ACEEs. The results of an abend on an application can be unpredictable and largely depend on the recovery mechanisms in place for the programs running in the environment.

Z/OS SECURITY SERVER RACF MESSAGES AND CODES - SA23-2291-00

IRR421I message

IRR421I ACEE modification detected for user *user-id* **in address space ID** *addrspace-ID* **running under user** *addrsp-user-id* **and job name** *jobname* **while program** *program-name* **is running. The RACF function detecting the modification is** *racf-function***. Rsn=***reason***. [***Reason-list.* **]Occurrences** *error-count***. Command=***cmd* **| Resource=***resource-name***(***class-name***).[Profiles in the ACEECHK class were ignored because the execution environment is not clean.] Call chain:** *module1* **[<-** *module2* **…]**

Explanation: RACF detected an ACEE that was modified since its creation. This could be the result of an intentional, non-malicious update by an authorized application. However, this could also indicate the presence of malware on your system.

The user ID (*user-id*) is taken from the modified ACEE, which could be a task-level ACEE, an address space-level ACEE, or an ACEE explicitly passed to a RACF service. The address space is identified by the ASID (*addrspace-ID*), the user ID from the address space-level ACEE (*addrsp-user-id*) and the job name (*jobname*). The job name will appear as "-None-" if there is no job name associated with the address space.

The currently running program is displayed as *program-name*. *Racf-function* is the RACF module name that detected the modified ACEE.

A hexadecimal reason code (formatted as 0x*nnnnnnnn*) is displayed in *rsn*. Depending on the value of *rsn*, it might be followed by *reason-list*, which describes specific changes detected to the ACEE. *Reason-list* is displayed as a list of attributes and values, each enclosed within parentheses. For example, if the SPECIAL bit was turned on, and the user ID was changed from MICHAEL to IBMUSER, *reason-list* is displayed as "(ACEESPEC is ON) (ACEEUSRI: expected MICHAEL, actual IBMUSER)."

To mitigate console flooding, RACF throttles the number of times IRR421I is issued. The variable *error-count* indicates how many times a modification for this ACEE was detected by the task issuing the message.

Next, additional information is displayed that either identifies the RACF command that detected the function, or, if the detection occurred during an authorization check, the class name and resource name being accessed.

Before issuing IRR421I, RACF will check the ACEECHK class to see if the message should be suppressed for certain programs. However, if the execution environment is dirty (uncontrolled), then exceptions are not honored. If this is the case, the following sentence appears in the message: "Profiles in the ACEECHK class were ignored because the execution environment is not clean." Stated another way, if this sentence appears, then an existing exception profile would have resulted in suppression of IRR421I if the environment were clean. Additional messages may follow IRR421I that indicate the reason the environment became dirty. For details on program control and clean environments, see the Protecting programs in z/OS Security Server RACF Security Administrator's Guide.

Finally, the call chain of programs is displayed, starting with the currently running program, and ending with the job step program. This list contains all the programs under the current task (TCB), and the first program for each parent task, up to the job step task. The list contains as many program names as will fit in a 256-character buffer, including the delimiters between program names, starting with the currently running program.

System action: RACF proceeds with the request.

Security administrator response: If *program-name* and the call chain can be associated with a known application, contact the provider of the application and ask if any of the identified programs are expected to modify the ACEE.

The program names identified in IRR421I provide information about which programs are involved in the current chain of execution. These program names are displayed to provide some insight into the execution environment, and to possibly help identify the provider of the software currently running. It is possible that this information can help determine whether the behavior is expected. IRR421I cannot identify which program actually modified the ACEE. In fact, it is possible that the modifying program is not even in the call chain.

If it can be determined that the behavior is expected, consider adding the appropriate program to the exception list in the ACEECHK class to suppress IRR421I for this application. See the z/OS Security Server RACF Security Administrator's Guide for information on the ACEECHK class.

Destination: Descriptor code is 4. Routing code is 9.

Abend 4C6

(New text for OA56851 in **bold.**)

Explanation: An error occurred because the required control blocks were not present, or were not acceptable, when a security service was processed. A hexadecimal reason code in register 15 describes the error. See the reason code for a description of the error.

System action: The system abnormally ends the task.

System programmer response: **For reason codes other than ACE,** run the job again or have the user log on again while RACF is active. If the abend occurs again, see z/OS Security Server RACF Diagnosis Guide for information about diagnosing abends and reporting abend problems to IBM.

Programmer response: **For reason codes other than ACE,** RACF input/output parameter list IRRPCOMP contains a SAF return code, RACF return code, and RACF reason code that describes an internal RACF error. For additional information about the parameter list IRRPCOMP, see z/OS Security Server RACF Callable Services.

