zSecure Command Verifier

User Guide

SC27-5648-07

Note

Before using this information and the product it supports, read the information in "Notices" on page 205.

July 2021

This edition applies to version 2, release 5, modification 0 of the IBM® Security zSecure Command Verifier (product number 5655-N19). It also applies to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

This publication provides information about installing and using IBM Security zSecure Command Verifier. IBM Security zSecure Command Verifier protects RACF® mainframe security by enforcing RACF policies as RACF commands are entered. The first three chapters provide an overview of the product purpose and product function along with installation instructions. The remaining chapters describe the auditing facilities and the installation policy profiles and provide reference information for policy definitions and messages.

This publication is intended for the following people:

- Systems support personnel responsible for the installation of IBM Security zSecure Command Verifier. See Chapter 3, “zSecure Command Verifier installation,” on page 9.
- Security administrators responsible for implementing the additional RACF command controls provided by IBM Security zSecure Command Verifier. See Chapter 5, “Policy profiles,” on page 39.
- Auditors responsible for designing and creating reports about RACF commands as issued or executed by terminal users. See the following sections:
  - Chapter 1, “Introduction,” on page 1
  - Chapter 2, “Product overview,” on page 5
  - Chapter 4, “Auditing commands and policy effects,” on page 17
  - Chapter 5, “Policy profiles,” on page 39

The IBM Security zSecure Command Verifier policies are implemented through RACF profiles. Readers must be familiar with RACF concepts and the RACF commands. People implementing the policies must have a thorough understanding of regular RACF (generic) profiles and RACF command keywords.

This manual does not provide instructions for using RACF. However, you can find RACF documentation resources listed in “Related documentation” on page xi .

zSecure documentation

The IBM Security zSecure Suite and IBM Security zSecure Manager for RACF z/VM libraries consist of unlicensed and licensed publications. This section lists both libraries and instructions to access them.

Unlicensed zSecure publications are available at IBM Documentation for IBM Security zSecure Suite (z/OS) or IBM Security zSecure Manager for RACF z/VM. For instructions to obtain the zSecure 2.5.0 licensed publications, see Obtain licensed documentation.

Obtain licensed documentation

The unlicensed zSecure 2.5.0 documentation is publicly available at IBM Documentation for IBM Security zSecure Suite. The licensed documentation is available to zSecure customers only. This document describes how to request access to the licensed documentation.

The zSecure 2.5.0 licensed documentation is available at IBM Security zSecure Suite Library.

To access the zSecure 2.5.0 licensed documentation, you must sign in to the IBM Security zSecure Suite Library with your IBM ID and password. If you do not see the licensed documentation, your IBM ID is probably not yet registered. Send a mail to zDoc@nl.ibm.com to register your IBM ID. Provide your organization's client name and number, as well as your own name and IBM ID. If you do not yet have an IBM ID, you can Create an IBM account. You will receive confirmation of registration by mail.

IBM Security zSecure Suite library

The IBM Security zSecure Suite library consists of unlicensed and licensed publications.

Unlicensed zSecure publications are available at IBM Documentation for IBM Security zSecure Suite (z/OS). Licensed publications are available to zSecure customers only. To obtain the licensed publications, see "Obtain licensed documentation" on page vii. Licensed publications have a form number that starts with L; for example, LC27-6533.

The IBM Security zSecure Suite library consists of the following publications:

- *About This Release* includes release-specific information as well as some more general information that is not zSecure-specific. The release-specific information includes the following:
  - *What's new*: Lists the new features and enhancements in zSecure 2.5.0.
  - *Release notes*: For each product release, the release notes provide important installation information, incompatibility warnings, limitations, and known problems for the IBM Security zSecure products.
  - *Documentation*: Lists and briefly describes the zSecure Suite and zSecure Manager for RACF z/VM libraries and includes instructions for obtaining the licensed publications.
  - *Related documentation*: Lists titles and links for information related to zSecure.
  - *Support for problem solving*: Solutions to problems can often be found in IBM knowledge bases or a product fix might be available. If you register with IBM Software Support, you can subscribe to IBM's weekly email notification service. IBM Support provides assistance with product defects, answers frequently asked questions, and helps to resolve problems.
- *zSecure CARLa-Driven Components Installation and Deployment Guide*, SC27-5638

  Provides information about installing and configuring the following IBM Security zSecure components:
  - IBM Security zSecure Admin
  - IBM Security zSecure Audit for RACF, CA-ACF2, and CA-Top Secret
  - IBM Security zSecure Alert for RACF and CA-ACF2
  - IBM Security zSecure Visual
  - IBM Security zSecure Adapters for SIEM for RACF, CA-ACF2, and CA-Top Secret
- *zSecure Admin and Audit for RACF Getting Started* , GI13-2324

  Provides a hands-on guide introducing IBM Security zSecure Admin and IBM Security zSecure Audit product features and user instructions for performing standard tasks and procedures. This manual is intended to help new users develop both a working knowledge of the basic IBM Security zSecure Admin and Audit for RACF system functionality and the ability to explore the other product features that are available.
- *zSecure Admin and Audit for RACF User Reference Manual*, LC27-5639 (licensed)

  Describes the product features for IBM Security zSecure Admin and IBM Security zSecure Audit. Includes user instructions to run the admin and audit features from ISPF panels. This manual also provides troubleshooting resources and instructions for installing the zSecure Collect for z/OS® component. This publication is available to licensed users only.
- *IBM Security zSecure Admin and Audit for RACF Line Commands and Primary Commands Summary*, SC27-6581

  Lists the line commands and primary (ISPF) commands with very brief explanations.
- *zSecure Audit for ACF2 Getting Started*, GI13-2325

  Describes the zSecure Audit for CA-ACF2 product features and provides user instructions for performing standard tasks and procedures such as analyzing Logon IDs, Rules, Global System Options, and running reports. The manual also includes a list of common terms for those not familiar with ACF2 terminology.
- *zSecure Audit for ACF2 User Reference Manual*, LC27-5640 (licensed)

  Explains how to use zSecure Audit for CA-ACF2 for mainframe security and monitoring. For new users, the guide provides an overview and conceptual information about using CA-ACF2 and accessing

functionality from the ISPF panels. For advanced users, the manual provides detailed reference information, troubleshooting tips, information about using zSecure Collect for z/OS, and details about user interface setup. This publication is available to licensed users only.

- *zSecure Audit for Top Secret User Reference Manual*, LC27-5641 (licensed)

  Describes the zSecure Audit for CA-Top Secret product features and provides user instructions for performing standard tasks and procedures. This publication is available to licensed users only.

- *zSecure CARLa Command Reference*, LC27-6533 (licensed)

  Provides both general and advanced user reference information about the CARLa Auditing and Reporting Language (CARLa). CARLa is a programming language that is used to create security administrative and audit reports with zSecure. The *zSecure CARLa Command Reference* also provides detailed information about the NEWLIST types and fields for selecting data and creating zSecure reports. This publication is available to licensed users only.

- *zSecure Alert User Reference Manual*, SC27-5642

  Explains how to configure, use, and troubleshoot IBM Security zSecure Alert, a real-time monitor for z/OS systems protected with the Security Server (RACF) or CA-ACF2.

- *zSecure Command Verifier User Guide*, SC27-5648

  Explains how to install and use IBM Security zSecure Command Verifier to protect RACF mainframe security by enforcing RACF policies as RACF commands are entered.

- *zSecure CICS Toolkit User Guide*, SC27-5649

  Explains how to install and use IBM Security zSecure CICS Toolkit to provide RACF administration capabilities from the CICS environment.

- *zSecure Messages Guide*, SC27-5643

  Provides a message reference for all IBM Security zSecure components. This guide describes the message types associated with each product or feature, and lists all IBM Security zSecure product messages and errors along with their severity levels sorted by message type. This guide also provides an explanation and any additional support information for each message.

- *zSecure Visual Client Manual*, SC27-5647

  Explains how to set up and use the IBM Security zSecure Visual Client to perform RACF administrative tasks from the Windows-based GUI.

Program directories are provided with the product tapes. You can also download the latest copies from Program Directories.

- *Program Directory: IBM Security zSecure CARLa-Driven Components*, GI13-2277

  This program directory is intended for the systems programmer responsible for program installation and maintenance. It contains information concerning the material and procedures associated with the installation of IBM Security zSecure CARLa-Driven Components: Admin, Audit, Visual, Alert, and the IBM Security zSecure Adapters for SIEM.

- *Program Directory: IBM Security zSecure CICS Toolkit*, GI13-2282

  This program directory is intended for the systems programmer responsible for program installation and maintenance. It contains information concerning the material and procedures associated with the installation of IBM Security zSecure CICS Toolkit.

- *Program Directory: IBM Security zSecure Command Verifier*, GI13-2284

  This program directory is intended for the systems programmer responsible for program installation and maintenance. It contains information concerning the material and procedures associated with the installation of IBM Security zSecure Command Verifier.

- *Program Directory: IBM Security zSecure Admin RACF-Offline*, GI13-2278

  This program directory is intended for the systems programmer responsible for program installation and maintenance. It contains information concerning the material and procedures associated with the installation of the IBM Security zSecure Admin RACF-Offline component of IBM Security zSecure Admin.

- Program Directories for the zSecure Administration, Auditing, and Compliance solutions:
  - 5655-N23: *Program Directory for IBM Security zSecure Administration*, GI13-2292
  - 5655-N24: *Program Directory for IBM Security zSecure Compliance and Auditing*, GI13-2294
  - 5655-N25: *Program Directory for IBM Security zSecure Compliance and Administration*, GI13-2296

IBM Security zSecure Manager for RACF z/VM library

The IBM Security zSecure Manager for RACF z/VM library consists of unlicensed and licensed publications.

Unlicensed publications are available at IBM Documentation for IBM Security zSecure Manager for RACF z/VM. Licensed publications are available to zSecure customers only. To obtain the licensed publications, see Obtain a licensed publication. Licensed publications have a form number that starts with L; for example, LCD7-5373.