Code (hex)

Explanation

04

A service call to a RACF module was not completed. No accessor environment element (ACEE) was available to describe the error.

08

A service call to a RACF module was not completed. No accessor environment element extension (ACEX) was available to describe the user.

0C

A service call to a RACF module was not completed. No user security packet (USP) was available to describe the user.

ACE

A service call to a RACF module was not completed. Profiles in the ACEECHK class requested this abend because a modified ACEE was detected by the service. See z/OS Security Server RACF Security Administrator's

Guide for details on the ACEECHK class.

Z/OS SECURITY SERVER RACF SYSTEM PROGRAMMER'S GUIDE - SA23-2287-00

(New text for OA56851 in **bold.**)

Common command exit

The IRREVX01 exit provides a common exit point for most RACF commands. The exit gets control before and after the execution of all RACF commands except:

- BLKUPD
- RACDCERT
- RACLINK
- RACMAP
- RACPRIV
- RVARY
- Commands that cannot be issued from TSO, such as DISPLAY, RESTART, SET, SIGNOFF, and STOP. (For information on whether a command can be issued from TSO, see z/OS Security Server RACF Command Language Reference.)

For a list of the commands for which the exit gets control, and the code that is passed for each command, see the mapping of the EVXP data area in z/OS Security Server RACF Data Areas in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Using the information it receives about the command and the command issuer, the exit routine can:

- Modify a command before it executes
- Prevent a command from executing
- **Request that the command run with system SPECIAL or system AUDITOR authority.**

INFORMATION PASSED IN THE PARAMETER LIST

The parameter list that is passed to the exit contains:

- A flag indicating whether the exit is executing in the RACF subsystem address space or the command issuer's address space.
- A function code that identifies the command name. If the exit changes this function code, the change has no effect on RACF's processing of the command.
- A flag indicating whether this is the preprocessing or postprocessing call.
- Flags indicating whether the command was directed to the node and if so, how, with the AT or ONLYAT keywords or by automatic command direction.
- A pointer to a command buffer that contains:
    - An image of the original command after parsing.

- Quoted text strings, such as values for the ADDUSER NAME keyword, appear as entered. Note that quoted text strings might be longer than allowed because of TSO parse processing. These strings are typically truncated in the RACF database.
- If the AT or ONLYAT keyword was specified, it does not appear in the command buffer.
- All general resource names appear in the appropriate case for the class. For classes specified with CASE=ASIS in the class descriptor table, such as EJBROLE and GEJBROLE, the case is as entered by the user. For all other classes, profile names appear in uppercase.
- If the user's TSO profile is not set to NOPREFIX, the value of the PREFIX is inserted as the high-level qualifier for unquoted data sets. This value can be set to whatever the caller wants, using TSO cmd PROFILE PREFIX(value). The typical value is the user ID. However, often a group of users set a common prefix and that is used.
- For the RDEFINE, RALTER, RLIST, RDELETE, PERMIT, and ADDSD commands, if a class name was abbreviated in the command, the full name of the class appears in the command image. Exception: A profile name in the GLOBAL class (which is a class name) is left as it was entered in the command.

  - The defaults for command keywords that have defaults and were not specified on the original command.
  - Any data that was provided by prompting.
  - An extra 300 bytes of blanks following the last keyword in the buffer, where the exit can add more keywords.

  The exit can change the values of keywords in the buffer, but if it changes the command name RACF fails the command. If the exit changes the pointer to the command buffer, RACF ignores the change.

- The address of an ACEE:
  - If the address is 0, the RACF parameter library issued the command, and the command runs with the authority of the RACF subsystem address space.
  - If the address is nonzero, it points to the ACEE of the user ID under whose authority the command runs. (This user ID is usually the one that issued the command, but not necessarily. For example, for directed commands it is the user ID specified on the AT or ONLYAT keyword.) The exit can examine the user ID and group name in this ACEE to do authority checking on the command. The exit can modify fields in the ACEE that are part of the defined programming interface, but the postprocessing call should restore the previous values when it gets control after command execution or after an abend.

    Note: Modifying fields in the ACEE may result in message IRR421I being issued by RACF when the ACEECHK class is active. The output parameter field can be used to request that the command run with system SPECIAL or system AUDITOR authority. The use of this mechanism will avoid message IRR421I.

    For information about the ACEE fields, see z/OS Security Server RACF Data Areas in the z/OS Internet library (www.ibm.com/servers/resourcelink/ svc00100.nsf/pages/zosInternetLibrary).