The IBM Security zSecure Manager for RACF z/VM library consists of the following publications:

- *IBM Security zSecure Manager for RACF z/VM Release Information*

  For each product release, the Release Information topics provide information about new features and enhancements, incompatibility warnings, and documentation update information. You can obtain the most current version of the release information from the zSecure for z/VM documentation website at IBM Documentation for IBM Security zSecure Manager for RACF z/VM.
- *IBM Security zSecure Manager for RACF z/VM: Installation and Deployment Guide, SC27-4363*

  Provides information about installing, configuring, and deploying the product.
- *IBM Security zSecure Manager for RACF z/VM User Reference Manual, LC27-4364*

  Describes how to use the product interface and the RACF administration and audit functions. The manual provides reference information for the CARLa command language and the SELECT/LIST fields. It also provides troubleshooting resources and instructions for using the zSecure Collect component. This publication is available to licensed users only.
- *IBM Security zSecure CARLa Command Reference*, LC27-6533

  Provides both general and advanced user reference information about the CARLa Auditing and Reporting Language (CARLa). CARLa is a programming language that is used to create security administrative and audit reports with zSecure. The *zSecure CARLa Command Reference* also provides detailed information about the NEWLIST types and fields for selecting data and creating zSecure reports. This publication is available to licensed users only.
- *IBM Security zSecure Documentation CD*, LCD7-5373

  Supplies the IBM Security zSecure Manager for RACF z/VM documentation, which contains the licensed and unlicensed product documentation.
- *Program Directory for IBM Security zSecure Manager for RACF z/VM, GI11-7865*

  To use the information in this publication effectively, you must have some prerequisite knowledge that you can obtain from the program directory. The *Program Directory for IBM Security zSecure Manager for RACF z/VM* is intended for the systems programmer responsible for installing, configuring, and deploying the product. It contains information about the materials and procedures associated with installing the software. The Program Directory is provided with the product tape. You can also download the latest versions from IBM Documentation for IBM Security zSecure Manager for RACF z/VM.

Related documentation

This section includes titles and links for information related to zSecure.

See:	For:
IBM Security zSecure Suite	All zSecure unlicensed documentation. For information about what is specific for a release, system requirements, incompatibilities and so on, select the version of your choice and *About This Release*; see "What's new" and "Release notes". To obtain the zSecure licensed documentation, see Obtain licensed documentation.
IBM Documentation for z/OS	Information about z/OS. Table 1 on page xi lists some of the most useful publications for use with zSecure.
IBM Z® Multi-Factor Authentication documentation	Information about IBM Z Multi-Factor Authentication (MFA) documentation.
z/OS Security Server RACF publications	Information about z/OS Security Server Information about z/OS Security Server Resource Access Control Facility (RACF). For information about the RACF commands, and the implications of the various keywords, see the *z/OS Security Server RACF Command Language Reference* and the *z/OS Security Server RACF Security Administrator's Guide*. You can find information about the various types of events that are recorded by RACF in the *z/OS Security Server RACF Auditor's Guide*.
QRadar® DSM Configuration Guide	For more information about QRadar, see the IBM QRadar Security Intelligence Platform on IBM Documentation.
IBM MQ	Information about IBM MQ.
IBM Z NetView®	Information about IBM Z NetView.

Table 1. Some of the most useful z/OS publications for use with zSecure

Manual Title	Order Number
z/OS Communications Server: IP Configuration Guide	SC27-3650
z/OS Communications Server: IP Configuration Reference	SC27-3651
z/OS Cryptographic Services ICSF Administrator's Guide	SC14-7506
z/OS Cryptographic Services ICSF System Programmer's Guide	SC14-7507
z/OS Integrated Security Services Enterprise Identity Mapping (EIM) Guide and Reference	SA23-2297
z/OS ISPF Dialog Developer's Guide and Reference	SC19-3619
z/OS MVS Initialization and Tuning Reference	SA23-1380
z/OS MVS Programming: Assembler Services Reference, Volume 1 (ABE-HSP)	SA23-1369
z/OS MVS Programming: Assembler Services Reference, Volume 2 (IAR-XCT)	SA23-1370
z/OS MVS Programming: Authorized Assembler Services Reference, Volume 1 (ALE-DYN)	SA23-1372
z/OS MVS System Codes	SA-0665

Table 1. Some of the most useful z/OS publications for use with zSecure (continued)	
Manual Title	**Order Number**
z/OS MVS Programming: Callable Services for High Level Languages	SA23-1377
z/OS MVS System Commands	SA38-0666
z/OS MVS System Management Facilities (SMF)	SA38-0667
z/OS Security Server RACF Security Administrator's Guide	SA23-2289
z/OS Security Server RACF Auditor's Guide	SA23-2290
z/OS Security Server RACF Command Language Reference	SA23-2292
z/OS Security Server RACF Macros and Interfaces	SA23-2288
z/OS Security Server RACF Messages and Codes	SA23-2291
z/OS Security Server RACF System Programmer's Guide	SA23-2287
z/OS UNIX System Services Messages and Codes	SA23-2284
z/OS UNIX System Services Planning	GA32-0884
z/Architecture® Principles of Operation	SA22–7832

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

Technical training

For technical training information, see the IBM Training and Skills website at IBM Training.

For a list of formal customer education for IBM Security zSecure, see the zSecure Course Offerings. This PDF file is part of the zSecure - Learning information, which also includes CARLa self studies and sample applications.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM Software Support site at www.ibm.com/mysupport.

Statement of Good Security Practices

IT system security involves protecting systems and information through prevention, detection, and response to improper access from within and outside your enterprise. Improper access can result in information being altered, destroyed, misappropriated, or misused or can result in damage to or misuse of your systems, including for use in attacks on others. No IT system or product should be considered completely secure and no single product, service, or security measure can be completely effective in preventing improper use or access. IBM systems, products, and services are designed to be part of a comprehensive security approach, which will necessarily involve additional operational procedures, and may require other systems, products, or services to be most effective. IBM DOES NOT WARRANT THAT ANY SYSTEMS, PRODUCTS, OR SERVICES ARE IMMUNE FROM, OR WILL MAKE YOUR ENTERPRISE IMMUNE FROM, THE MALICIOUS OR ILLEGAL CONDUCT OF ANY PARTY.

Chapter 1. Introduction

Resources and users in the MVS Operating System can be described in RACF using profiles. Almost any resource, whether it is a data set, a CICS transaction, or something else, can be described using a *resource* profile. Historically, RACF separates data sets from other types of resources, called General Resources. This separation is also reflected in the RACF command syntax, where two distinct sets of commands are being used: one for data sets and one for general resources.

A user profile describes a user. For efficiency purposes, users are collected into RACF *groups*. These groups can be used for the following purposes:

- To access resources
- To authorize to modify profiles

Access to the resources is controlled through the resource profiles. These resource profiles can be discrete or generic.

- One *discrete* profile describes exactly one resource.
- One *generic* profile describes zero or multiple different resources.

People mostly use generic profiles. The resource profiles contain a universal access, or UACC, and two forms of Access List, or ACL. The UACC controls the access that everybody has, provided the user is not specified in the ACL. The Standard ACL is a list of users and groups and their respective access. The Conditional ACL is a list of users and some condition, such as program, terminal, or console, which is combined with their corresponding access rights.

Authorization to modify profiles is based on ownership of the profile. The owner of a profile must be an existing RACF user or group. If you specify a user as the owner, then only that particular user is authorized to maintain the profile. If you specify a group as the owner, then all people with administrative authorization in the group (that is, `group-SPECIAL`) can maintain the profile.

Like users and resources, groups are described in RACF by profiles. These group profiles, in their turn, also have an owner. The owner of a group profile can authorize people to modify the definition of the group. However, when `group-SPECIAL` is used as the authorization method, the user with `group-SPECIAL` also has authorization over many of the subgroups of the group. You can find more information about this percolation of `group-SPECIAL` authority in the *RACF Security Administrator's Guide*. See the information about user attributes at the group level in the chapter about defining groups and users.

Authorization to define, modify, and delete RACF profiles is based on the RACF profile itself. Sometimes, other authorizations such as `system-SPECIAL` can be used as well; but these authorizations are not standard authorizations available to regular users. However, RACF hardly ever controls the attribute or the new value you store into the profile in any way. For instance, if you are the owner of a group, you can connect any user in the system to your group. RACF does not verify which users you are connecting or removing. Another example is changing the owner and ACL of a profile. If you are the owner, you can specify any other user or group to be the new owner. In effect, you can give away any profile you own to anyone.

These examples are typical of the types of actions that an installation might want to prevent from occurring. Often, you want to control not only which profiles and attributes are changed, but also to what they are being changed. For example, you might want to specify that an authorized user can change the Owner value to `Mary` but not to `Joe`. With zSecure Command Verifier you can do just that:
Control the new value of any field, option, or attribute of any profile.

RACF commands

Using the RACF commands, you can add, change, or delete RACF profiles and define system-wide options.

Only users that are defined by RACF can issue RACF commands. Although you can issue most RACF commands, RACF verifies that you are authorized to issue the command against the profile that is

specified in the command. Most of the RACF commands can be issued from the TSO environment. In addition, there are some RACF commands that can be issued only from the MVS operator console. Nowadays, the traditional TSO RACF commands are no longer restricted to the TSO environment. The commands can also be issued from the operator console, the RACF Parameter Library, and through an R-Admin RACF Callable Service. When issued from the operator console, the console operator must be logged on, and the authorization is based on the user ID of the operator.

Historically, RACF commands are grouped by the type of profile. This grouping is still useful for users and groups, but less so for data sets and the General Resources. In particular, the **PERMIT** command often confuses people when they try to authorize access to a general resource profile. The long history of RACF can be seen in the implementation of discrete profiles and the user attributes that can be used to automatically set data set attributes like UACC. Using the RACF ISPF panel interface alleviates some of the problems that are caused by the history and compatibility with an earlier version of RACF.

Some command-related problems are caused by the basic philosophy of RACF:

> The owner of a profile can change any attribute of the profile as long as changing the attribute does not increase the authority or access of the owner.

Some installations do not want to allow their users this flexibility. Users can change the access rules to effectively disable RACF access control for their resources. You can reduce this exposure by disallowing the RACF commands entirely, or through coding of exits. Both solutions have their drawbacks. Disallowing the commands (for example, through RACF program control) also prevents legitimate changes that an owner of a resource might want to make. The standard RACF exits often do not provide the amount of control that is required by the installation. See “RACF command exits” on page 2. Subsequent sections describe how zSecure Command Verifier introduces an extra flexible control point.

RACF command exits

This section describes several types of exits supported by RACF. Most of these are ill-suited for the purpose of verifying the values of fields in RACF profiles. zSecure Command Verifier provides an interface where the authorization to manage fields and field values is defined using simple policy profiles.

The first category consists of exits that are not even RACF exits, but MVS exits: The System Authorization Facility (SAF) exits. These exits are started for all instances in which any system component needs a function that is provided by the security product. However, for most of the RACF commands, these exits are not started because there is not yet any profile to be verified. In other situations, the RACF command does the verification itself, based on information already in storage, or retrieved directly from the RACF database.

The second category of exits consists of the RACF SVC-processing exits. During RACF SVC-processing, a preprocessing and a post-processing exit are started. These exits are primarily intended to change the behavior of RACF for low-level functions in a limited way. It is possible to misuse these exits and include more processing, but that is not an intended function of the exit. In addition, some RACF commands do not use the specific RACROUTE requests that use these exits.

A third category of exits is that for data set naming conventions. As the name implies, these exits are only started during RACF command processing if a data set name is present or implied. However, for most commands, no data set profile is involved, and thus none of these exits is called.

The next category of exits comprises the password-related exits. The new password exit is called only when a password or password interval is changed. The encryption exit is called when the new password must be encrypted in the RACF database. These exits are not called for those commands that do not involve passwords or other encrypted data.

RACF also provides an ACEE Compression/ Expansion exit, but that exit, similarly to the RACFRW exit, is not relevant to RACF command processing.

RACF provides a Common Command exit. This exit is called for most RACF commands. The command string is passed as a single argument, adding all the complexity of parsing and interpreting its contents to the exit. The exit:

- Can either disallow or change the command string.

- Cannot change the command name.
- Cannot generate more commands.

Standard RACF command exits

There are several reasons for using the standard RACF command exits.

The *RACF System Programmers Guide* mentions some examples of usage of the RACF exits:

- Controlling password quality
- Restricting a SPECIAL user to resume and password reset

Other purposes for the RACF exits are sometimes used as well. In some installations, RACF exits are used for the following purposes:

- Enforcing a smaller password interval for selected users
- Setting auditing attributes for users with non-standard authorizations
- Preventing changes to the UACC of data sets
- Preventing addition of the user ID "*" to an access list

The password quality control can be done by comparison of the new password against a list of forbidden words or against the characters in the current password. An example is testing for keyboard patterns like QWERTY and LKJHGF, or months like MARCH and APRIL. Comparison against current password can involve things like `more than 3 characters in the same position`. For this second test, an example of an invalid password is QP11AL if the current password is QP10AL.

The following description illustrates one of the more advanced uses of the RACF exits. This example describes an attempt to prevent changes to the UACC of data sets.

Preventing changes to the UACC of data sets involves several RACF exits. The first is the one called for the special form of the `RACROUTE REQUEST=AUTH` used internally in the RACF commands. On its own, this exit is not enough. It must be combined with the RACF exit called in `RACROUTE REQUEST=DEFINE`. However, even this combination is not foolproof against all possible ways that a user can influence the UACC setting of a data set profile.

These type of exits are insufficient to control all desired aspects of profiles. The RACF Common Command allows inspection and change of many RACF keywords. It is called before and after execution of most RACF commands. However, RACF places the following restrictions on this exit:

- The RRSF keywords `AT` and `ONLYAT` are already processed and stripped from the command.
- You cannot check which keywords are provided by the terminal user because defaults are already supplied.
- You cannot change the command itself.

In addition, coding such an exit is not trivial mainly because the keywords are presented in the form of a long character string. Processing of the TSO command syntax, including parenthesis and quoted strings, is considered a complicated and difficult task by many people. It is partly for this reason that zSecure Command Verifier is an effective way to implement additional security controls. Another advantage of zSecure Command Verifier is that no assembler or other programming skills are required. The installation policy rules can be defined by policy profiles. zSecure Command Verifier takes care of parsing, verification, error messages, and generation of the audit trail.

Advantages of using zSecure Command Verifier to monitor RACF

Read this information to understand how zSecure Command Verifier accesses and processes RACF commands.

zSecure Command Verifier intercepts RACF commands at an earlier stage than most other exits provided by RACF. Thus, the installation can verify keywords on the RACF commands before any significant RACF processing takes place. The installation can also change keywords in such a way that the RACF command processors cannot distinguish these modified keywords from those keywords that are entered by the

terminal user. On the other hand, zSecure Command Verifier intercepts at a late enough stage to allow normal TSO command keyword prompting to take place. However, this latter feature is not supported for all keywords. Some keyword validations can be done only during the final processing of the command, and are thus not eligible for terminal prompting.

Although it is possible for the console operator to issue RACF commands, not all operator commands are intercepted by zSecure Command Verifier. zSecure Command Verifier does not intercept the original operator commands like **DISPLAY** and **SIGNOFF**, but it does intercept the other RACF commands like **ALTUSER** and **LISTUSER**.

z/OS also provides a USS callable service to execute RACF functions. This R_Admin service can execute some predefined functions, but also all TSO RACF commands. These RACF commands are executed in the RACF address space under the authority of the RACF user ID associated with the USS process. Because these commands also invoke the standard RACF Common Command exit (IRREVX01), they can also be controlled by zSecure Command Verifier.

The current version of zSecure Command Verifier does not differentiate between the various sources of RACF commands or the execution environment. The execution environment includes TSO, Operator command, and RRSF propagated command, or R_Admin command.

Prerequisite software

Ensure that the software support levels in this topic are met before you implement zSecure Command Verifier policy profiles.

zSecure Command Verifier requires at least the software levels that are indicated in Table 2 on page 4 for correct installation and functioning. You might be able to install parts of the product on lower releases of RACF but such usage is not supported. zSecure Command Verifier is tested and supported on the following levels:

Table 2. Required software and levels for zSecure Command Verifier

Product	**Supported level**
z/OS	2.1 and higher
SMP/E for z/OS	3.5 or later

Chapter 2. Product overview

zSecure Command Verifier is implemented as an exit to the RACF commands. It uses the RACF Common Command exit (IRREVX01).

The zSecure Command Verifier routines are started by the RACF command processor. These routines scan the keywords and parameters as entered by the terminal user and pass them to the RACF command processor. Only those keywords that are accepted are passed to the RACF command processor. In the following description, the user who issues the command is called the terminal user. This term also applies to all other methods of issuing RACF commands. For example, issuing a command from the MVS operator console or from a Batch job. The following steps are performed:

1. The command as entered by the terminal user is analyzed and verified. Obvious syntax errors are reported back to the user and the user must correct them. This process is like the standard RACF command interface.
2. If the command is syntactically correct, the keywords and parameters are translated into an internal format. This format permits the zSecure Command Verifier installation policy interpretation and enforcement routines to easily access the keywords and parameters.
3. The keywords and parameters are evaluated and matched against the installation policies as specified in the C4R profiles in the XFACILIT resource class. If the keywords and parameters violate the installation policy, the command is rejected, or the keyword is suppressed.

   The installation policy profiles can specify one pre-command, which is executed before the user-entered RACF command, and one post-command, which is executed after the user-entered RACF command.
4. The command or commands are executed. Normally, the authority of the terminal-user to execute the commands is not modified by zSecure Command Verifier. It is possible that the commands fail because of insufficient authority. It must be noted that the policy specified pre-commands and post-commands are executed independently of the acceptance of the terminal-user-specified command, keywords, or parameters. In addition, a complete ABEND of one of the commands can result in the termination of the entire set of commands such as pre-command, policy-accepted-command, and post-command.
5. Auditing is being done through an Audit-only RACROUTE REQUEST=AUTH for a specific C4R profile in the XFACILIT resource class. For more information about these profiles and the related auditing options, see “Profile audits in zSecure Command Verifier” on page 6.

How RACF processes commands

zSecure Command Verifier verifies all RACF commands that invoke the RACF Common Command exit. In processing the verified RACF commands, there is a minor incompatibility with the way RACF normally handles these commands. The incompatibility is related to the processing of repeated keywords.

In some situations, RACF processes all parameter values for repeated keywords, while in other situations only the last specification is used. For example:

- If you specify `ALTUSER` *`userid`* `CICS(ADDOPCLASS(01) ADDOPCLASS(02))`, RACF adds both OPCLASSes.
- In contrast, if you specify `ALTUSER` *`userid`* `ADDCAT(cat1) ADDCAT(cat2)`, RACF adds only a single CATEGORY.

The zSecure Command Verifier uses only the last specified value of any keyword, and all other specifications of the same keyword are ignored.

RACF commands that do not start the Common Command exit

Some commands cannot be handled by zSecure Command Verifier or they are handled in an exceptional manner.

The following list describes these commands.

RVARY
Because of the high risk of causing extra problems when you implement protection for the **RVARY** command, support for **RVARY** is not implemented in zSecure Command Verifier. In the RACF product code, **RVARY** has a special position that is based on similar availability concerns.

RACLINK
In the RACF implementation of RACF Remote Sharing Facility (RRSF) the **RACLINK** command and its keywords and parameters are protected by RACF profiles in the RRSFDATA resource class.

RACDCERT
RACF implemented the **RACDCERT** command as a command that is not handled by the RACF command envelope. As a result, it does not start the Common Command exit point. Fortunately, the use of the **RACDCERT** command is controlled by SPECIAL and profiles in the FACILITY class. The following text is taken from the *RACF Command Language Reference*:

To issue the **RACDCERT** command, you must have one of the following authorities:

- SPECIAL
- Sufficient authority to resource IRR.DIGTCERT.*function* in the FACILITY class, where *function* is LIST, ADD, ALTER, or DELETE
- READ access to IRR.DIGTCERT.*function* to run the function for yourself
- UPDATE access to IRR.DIGTCERT.*function* to run the function for others

This combination of existing controls reduces the need for zSecure Command Verifier controls.

RACPRIV
The **RACPRIV** command affects only the status of the write-down privilege in the address space of the user. It has no impact on profiles in the RACF database, and cannot be propagated to other systems. Use of the command is partially controlled by the IRR.WRITEDOWN.BYUSER profile. For this reason, no additional controls are implemented in zSecure Command Verifier.

RACMAP
The **RACMAP** command creates, deletes, and lists a distributed identity filter. The command is not eligible for routing to other RRSF nodes that use command direction, and it does not start any RACF exit. The command can be controlled by using profiles in the FACILITY class of the form IRR.IDIDMAP.*function*, where *function* is MAP, DELMAP, or LISTMAP.

For more information about the **RACMAP** command, see the *RACF Command Language Reference*.