    The exit cannot change the pointer to the ACEE.
- The originating node and user ID, if the command was directed with the AT or ONLYAT keyword, or with automatic command direction. The exit can use these values to make decisions, but cannot change them.
- A pointer to a word that the exit can use to communicate between the preprocessing call and the postprocessing call to an exit routine, or between different exit routines associated with the exit. The exit can change the contents of this communication area, but not the pointer to it.

- A command return code, an abend completion code, and a flag indicating whether the command abended:
    - On the preprocessing call, these values are 0. If the exit changes these values, the changes are ignored.
    - On the postprocessing call:
        - If the command did not abend, the command return code field contains the value set by the command processor during command execution. The exit can change this return code.

        - If the command abended, the flag is set to indicate that the abend has occurred, the abend completion code is passed, and the command return code field is set to the abend reason code, if available. If the exit changes these values, the changes are ignored.
- A pointer to a message area. If the exit fails the command with a message, it can provide message text in this area to be inserted into message IRRV022I. The exit cannot change the pointer.
- **A pointer to an output parameter area, in which the exit can set bits to affect aspects of RACF command processing, such as running the command with increased authority.**

THE PREPROCESSING CALL

Before RACF makes the preprocessing call to IRREVX01, it parses the command to ensure that it is syntactically correct. If RACF cannot successfully parse the command, it does not give control to the exit. If the command was issued as a RACF operator command, RACF verifies that the user is allowed to issue the command as an operator command based on the OPERCMDS authority check. If the user does not have the required authorization, RACF does not give control to the exit.

RACF does not process the naming convention table before making the preprocessing call. Therefore, data set names in the command buffer might be changed later by the naming conventions.

For commands that specify the AUTOUID or AUTOGID keyword, the command image contains only the AUTOUID or AUTOGID keyword. The derived UID or GID value is contained in the command image that is sent to the postprocessing call.

Based on the information passed to it in the parameter list, the exit routine can:

- Make changes to the command before the command executes, by updating the command image passed to it.
- **Specify that the command should run with system SPECIAL or system AUDITOR authority (RACF removes the authority when the command completes. The postprocessing exit has no responsibility in this regard.) Note: This function is not supported for managed ACEEs. A managed ACEE is one created using the initACEE callable service (IRRSIA00) with the INTA_MANAGED option.**
- Determine whether the command executes or fails, by setting a return code in register 15, as described in Registers at exit. If the routine returns a nonzero return code, the command fails with a return code of 8.
- Determine whether RACF issues message IRRV022I for a failed command, by setting a return code in register 15, as described in Registers at exit. The exit routine has the option to provide text to be appended to message IRRV022I.

If the exit routine makes a change to the command buffer in the preprocessing call, RACF parses the command again. RACF does this parse in noprompt mode, to prevent the user from entering values or adding keywords that the exit would find unacceptable. If this parse fails, or if RACF finds that the exit routine changed the command name, RACF fails the command and calls the exit routine for the postprocessing call, to allow the exit to clean up and reset values set by the preprocessing call.

THE POSTPROCESSING CALL

After the preprocessing call completes and RACF optionally re-parses the command, RACF processes the command. The processing of the command includes processing the naming convention table. RACF then makes the postprocessing call to the exit. If the exit changed the command buffer in the preprocessing call, RACF passes the re-parsed command image to the postprocessing call.

For commands that specify the AUTOUID or AUTOGID keyword, the command image in the postprocessing call contains both the AUTOUID or AUTOGID keyword, and the UID or GID keyword with the value RACF has derived for the UID or GID. For example, if the user entered:

ADDUSER MARTIN OMVS(AUTOUID HOME(/u/martin) PROGRAM(/bin/sh))

and RACF derived a UID value of 907, the postprocessing exit would see:

ADDUSER MARTIN OMVS(AUTOUID UID(907) HOME(/u/martin) PROGRAM(/bin/sh))

RACF makes the postprocessing call regardless of the return code from the preprocessing call. The exit receives the same parameter list on the postprocessing call as on the preprocessing call. The postprocessing call can alter the ACEE (although this is not recommended) and the communications area passed to it, and do any cleanup required due to changes made in the preprocessing call.

If the command did not abend, the command return code field contains the return code set by the command processor during command execution. The command can change this return code, and if it does, the changed value is returned to the command issuer. However, if automatic command direction is active, RACF uses the original return code to determine whether to automatically direct the command.

If the command abended, the postprocessing call receives a flag indicating that the command abended. The exit routine should do any cleanup required due to changes made in the preprocessing call, just as it would if the command had completed without an abend.