Installation policies

In zSecure Command Verifier, a point is needed for the decisions about the commands, keywords, and parameters. Because there are many different options possible, the installation must specify which decisions must be made based on which criteria. This is done through the definition of policy profiles in the XFACILIT resource class.

For information about these policy profiles, see Chapter 5, “Policy profiles,” on page 39.

Profile audits in zSecure Command Verifier

Using the definition of specific C4R profiles in the XFACILIT resource class, an auditor can specify the events that are logged to SMF.

For every RACF command, the product can log the command as entered by the terminal user and the command as finally executed. It is also possible to specify that both must be logged. The auditor can

indicate that auditing through SMF must be done only for selected users, or for all users. For each command, only the first 255 characters are available from the generated SMF records.

The error message as issued to the terminal user is available as LOGSTR in the SMF record from the C4R.ERRMSG.*command* profile.

An example of a profile that is used for audit specification is shown. It indicates that the **ADDUSER** command must be audited for all users before it is processed according to the installation policy profiles. When **ADDUSER** is issued by user IBMUSER, it must not be audited.

```
C4R.PREAUD.ADDUSER UACC(READ) AUDIT(SUCCESS(READ))
             ACL: IBMUSER NONE
```

Note: Any sensitive fields like passwords and encryption keys are removed from the RACF command before it is logged in SMF.

For more information, see Chapter 4, “Auditing commands and policy effects,” on page 17.

Product version and status verification with the C4RSTAT command

Use the **C4RSTAT** command to verify the status and version of zSecure Command Verifier.

The **C4RSTAT** command verifies that the main code is activated as part of the RACF Common Command exit, and that the Policy Interpretation and Enforcement Routine, or C4RPIER can be located for the current session. The command also shows the resource class that is used for the Policy Profiles, and the number of Policy Profiles defined.

The following figure lists example output from the **C4RSTAT** command:

```
C4R982I zSecure Command Verifier is active
C4R971I EXIT version is 2.5.0
C4R973I PIER version is 2.5.0
C4R985I Resource class used for policy profiles is XFACILIT
C4R976I Resource class is active
C4R969I Generic profiles are enabled
C4R978I Number of policy profiles is 29
```

Figure 1. CR4STAT command output

Chapter 3. zSecure Command Verifier installation

Use the guidelines in these topics to install zSecure Command Verifier into the production environment.

The installation of zSecure Command Verifier is done through SMP/E.

Before you install, verify that you have the zSecure Command Verifier version that matches the level of z/OS with RACF that is active on your system.

Installation preparation

Use this information to understand how zSecure Command Verifier is packaged for installation.

zSecure Command Verifier is shipped in the form of two SMP/E FUNCTIONs. The first FUNCTION contains all the parsing and command building code. The second FUNCTION contains all the policy control code. Both functions are required to use the product.

The installation of zSecure Command Verifier consists of several steps. Aside from the creation of the executable modules in the system libraries, you must issue some operator commands or make updates to a parmlib member to activate the product.

Resource class selection

Use these guidelines to update the installation module C4REXP and to define a resource class with the same attributes as the XFACILIT class to modify the resource class.

zSecure Command Verifier policy rules are defined through profiles in the XFACILIT resource class. It is possible to use a different resource class; however, it is best to use the default resource class.

If you want to modify the resource class, you must update installation module C4REXP, and define a resource class with the same attributes as the XFACILIT class.

The administrator defines the zSecure Command Verifier policy rules through profiles in the XFACILIT resource class. A different resource class can be used. However, it is best to use the default resource class.

The resource class that is needed for the zSecure Command Verifier control profiles must have the following characteristics:

- Maximum profile length 246 characters.
- The default return code can be 4 or 8. In most cases, zSecure Command Verifier ignores the default return code.
- First character must be specified as alphanumeric, and other characters must be specified as allowing any character.
- RACLIST must be allowed for performance reasons. You can decide to RACLIST the resource class through SETROPTS or not. If the resource class is not SETROPTS RACLISTed, it is GLOBAL ONLY RACLISTed at the first RACF command invocation.

Overview of installation steps

Use this checklist to track the tasks completed during the zSecure Command Verifier installation process. The checklist provides an overview of the installation process and links to more information about each task.

See the links to procedures in the table for instructions for completing each task.

Table 3. Installation checklist for SMP/E installation		
Step	**Procedure**	**Jobname**
1	“Step 1: Define the data set naming conventions” on page 10	
2	“Step 2: Load the installation JCL” on page 10	
3	“Step 3: Create and initialize SMP/E zones” on page 10	C4RJSMPA C4RJSMPB C4RJSMPC
4	“Step 4: Receive the SYSMODs” on page 12	C4RJREC
5	“Step 5: Allocate the TARGET and DLIB data sets” on page 12	C4RJALL
6	“Step 6: Update SMP/E DDDEFs” on page 12	C4RJDDD
7	“Step 7: Add the zSecure Command Verifier code” on page 13	C4RJAPP
8	“Step 8: Specify the resource class for policy profiles” on page 13	C4RJEXP
9	“Step 9: Update the parmlib for APF-authorized TSO commands” on page 13	C4RJIKJ
10	“Step 10: Activate and test zSecure Command Verifier” on page 14	Parmlib Operator Commands
11	“Step 11: Accept the zSecure Command Verifier product” on page 15	C4RJACC

Step 1: Define the data set naming conventions

Before you install SMP/E, establish the data set naming conventions that you want to use during the installation process.

Define conventions for all required data set types, including the following data sets:

- Data set with installation JCL or SC4RINST
- SMP/E control data sets like CSI, PTS, and others
- System data sets for the installed software

Step 2: Load the installation JCL

The JCL used during the zSecure Command Verifier installation process is in the SC4RINST data set.

If you install from tape, use the following JCL to copy the following data to a DASD data set.

```
//jobname   JOB (account info),'Copy install JCL',
//             CLASS=a,MSGCLASS=r
//*-------------------------------------------------------------
//FILE8  EXEC PGM=IEBCOPY
//SYSUT2   DD DISP=(NEW,CATLG),UNIT=SYSALLDA,SPACE=(CYL,(1,1,10)),
//            DSN=userid.C4R240.INSTJCL
//SYSUT1   DD DISP=SHR,VOL=(,RETAIN,SER=C4R240),UNIT=3480,
//            LABEL=(6,SL),DSN=IBM.JC4R240.F3
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
```

After successful execution of this job, you can continue with “Step 3: Create and initialize SMP/E zones” on page 10.

Step 3: Create and initialize SMP/E zones

Before starting the zSecure Command Verifier installation process, determine the SMP/E zones for the installation.

You can choose from the following installation options:

- Install in existing z/OS zones
- Install in new (dedicated) zones in an existing CSI
- Install in new (dedicated) zones in a new CSI

Sample installation jobs are only provided for the third option.

If you decide to install the product in existing zones, you do not need to define any SMP/E CSIs or zones. You can immediately continue with the procedure "Step 4: Receive the SYSMODs" on page 12.

If you decide to install in dedicated zSecure Command Verifier zones by using new or existing CSIs, complete the pre-installation steps by using the example jobs that are provided in SC4RINST before you continue the installation process. For details on the pre-installation steps, see "Preinstallation tasks" on page 11.

The example jobs that are provided in the SC4RINST all use lowercase strings for the values that must be adapted to fit your installation standards. The following are the values that are currently used.

Your-Global
The data set prefix that you want to use for the GLOBAL SMP/E data sets. This prefix is used for the name of the GLOBAL CSI and for the SMP/E data sets shared between all SMP/E zones.

Your-Product
The data set prefix that you want to use for the zSecure Command Verifier data sets. This data set is also the prefix for the SMP/E data sets specific to zSecure Command Verifier.

SYSALLDA
The unit name that is used for all data set allocations.

volser
The name of the DASD volume in your system where you want to create the zSecure Command Verifier data sets. In an SMS environment, the ACS routines can assign another volume than the one specified by the *volser*.

tape
The unit name of the tape-unit where the zSecure Command Verifier distribution tape can be mounted.

Note: The value for *Your-Global* cannot be the same as *Your-Product*. If you want to use similar prefixes, you can add a qualifier for the GLOBAL zone. For example, you can use the following values.

- SMPE.CMDVFY.GLOBAL as the value for *Your-Global*
- SMPE.CMDVFY as the value for *Your-Product*

Table 4. Pre-installation variable values used to define SMP/E zones

Variable	Your Value
Your-Global	
Your-Product	
sysda	
volser	
tape	

Preinstallation tasks

This list provides the tasks that you must complete before you install the zSecure Command Verifier.

1. Create and initialize GLOBAL CSI and GLOBAL ZONE:

   If you want to use an existing GLOBAL zone, you can skip the definition of the GLOBAL CSI, the GLOBAL ZONE, and the related data sets. In that case, continue with the next step. If you want to

create a GLOBAL zone, start with running sample job C4RJSMPA to define and initialize the data sets for the GLOBAL zone.

```
Submit C4RJSMPA
```

2. Create zSecure Command Verifier TARGET and DLIB ZONES:

   The zSecure Command Verifier TARGET and DLIB ZONES can be created in their own CSI. The sample job that is provided creates a product CSI and defines two SMP/E zones in that CSI.

   ```
   Submit C4RJSMPB
   ```

3. Create the OPTIONS entry in the zSecure Command Verifier ZONEs:

   The next job is used to specify the OPTIONS entry that is used during the subsequent SMP/E installation steps that follow.

   ```
   Submit C4RJSMPC
   ```

Step 4: Receive the SYSMODs

If you are installing from the zSecure Command Verifier product tape, the first file on the tape is the SMPMCS data set.

This data set contains the SMP/E Modification Control Statements that are needed for correct installation of zSecure Command Verifier. In this situation, you can use sample job **C4RJREC** to RECEIVE the product.

```
Submit C4RJREC
```

Step 5: Allocate the TARGET and DLIB data sets

You can use job **C4RJALL** to allocate required data sets.

zSecure Command Verifier adds four target data sets and four distribution data sets to your SMP/E environment. See Table 5 on page 12 for the sizes and attributes of these data sets.

Table 5. Target and Dlib data sets needed for zSecure Command Verifier

DDname	Type	Recfm	Blksize (suggested)	Lrecl	Space (tracks)	Dir
AC4RLNK	DLIB	U	32760	N/A	25	20
SC4RLNK	Target	U	32760	N/A	15	2
AC4RSMP	DLIB	FB	27920	80	2	2
SC4RSMP	Target	FB	27920	80	2	2
AC4RINST	DLIB	FB	27920	80	3	3
SC4RINST	Target	FB	27920	80	3	3

For example, job **C4RJALL** contains the necessary JCL to allocate the required TARGET and DLIB data sets.

```
Submit C4RJALL
```

Step 6: Update SMP/E DDDEFs

You can use job **C4RJDDD** to define the allocated data sets to SMP/E.

In this step, the data sets that you allocated in the previous step are defined to SMP/E. If you decide to include appropriate DD-statements in all your SMP/E jobs, you can omit this step. If you want to use the preferred setup through dynamic allocation, this step is required. The example job **C4RJDDD** contains the JCL needed for this step.

```
Submit C4RJDDD
```

Step 7: Add the zSecure Command Verifier code

During this step, use the following SMP/E statement to add the zSecure Command Verifier code, examples, and documentation to the system.

```
APPLY  SELECT(JC4R240,HC4R240) GROUPEXTEND.
```

Because of the use of a **SELECT** for the product FMID, SMP/E does not require the use of the **FUNCTIONS** keyword. An example job is included in member **C4RJAPP**. Before you run this job, specify the data set name of your GLOBAL CSI.

Submit C4RJAPP

Step 8: Specify the resource class for policy profiles

You can specify the resource class in zSecure Command Verifier that is used for all installation policy profiles. The default value that is provided is XFACILIT.

Normally, you do not need to change the resource class. If your installation requires a different setting of the resource class, review and submit the sample job C4RJEXP. The first time that you run this job, it is likely to end with a return code **12**. It is caused by an inline SMP/E REJECT step that ensures that you can run the same job multiple times.

Submit C4RJEXP

Review the following fields:

RSVDx
: These fields are reserved. Do not modify them unless you are instructed otherwise by zSecure Command Verifier support personnel.

CLASS
: Complete the resource class that you want to use for the zSecure Command Verifier policy profiles. The default name is XFACILIT.

Step 9: Update the parmlib for APF-authorized TSO commands

Update the parmlib to enable the use of the APF-authorized modules that contain required TSO commands.

zSecure Command Verifier requires two APF-authorized TSO commands. The first command displays information about the current state of the zSecure Command Verifier module, which can be active or inactive, and about the resource class currently in use. The other command displays and manages the Command Audit Trail information in various profiles. These APF-authorized modules are installed in the SC4RLNK library. To enable use of these modules as a TSO command, you must add their names to the TSO-authorized command table in **PARMLIB**. Add lines like the following ones to the AUTHCMD section in your IKJTS0xx member. After you update the member, you can activate it using the **TSO PARMLIB UPDATE xx** command.

```
AUTHCMD NAMES(          /* AUTHORIZED COMMANDS      */      +
   ...  Leave the first part of this list of commands as is.
   ...  Insert the following line at the end of the list.
   C4RSTAT             /* zSecure Command Verifier status disp*/ +
   C4RCATMN            /* zSecure Command Verifier Audit Trail*/ +
   ...  Ensure that the last line in the AUTHCMD block ends
   ...  with a right parenthesis as shown below.
   ...                 /* SOME COMMENT             */)
   ...  Rest of member need not be modified.
```

An example parmlib member that describes the required changes is provided in member **C4RJIKJ**.

Step 10: Activate and test zSecure Command Verifier

Use this information to activate zSecure Command Verifier without a system IPL and test its operation.

Because zSecure Command Verifier is implemented through the dynamic exit facility, it is possible to activate zSecure Command Verifier immediately without a prior IPL of your system. A prerequisite for this type of implementation is that the two main routines of zSecure Command Verifier are in standard *linklist* libraries. You can use one of the following methods:

- Install the product directly in the system libraries. Although installing directly in active system libraries works, it is considered to be bad systems programming practice.
- Copy the modules from the SMP/E controlled SC4RLNK data set to another data set that is already part of the *linklist*.
- Use the z/OS dynamic *linklist* facility to add the SC4RLNK data set to the active *linklist*.

In all of the preceding cases, you must also issue the **F LLA,REFRESH** operator command before you attempt to activate the **C4RMAIN** exit. Do not use a directed load through the DSN keyword on the **SETPROG** command. The **C4RPIER** module cannot be activated in this way; it must be present in an active *linklist* library or STEPLIB.

To add the zSecure Command Verifier library to the active APF list, add a member like the following example to **PARMLIB**, and run the **T PROG=xx** operator command:

```
APF ADD DSNAME(Your-Product.SC4RLNK)                SMS
```

To add the zSecure Command Verifier library to the active *linklist*, add a member like the following example to **PARMLIB**, and run the **T PROG=xx** operator command:

```
LNKLST DEFINE NAME(LNKLSTC4) COPYFROM(CURRENT)
LNKLST ADD NAME(LNKLSTC4) DSN(Your-Product.SC4RLNK)
LNKLST ACTIVATE NAME(LNKLSTC4)
LNKLST UPDATE,JOB=*
```

To activate zSecure Command Verifier, add a member like the following example to **PARMLIB**, and run the **T PROG=xx** operator command.

```
EXIT ADD EXITNAME(IRREVX01) MODNAME(C4RMAIN) STATE(ACTIVE)
```

Alternatively, you can run the following **OPERATOR** command directly:

```
SETPROG EXIT,ADD,EXITNAME=IRREVX01,MODNAME=C4RMAIN,STATE=ACTIVE
```

Note that this direct operator command does not persist over an IPL.

If you must remove zSecure Command Verifier, use the following **OPERATOR** command:

```
SETPROG EXIT,DELETE,EXITNAME=IRREVX01,MODNAME=C4RMAIN
```

The **C4RSTAT** command is an APF-authorized TSO command that can be used to display if the zSecure Command Verifier module is active, and to provide information about the currently used resource class. If zSecure Command Verifier is installed and active, the output of the **C4RSTAT** command now looks like the following example:

```
C4R982I zSecure Command Verifier is active
C4R971I EXIT version is 2.5.0
C4R973I PIER version is 2.5.0
C4R985I Resource class used for policy profiles is XFACILIT
C4R976I Resource class is active
C4R969I Generic profiles are enabled
C4R978I Number of policy profiles is 0
```

To test zSecure Command Verifier after you activate it, issue some RACF commands that must work or fail exactly as you expect. For example, a **LISTUSER** command without any keywords must still show information for your own `userid`. If you also want to verify that zSecure Command Verifier policy profiles are interpreted as you want, you can use sample job **C4RJTST** in SC4RSMP. This sample job defines

several system-wide policies and issues some commands that show the effects of these policies. These policies apply to all users, and might thus affect other users on the system or sysplex where this job is executed. Inspect the policy profiles in this sample to ensure that they are appropriate for the system where you are installing zSecure Command Verifier. Use the sample only in a test environment. The sample job consists of several parts:

- Defining several sample policy profiles. Some of these affect only messages that are being issued to the terminal user, while others affect the creation of user profiles and data set profiles.
- Issuing several RACF commands that fail or are modified. The successful commands must be echoed at the terminal, as part of the **C4R913I** message. The commands that violate one of the defined policies must receive RACF violation messages (ICH408I) because of insufficient access to the policy profile.
- Removing the sample policy profiles, thus returning the system to the state before you run the test job.
- Removing the user and data set profiles that were created as the result of the test commands.

Running the sample test job requires non-standard RACF authorizations. At a minimum, `CLAUTH` in the `XFACILIT` or the alternative resource class that is specified in "Step 8: Specify the resource class for policy profiles" on page 13 and the USER class, and group-special in the current connect group is required. It is also possible to use only System-SPECIAL authority for the entire job. In that case, `CLAUTH` and `Group-SPECIAL` are not needed.

Attention: This sample job defines system-wide policies. These policies apply to all users, and might thus affect other users on the system/sysplex where this job is executed. Inspect the commands as issued, and evaluate if execution of this sample installation verification procedure is appropriate for your environment.

```
Adapt and submit optional job C4RJTST from SC4RSMP
```

If the commands work as expected and you are satisfied with the results, continue with the next step, which is accepting the installation of zSecure Command Verifier code on your system.

Step 11: Accept the zSecure Command Verifier product

If you are satisfied with the implementation of zSecure Command Verifier, run an **ACCEPT** job to integrate the product with your system.

An example **ACCEPT** job is provided in `C4RJACC`. After you run this job, there is no need for any further system programming work to use zSecure Command Verifier.

```
Submit C4RJACC
```

Resources for auditing profiles

After you install zSecure Command Verifier, the auditor for your installation might need to create policy profiles to activate zSecure Command Verifier auditing.

To allow your auditor to specify when and what to audit, zSecure Command Verifier implemented an extra **RACROUTE REQUEST=AUTH** for a dummy resource. This resource is the name of the command that is being issued prefixed by `C4R`. The profile must be defined in the `XFACILIT` class. If you do not define any profiles, no additional auditing of zSecure Command Verifier processed commands is available. You cannot modify normal RACF command auditing by using any zSecure Command Verifier setting. During the installation process, contact the auditor to specify which profiles must be defined and what auditing options must be activated for these profiles. For more information about the profiles and auditing options available, see Chapter 4, "Auditing commands and policy effects," on page 17.

Chapter 4. Auditing commands and policy effects

zSecure Command Verifier provides various functions for auditing both the commands as issued and the effects of the implemented policies.

The following list contains the available auditing functions:

Command Audit Trail function
Records information about the RACF commands that are issued in the affected profiles themselves. By using easy RACF list commands, it is possible to obtain information about who last changed a particular part of a profile, like the OWNER, the UACC, or the Access List. For more information, see “Command audit trail” on page 17.

Policy Profile Effect function
Records information about the effect of the zSecure Command Verifier policy profiles through SMF records. The RACF command before and after the processing of the Policy Profiles is recorded in LOGSTRING information for access to special Policy Profile Effect recording profiles. For more information, see “Policy profiles for command auditing” on page 34.

SMF access recording

You can also use regular SMF access recording for the Policy Profiles themselves. For more information, see “Regular access recording through SMF” on page 37.

Command audit trail

zSecure Command Verifier provides a function to collect and retain additional data about issued commands in the RACF profiles affected by these commands.