CODED EXAMPLE OF THE EXIT ROUTINE

The RACEXITS member in SYS1.SAMPLIB includes two sample IRREVX01 exit routines, IRREVX1A and IRREVX1B.

IRREVX1A illustrates how to use the IRREVX01 exit point to fail certain commands.

IRREVX1B illustrates how to use the IRREVX01 exit point to limit SPECIAL authority for certain user IDs to updating password information. The exit checks whether a FACILITY class profile of the form HELPDESK.userid exists, and if so, limits the SPECIAL authority to password updates. Note however, that generic profile checking applies to this profile lookup. If your installation already uses the FACILITY class, before you activate the IRREVX1B routine to the IRREVX01 exit point

make sure that no profile such as ** exists. Otherwise, no user IDs have SPECIAL authority until you deactivate the exit routine.

Note: Since IRREVX1B was first made available, RACF has been enhanced to provide a helpdesk function, as well as to provide a mechanism by which the exit can request that a command run with system SPECIAL authority. Thus, this sample is largely obsolete, but remains as an example of coding the exit routine.

Z/OS SECURITY SERVER RACF MACROS AND INTERFACES - SA23-2288-00

Supplied class descriptor table entries

Table 1 lists the class entries supplied by IBM in the class descriptor table (ICHRRCDX). Other classes can be added to the class descriptor table (CDT) by your installation.

If you share a database between MVS and VM systems, you can use the VM classes in the CDT.

Programming interface information: The class descriptor table (ICHRRCDX) is mapped by the ICHPCNST macro. Programming interface information for ICHPCNST is found in the data area section called CSNT/CSNX (RACF) in z/OS Security Server RACF Data Areas in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

Table 1. Classes supplied by IBM

ACEECHK CLASS (NEW)

Class	Attributes	
ACEECHK	POSIT=605	OTHER=ANY
	RACLIST=ALLOWED	MAXLNTH=246
	GENLIST=DISALLOWED	DFTRETC=4
	RACLREQ=YES	DFTUACC=NONE
		SLBLREQ=NO
		RVRSMAC=NO
	OPER=NO	KEYQUAL=0
	PROFDEF=YES	ID=1
	FIRST=ANY	CASE=UPPER
	SIGNAL=YES	
		GENERIC=ALLOWED

Z/OS SECURITY SERVER RACF DATA AREAS - GA32-0885-00 ACEE

Offset (dec)	Offset(Hex)	Type	Len	Name(Dim)	Description
...					
9	9	CHARACTER	3	ACEESBVR	Reserved for use by security product

Note: When the ACEECHK class is active, a program that updates an ACEE or user token field affecting the user's authorization may cause IRR421I messages to be issued by the security product when the ACEE is checked to determine authorization. IBM recommends that programs do not directly modify authorization-related fields in the ACEE, but instead use interfaces provided by the security product to create an ACEE with the required security attributes. In cases where this is not possible, consider documenting that your program should be added to the exception list in the ACEECHK class.

EVXP

This is the parameter list to the common command exit, IRREVX01. Additions in **BOLD.**

Offset (dec)	Offset(Hex)	Type	Len	Name(Dim)	Description
0	0	ADDRESS	4	EVXLEN	Length address: points to a fullword containing the number of fullwords in this parameter list.
...					
40	28	ADDRESS	4	EVXMSSG	Message text address: points to a 200 byte area initialized to blanks. Can be used to supply message insert for IRRV0221 when the pre-processing call sets register 15 to a value other than 0 or 4.
44	**2C**	**ADDRESS**	**4**	**EVXOPARM**	**Output parameters address: points to a flag word initialized to zeroes that can be used by the exit to request various functions.**
		1..... …….. …….. ……..		**EVXSPEC**	**Pre-exit requests to run the command with system SPECIAL authority. This is ignored if the command is running with a managed ACEE. A managed ACEE is one created using the initACEE callable service (IRRSIA00) with the INTA_MANAGED option.**
		.1..... …….. …….. ……..		**EVXAUDT**	**Pre-exit requests to run the command with system AUDITOR authority. This is ignored if the command is running with a managed ACEE. A managed ACEE is one created using the initACEE callable service (IRRSIA00) with the INTA_MANAGED option.**

Z/OS SECURITY SERVER RACF COMMAND LANGUAGE REFERENCE - SA23-2292-00

Appendix B

Table 66: Resource classes for z/OS systems.

RACF, MVS, and miscellaneous classes

Class name	Description
ACEECHK	Configuration of RACF ACEE Privilege Escalation Detection.