For example, if user C4RTEST issues the command **ALTUSER IBMUSER RESTRICTED**, information is saved in the IBMUSER profile. The information includes an indication for the RESTRICTED attribute, the date and time, and the userid C4RTEST. The information that is retained can be used as a Command Audit Trail. The same information can usually be obtained from SMF audit records. However, the zSecure Command Verifier function has the advantage of finding the same information quicker, and without processing potentially large amounts of SMF data. An example of this Command Audit Trail information for a USER, as maintained by zSecure Command Verifier is shown:

```
Command Audit Trail for USER C4RUSER
Profile:          Created on 20.238/14:24 by C4RTEST
Segment:  CICS    Added on 20.241/03:19 by C4RTEST
                  Changed on 20.241/03:20 by C4RTEST
          TSO     Changed on 20.241/03:19 by C4RTEST
Attrib:   PASSWRD Removed on 20.238/14:24 by C4RTEST
          INTERV  Changed on 20.241/04:42 by C4RTEST
          RESTR   Added on 20.238/14:24 by C4RTEST
Connect:          BCSC Added on 20.238/14:24 by C4RTEST
GrpAttr:  ADSP    BCSC Removed on 20.238/14:24 by C4RTEST
```

Figure 2. Command Audit Trail data for a user

The data is maintained in the **USRDATA** fields in each profile. The **USRDATA** fields are normally not shown as part of the regular RACF commands. When appropriate controls are set, the **USRDATA** fields that are used for the Command Audit Trail are shown as part of the various RACF list commands, like **LISTUSER**. The data is displayed following the regular command output.

Because the Command Audit Trail data is maintained in the affected profile, no information is collected, and all existing information is deleted if the profile is deleted.

Control of the Command Audit Trail function

You can define =CMDAUD policy profiles to control the Command Audit Trail function.

When you create these policies, you can control whether zSecure Command Verifier collects and retains Command Audit Trail information for all terminal users, and whether the information is shown as part of the regular LIST output. The collected information provides accountability for changes to profiles because it shows who made a certain change, and at what date and time. If you do not define the =CMDAUD policy profiles, then zSecure Command Verifier does not collect the Command Audit Trail information.

The policy profile definition determines whether Command Audit Trail information is collected. For most =CMDAUD policy profiles, the access level is ignored. The only exception is the =CMDAUD.=MAINT policy profile. Two access levels are reserved for authorization of the **C4RCATMN** command that can be used to remove the collected Command Audit Trail data.

Structure of =CMDAUD policy profile

Use these variable details and examples to define =CMDAUD policy profiles that control the Command Audit Trail function.

The basic structure of the =CMDAUD policy profiles includes five separate sections in the following format:

```
C4R.class.=CMDAUD.data-type.profile-identification
```

The *class*, the *data-type*, and the profile itself (*profile-identification*) are used to select which type of Command Audit Trail is collected. The parts of the =CMDAUD policy profiles are described in the following list.

class
This qualifier in the policy profile describes the resource class of the profile as used in or implied by the command.

=CMDAUD
This qualifier of the policy profile must be present exactly as shown. If the best matching generic profile for this policy does not contain this qualifier, zSecure Command Verifier searches for a next best policy profile where the class qualifier is represented by a single generic character (*). For an example, see “Examples” on page 19.

data-type
This part of the =CMDAUD policy profile can have any of the following values:

=SURROGATE
Controls the displaying of the submitting userid in the Command audit trail instead of the actual user.

=SEGMENT
Information about adding, changing, and deleting segments.

Although technically, the MFA data in the USER profile is not kept in a separate segment, modifications to the MFA data are recorded based on the =SEGMENT policy for the Command Audit Trail.

The =SEGMENT policy profile also controls the creation of the "Profile Created" entry.

=ATTR
Information about adding and deleting attributes.

=CONNECT
Information about adding, changing, and deleting user to group connections.

=ACL
Information about use of the **PERMIT** command to manage Access List entries.

=MEMBER
Information about adding and deleting members in a grouping resource class profile.

=MAINT
Controls display and removal of the Command Audit Trail data.

profile-identification
This part of the =CMDAUD policy profile is dependent on the *class* of the target profile. For USER and GROUP profiles, it includes the owner of the profile. For other profiles, it is the resource profile itself. The policy profile for the =SURROGATE data type does not have any profile-identification. The policy applies to all target profiles in the specified *class*.

USER
owner.userid

GROUP
owner.group

resource
resource-profile

Examples

In this first example of an =CMDAUD policy profile, recording in the Command Audit Trail of changes to a segment of the USER profile IBMUSER, owned by the GROUP SYS1, is controlled through policy

```
C4R.USER.=CMDAUD.=SEGMENT.SYS1.IBMUSER
```

It is also possible to define a more generic policy profile. For example, if you want to activate the Command Audit Trail for all profiles in all resource classes, you can define a policy profile Policy Profile A (**PPA**):

```
PPA:  C4R.*.=CMDAUD.*.**
```

In both of these examples, the required =CMDAUD qualifier is present.

When you want to restrict management of FACILITY class profiles to certain administrators, you probably also define additional policy profiles; for example, Policy Profile B (**PPB**):

```
PPB:  C4R.FACILITY.**
```

However, updating the Command Audit Trail for a PERMIT command to FACILITY profile BPX.SUPERUSER is controlled by the following policy:

```
C4R.FACILITY.=CMDAUD.=ACL.BPX.SUPERUSER
```

The best matching generic profile for this policy is **PPB**. Because this policy profile does not contain the required qualifier =CMDAUD, zSecure Command Verifier bypasses this best matching profile, and instead tries to locate the best matching policy profile for

```
C4R.*.=CMDAUD.=ACL.BPX.SUPERUSER
```

In this example, **PPA** is used instead of the regular best matching profile **PPB**.

Access level for the =CMDAUD policy profile

Access to most =CMDAUD policy profiles is not used to control collection of the Command Audit Trail data. Only the profile existence is used to determine whether audit trail data must be collected or not. The access level that is defined for the =CMDAUD.=MAINT profile determines whether the Command Audit Trail is displayed.

If the profile has READ access, Command Audit Trail information is included in the output of the various RACF list commands. Access to this policy profile also controls the use of the **C4RCATMN** command. You can use this command to display or remove the Command Audit Trail information from selected profiles. For details, see “C4RCATMN command” on page 21. For all other =CMDAUD policy profiles, the access level is not used.

If the relevant Policy Profiles are defined, the Command Audit Trail information is collected and retained for all terminal users. For this reason, the Command Audit Trail provides accountability for changes to profiles. Using the collected information, it is possible to determine who made a certain change, and at what date and time. The information is missing only if zSecure Command Verifier is not active or if the policy profiles did not exist at the time the command was run. The nature of an audit trail implies that no terminal users must be exempt. For this reason, the access level is not used to control the collection of the Command Audit Trail.

The following access levels are currently used for the policy profiles, except for the =CMDAUD.=MAINT policy profile.

No Profile Found
No command audit data is collected or retained.

NONE
Command Audit Trail data is collected and retained.

READ
Same as NONE.

UPDATE
Same as NONE.

CONTROL
Same as NONE.

The following access levels are currently used for the =CMDAUD.=MAINT policy profile.

No Profile Found
Command Audit Trail data is not displayed and cannot be maintained by using the **C4RCATMN** command.

NONE
The Audit Trail data is not shown and cannot be maintained through the **C4RCATMN** command.

READ
The Audit Trail data is shown as part of the RACF list command.

UPDATE
The Audit Trail data is shown as part of the RACF list command. It can also be displayed through the **C4RCATMN** command. The **C4RCATMN** command does no scope verification. When the terminal user is authorized for the display function through **C4RCATMN**, the Command Audit Trail of all profiles in the RACF database can be inspected.

CONTROL
The terminal user is also authorized to use the **C4RCATMN** command to remove the Command Audit Trail data.

The following access levels are currently used for the =CMDAUD.=SURROGATE policy profile.

No Profile Found
Command Audit Trail data is not changed and contains the actual user and not the submitting user.

NONE
Command Audit Trail data is not changed and contains the actual user and not the submitting user.

READ
Command Audit Trail data contains the submitting user instead of the actual user.

UPDATE
Same as READ

CONTROL
Same as READ

The profiles for the various types of information are independent. For example, an installation can decide to only record changes to user attributes, and not record any changes to user segments. Or an installation can implement a policy to record changes only to those users that are owned by the group SYS1.

When you use the RACF list commands, the Command Audit Trail information is shown only if the terminal user did not suppress the RACF BASE segment information. If the **NORACF** keyword is used to suppress the RACF BASE segment information, the Command Audit Trail information is also suppressed. To display the Command Audit Trail without the RACF BASE segment, you can use the **C4RCATMN** command. Because the **C4RCATMN** command does no scope verification, UPDATE access to the applicable =CMDAUD.=MAINT policy profile is required.

In most situations, you set the UACC of all Command Audit Trail policy profiles to NONE. Only a few auditors or system administrators must have READ or UPDATE access to the =CMDAUD.=MAINT policy profile. CONTROL access to this policy profile is normally only granted to a few individuals. This level of access can be used to correct errors, or to remove the Command Audit Trail if such information is no longer needed.

Typically, generic =CMDAUD policy profiles are sufficient to control collection and maintenance of the Command Audit Trail.

C4RCATMN command

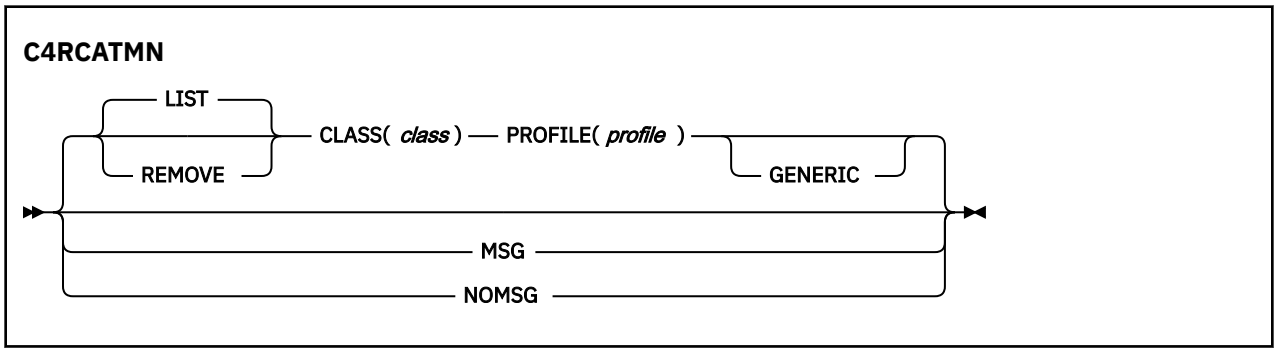
You can use the **C4RCATMN** command to display and remove Command Audit Trail information in various profiles.

Before the Command Audit Trail information is displayed or removed, the applicable =CMDAUD.=MAINT policy profile is checked. If the terminal user has insufficient access, a RACF access violation event is created. The =CMDAUD.=MAINT policy profile has the following form:

```
C4R.class.=CMDAUD.=MAINT.profile-identification
```

For the **C4RCATMN** command, the required access level is UPDATE to display the Command Audit Trail, and CONTROL to remove the Command Audit Trail.

The **C4RCATMN** command has the following syntax:



The keywords and parameters are described here:

LIST
This action is the default action. The Command Audit Trail data for the *profile* in class *class* is shown. If this action is used, you need at least UPDATE access to the =CMDAUD.=MAINT policy profile. If you do not have sufficient access, the Command Audit Trail is not shown.

REMOVE
The Command Audit Trail data for the *profile* in class *class* is removed. If this action is used, you need at least CONTROL access to the =CMDAUD.=MAINT policy profile. If you do not have sufficient access, a RACF violation is recorded, and the Command Audit Trail is not removed.

MSG
This option indicates that the Command Audit Trail information is to be shown as part of the regular RACF list commands. This option is saved across sessions. It is only effective if you fulfill the other requirements for displaying the Command Audit Trail information, like sufficient access to the applicable =CMDAUD.=MAINT policy profile. The initial setting of the MSG / NOMSG setting if you did not issue the **C4RCATMN (NO)MSG** command is MSG.

NOMSG
This option indicates that the Command Audit Trail information is not shown as part of the regular RACF list commands. This option is saved across sessions. If this option is activated, you can display the Command Audit Trail information only through the **C4RCATMN** command. The **C4RCATMN** command requires a higher authorization than the regular RACF list commands to display the information. The initial setting of the MSG / NOMSG setting if you did not issue the **C4RCATMN (NO)MSG** command is MSG.

class
The resource class of the profile that you want to display or remove the Command Audit Trail data from. This keyword and parameter are required when you use the LIST or REMOVE keywords.

profile
The profile that you want to display or from which you want to remove the Command Audit Trail data. The profile must be the exact profile that is stored in the RACF database. No matching of the best fitting generic is done. For data sets, the profile name must include the prefix, and must not be quoted. This keyword and parameter are required when you use the LIST or REMOVE keywords.

GENERIC
This optional keyword indicates that the *profile* is a generic profile, even though it does not contain any generic characters. In general, generic profiles without generic characters occur only in the DATASET class.

Figure 3 on page 22 shows an example of the **C4RCATMN** command. The output of the **C4RCATMN LIST** command is the same as the lines appended at the end of the output of the regular RACF list commands.

```
c4rcatmn list class(user) profile(ibmuser)

Command Audit Trail for USER IBMUSER
Segment:  CICS    Added on 05.241/03:19 by C4RTEST
                  Changed on 05.241/03:20 by C4RTEST
          TSO     Changed on 05.241/03:19 by C4RTEST
Attrib:   PASSWRD Removed on 05.238/14:24 by C4RTEST
          INTERV  Changed on 05.241/04:42 by C4RTEST
          RESTR   Added on 05.238/14:24 by C4RTEST
Connect:          BCSC Added on 05.238/14:24 by IBMUSER
GrpAttr:  ADSP    BCSC Removed on 05.238/14:24 by IBMUSER
```

Figure 3. C4RCATMN LIST command output

The following example shows the output of the **C4RCATMN** command when the Command Audit Trail information is removed.

```
c4rcatmn remove class(gcicstrn) profile(cicsa.spro)
Command Audit data for segments has been removed
Command Audit data for attributes has been removed
Command Audit data for access list unchanged
Command Audit data for members has been removed
```

Figure 4. C4RCATM command output when the Command Trail Audit data is removed

Format of the Command Audit Trail data display

Use this information to understand how to suppress, filter, and interpret Command Audit Trail data.

The example in Figure 3 on page 22 shows the output of the **C4RCATMN** command. This output is the same as the lines appended at the end of the regular RACF list commands. For the RACF list commands, the information is shown if the user has READ access to the =CMDAUD.=MAINT policy profile. If the RACF list command specifies multiple RACF profiles, the Command Audit Trail information for the specified profiles is shown after all RACF information for all profiles. Examples of such list commands are:

```
LISTDSD   DA(dsn1,dsn2)
LISTUSER  (user1,user2)
```

Each Command Audit Trail section is identified by a header line, like:

```
C4R736I Command Audit Trail for USER user1
```

The Command Audit Trail is not included if the RACF list command specifies a pattern or prefix for the profiles to be shown. Examples of such list command are:

```
LISTDSD   PREFIX(user1)
RLIST     FACILITY *
```

If the terminal user has READ access to =CMDAUD.=MAINT policy profile, the Command Audit Trail information is shown. There is no option on the RACF list commands to suppress these additional lines. There are two indirect ways to suppress the Command Audit Trail information:

- Issue the **C4RCATMN** command with the NOMSG keyword. The Command Audit Trail information is no longer shown. It is still possible to show the information by using the **C4RCATMN** command, but it requires a higher authorization than the regular RACF commands need. You can use the **C4RCATMN MSG** command to reactivate showing the Command Audit Trail. The MSG / NOMSG setting is saved across sessions. The initial setting of the MSG / NOMSG setting if you did not issue the **C4RCATMN (NO)MSG** command is MSG.
- Allocate a ddname (=`filename`) with the name C4RNOCAT. This ddname does not need to be allocated to a particular data set, sysout class, or device. The preferred allocation is to DUMMY. The allocation of this ddname is sufficient to suppress display of all Command Audit Trail information as part of the regular RACF list commands. It is still possible to show this information by using the **C4RCATMN** command, although it requires a higher authorization to the =CMDAUD.=MAINT policy profile.

The user can be the submitting user ID if the actual user ID has more than NONE access to the =CMDAUD.=SURROGATE policy profile.

The Command Audit Trail information consists of several sections.

- **The Header**

  Shows the class and profile that is listed.

- **The PROFILE section**

  Contains information about who created the profiles (User, Group, Dataset, General Resource profile). The first line starts with the word `Profile:`, followed by information about when the profile was created and which user ran the command. It also contains the highest non-zero return code from the pre-, RACF, and post-command.

  Collection is controlled by the policy profile

  ```
  C4R.class=CMDAUD.=SEGMENT.profile-identification
  ```

- **The Segments section**

  Contains the information about the last change to non-base segments. The first line starts with the word *Segment:*, followed by an abbreviated name for the segment. The remainder of the line contains information about the type of change, like add, change, delete, when the change was made, and which user ran the command. It also contains the highest non-zero return code from the pre-command, RACF, and post-command. For modifications to existing segments, only the last change is shown.

  Collection is controlled by the policy profile

  ```
  C4R.class=CMDAUD.=SEGMENT.profile-identification
  ```

  A separate block (add, change, delete) is shown for each segment that was modified. The following segments and pseudo-segment are currently supported.

  USER

CICS DCE	LANGUAGE LNOTES	OPERPARM OVM

DFP CSDATA EIM KERB	MFA NDS NETVIEW OMVS	PROXY TSO WORKATTR

GROUP
CSDATA, DFP, OMVS, OVM, TME

DATASET
CSDATA, DFP, TME

General Resource

CFDEF CDTINFO CSDATA DLFDATA EIM ICSF ICTX	IDTPARMS JES KERB MFA MFPOLICY PROXY	SESSION SIGVER SSIGNON STDATA SVFMR TME

- **The Attributes section**

  Contains the attributes and the information about the last change to the attributes. The first line starts with the word *Attrib:*, followed by an abbreviated name for the attribute. The remainder of the line contains information about the type of change such as add or remove, when the change was made, and which user ran the command. It also contains the highest non-zero return code from the pre-, RACF, and post-command. If the profile already has the attribute, a possible *confirmation* command is not shown. The information that is shown reflects the date, time, and ID that changed the profile.

  Collection is controlled by the policy profile

  ```
  C4R.class=CMDAUD.=ATTR.profile-identification
  ```

  A separate block (add, change, remove) is shown for each attribute that was modified. The following attributes are currently supported.

  USER

ADSP AUDITOR CATEGORY CLAUTH DFLTGRP EXPIRED GRPACC INSTDATA INTERVAL	MODEL NAME OIDCARD OPERATIONS OWNER PASSWORD PHRASE RESTRICTED	RESUME REVOKE ROAUDIT SECLEVEL SECLABEL SPECIAL UAUDIT WHEN

  GROUP
  INSTDATA, MODEL, OWNER, SUPGRP, TERMUACC, UNIVERSAL

  DATASET

ACL AUDIT CATEGORY	GAUDIT INSTDATA LEVEL	SECLEVEL SECLABEL UACC WARNING

ERASE FROM	NOTIFY OWNER	

General Resource

ACL APPLDATA AUDIT CATEGORY FROM GAUDIT	INSTDATA LEVEL NOTIFY OWNER SECLEVEL SECLABEL	SINGLEDS TVTOC TIMEZONE UACC WARNING WHEN

- **The Connects section**

  Contains the Groups, the Authorizations, and the UACC together with information about the last change to the connect.

  Collection is controlled by the policy profile

  ```
  C4R.class=CMDAUD.=CONNECT.profile-identification
  ```

  The Connects section is only present for USER profiles. It is not included for GROUP profiles. The first line in this section starts with the word *Connect:*. Each line shows the GROUPNAME, followed by the UACC, the GROUP-Authority, the date and time when the change was made, which user ID executed the command, and the highest non-zero return code from the pre-, RACF and post-command. If both the UACC and the GROUP-Authority have their default value (that is, UACC=NONE and AUTH=USE) their values are not explicitly shown. This makes it easier to spot non-default settings. For more information about the UACC and AUTH settings, see the *RACF Security Administrator 's Guide* and the *RACF Command Language Reference*.

  Because of size limitations, only the last 64 changes to the connect groups are shown.

- **The Group-Attributes section**

  This section immediately follows the Connect section and it contains information about the last change to any GROUP-attribute. The first line starts with the word *GrpAttr:*, followed by an abbreviated name for the attribute.

  Collection is controlled by policy profile

  ```
  C4R.class=CMDAUD.=CONNECT.profile-identification
  ```

  The Group-Attributes section is only present for User profiles. It is not included in Group profiles. The lines show the attribute, followed by the GROUP name, when the change was made, and which user ran the command. It also shows the highest non-zero return code from the pre-, RACF, and post-command. There can be multiple lines for the same attribute, if the attribute was added and removed. The lines for each attribute are in date/time sequence, so the last line reflects the status.

  Because of size limitations, only the last 64 changes to the connect groups are shown. The following attributes are currently supported.

  ```
  ADSP, SPECIAL, OPERATIONS, REVOKE, GRPACC, AUDITOR, RESUME
  ```

- **The Access List section**

  Contains access list entries and the information about the last change to the access list entries. The lines show the access level that was granted, followed by when the change was made, and which user ran the command. It also shows the highest non-zero return code from the pre-, RACF, and post-command. There is only one line for each user, group, or profile. The last instance of granting or removing access is shown. If a user was removed from the access list, the value Removed is shown. The

special ID **ALL** is used to reflect the use of the RESET keyword on the PERMIT command. Because of size limitations, only the last 64 changes to the access list are collected.

Collection is controlled by policy profile

```
C4R.class=CMDAUD.=ACL.profile-identification
```

- **The Member section**

  Contains members that are part of a grouping class profile. The lines reflect adding or removing entries to and from the member list of grouping class profiles. Each line has one member, followed by when the change was made, and which user ran the command. It also shows the highest non-zero return code from the pre-, RACF, and post-command. There is only one line for each member, reflecting the last action. Because of size limitations, only the last 64 changes to the member list are shown. Also, only the first 128 bytes of the member name are collected and thus included in the display.

  Collection is controlled by policy profile

  ```
  C4R.class=CMDAUD.=MEMBER.profile-identification
  ```

An example for a user profile is shown here:

```
Command Audit Trail for USER C4RUSER
Profile:          Created on 19.238/14:24 by C4RTEST
Segment:  CICS    Added on 19.241/03:19 by C4RTEST
                  Changed on 19.241/03:20 by C4RTEST
          TSO     Changed on 19.241/03:19 by C4RTEST
Attrib:   PASSWRD Removed on 19.238/14:24 by C4RTEST
          INTERV  Changed on 19.241/04:42 by C4RTEST
          RESTR   Added on 19.238/14:24 by C4RTEST
          WHEN    Added on 19.238/14:24 by C4RTEST
Clauth:           USER Added on 19.241/10:04 by C4RTEST
                  TCICSTRN Removed on 19.241/10:05 by C4RTEST
Connect:          C4RGRP1 Added on 19.238/14:24 by C4RTEST
GrpAttr:  ADSP    C4RGRP1 Removed on 19.238/14:24 by C4RTEST
```

Figure 5. Command Audit Trail data for a user profile

An example for a data set profile is shown in the following figure. In this example, a DFP segment was added, the profile was placed in WARNING mode, and several access list entries were changed or removed. On 14 September 2019 (19.257) the entire access list was reset by IBMUSER by using the **PERMIT RESET** command.

```
Command Audit Trail for DATASET C4RUSER.**
Profile:          Created on 19.234/09:39 by C4RTEST
Segment:  DFP     Added on 19.245/05:21 by C4RTEST
Attrib:   FROM    DATASET C4RUSER.TEST.** on 19.234/09:39 by C4RTEST
          WARNING Added on 19.245/05:20 by C4RTEST
          AUDIT   SUCCESS Removed on 19.245/08:41 by C4RTEST
                  FAILURES Changed on 19.246/01:30 by C4RTEST
          GAUDIT  SUCCESS Changed on 19.245/09:38 by C4RTEST
                  FAILURES Changed on 19.245/09:38 by C4RTEST
Access:           DATASET C4RUSER.TEST.** access Added on 19.234/09:39 by C4RTEST
                  C4RGRP1 access READ on 19.234/09:39 by C4RTEST
                  C4RGRP2 access READ on 19.234/09:39 by C4RTEST
                  C4RTEST access READ on 19.234/09:39 by C4RTEST
                  SYS1 access READ on 19.234/09:39 by C4RTEST
                  IBMUSER access READ on 19.234/09:39 by C4RTEST
                  * access UPD on 19.234/09:39 by C4RTEST
                  CRMBGUS access Removed on 19.234/09:39 by C4RTEST
                  **ALL** access Removed on 19.257/15:06 by C4RTEST
```

Figure 6. Command Audit Trail data for a data set profile

The following example shows the Command Audit Trail information for adding and removing members from a profile in a grouping resource class.

```
Command Audit Trail for GCICSTRN CICSA.SPRO
Member:           CICSA.CEDA Added on 19.249/14:21 by C4RTEST
                  CICSA.CEMT Removed on 19.249/14:21 by C4RTEST
```

Figure 7. Command Audit Trail data for managing members in a profile in a grouping resource class

The information about a segment or attribute is presented in date/time sequence. The last line that is shown for a particular segment or attribute is the last recorded action. If an attribute was granted and later removed, the first line shows who granted the attribute and the last line shows who removed the attribute.

For Access List entries and Member Lists, only the last 64 changes are retained. This restriction is mainly for profile size and performance reasons. Only the last action for each ID or member is recorded.

RRSF overview

Use this information to understand how to manage Command Audit Trail data in an RRSF environment.

When the zSecure Command Verifier Command Audit Trail function is used in an RRSF environment, the Command Audit Trail information is maintained on each system in the RRSF environment individually. It means that the policy profiles as defined on the target system control if and how the Command Audit Trail is maintained. Flow of the data across RRSF is based on the RRSFDATA profiles and operational setting for Command propagation. If the command is propagated on the target system, zSecure Command Verifier adds Command Audit Trail data if applicable. The individual entries in the Command Audit Trail are not propagated. The Command Audit Trail on each system in the RRSF environment is maintained independently of the Command Audit Trails on the other systems.

If zSecure Command Verifier is not installed or not active on a system, the Command Audit Trail is not maintained. The same applies when the required policy profiles are absent.

As a result, Command Audit Trail data can be slightly out of synch. For instance, if the SPECIAL attribute on the RRSF node SYSA is removed from user IBMUSER on 05.283/14:13 and the RRSF command propagation to the RRSF node SYSB is completed a few minutes later, the Command Audit Trail data on SYSB shows 05.283/14:15. In every situation, the Command Audit Trail shows the date and time that the change became effective on the current system.

Storage space planning

Use these guidelines to plan the allocation of RACF storage for the Command Audit Trail function.

The Command Audit Trail function requires additional space in the RACF database. The initial amount of space that is required depends on the rate at which changes are made to the profiles. After a certain number of RACF commands, the space requirements stabilize. This stabilization probably occurs at a significant lower requirement than the possible maximum space requirement. For instance, the maximum space that is needed for the Command Audit Trail of a user profile is used if commands are issued to give and remove every possible attribute to a user ID. In that case, you need space for recording 2 events (give, and remove) for 20 attributes. This space amounts to 20 * 58 bytes = 1160 bytes.

Typically, only one or two attributes are managed for each user ID. In this case, the estimated storage would be 60 bytes per user ID.

For data sets and general resources, the amount of storage that is needed depends mainly on the access list entries. Each access list entry requires 34 bytes. A maximum of 64 access list entries are recorded, which can take up to 2176 bytes. For most profiles, the total access list activity probably stabilizes at 20 entries, which can require 680 bytes.

The following table shows the required space for each type of information. It also shows estimates of how much storage would be needed on average for each profile. The actual space that is required in your RACF database depends strongly on the RACF command activity in your environment.

Table 6. Storage estimates per audit data type

Data-Type	Class	Min	Maximum	Maximum # Entries	Total	Estimate
Segment	User	33	80	15	1200	56
Segment	Group	33	80	4	320	56
Segment	Dataset	33	80	2	160	0
Segment	General	33	80	10	800	56
Attr	User	33	58	20	1060	64
Attr	Group	33	58	6	348	64
Attr	Dataset	33	58	10	580	64
Attr	General	33	58	13	754	64
Connect	User	43	2249	1	2057	168
GrpAttr	User	42	2185	7	15295	195
ACL	Dataset	42	2185	1	2185	319
ACL	General	42	2185	1	2185	319
Member	Transaction Groups	47	2505	1	2505	2505
Member	General	47	2505	1	2505	369

Internal format of USRDATA entries

The information in this section is only relevant for people who want to inspect the USRDATA entries as maintained by zSecure Command Verifier manually, or who must diagnose problems in these fields.

In each profile, relevant information is kept in multiple USRDATA fields. The USRDATA is accessed as a name-value pair. The USRNAME field describes the information kept in the corresponding USRDATA field. The following USRNAME values are used:

```
$C4RS seg    profile segment seg
$C4RA att    profile attribute att
$C4RAFRM     from profile class and name
$C4RAAUD     audit settings
$C4RAGAU     globalaudit settings
$C4RCLAU     class authority
$C4RCONN     connect groups
$C4RC att    connect group attribute att
$C4RPACL     access list
$C4RRMEM     member list
```

The corresponding data fields contain the information in EBCDIC format. The information in these data fields is specific for the profile class. For instance, for USERs, the attribute might be SPECIAL (abbreviated to SPC), while for GROUPS, the TERMUACC attribute might be present (represented by $C4RATRM).

The data fields for each segment or attribute are treated as a block of data that contains multiple statistics. The different events (add, change, remove) for that particular attribute or segment are kept in one statistics block. For the access-list-related field, the last 64 `userid` values are kept together in one block. The format of the data is:

$C4RS*seg*
: This field is used to retain information about one segment. The field has four subfields that are separated from each other by a comma. Information about adding (A), changing (C), and deletion (D) of the segment is separated by a semicolon.

Value BAS in $C4RSBAS usrdata is used to represent the BASE segment of the profile, which is used to record information about the creation of the profile.

The following subfields are present:

Action
Character that indicates if information is about adding (A), changing (C), or deleting (D) the segment.

DATETIME
10 characters when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that handled the segment.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry for a TSO segment might be:

```
A,09220/0801,CRMBTST,00;C,09221/0815,IBMUSER,00
```

$C4RA*att*
This field is used to retain information about attributes that were added or removed from the profile. The field has four subfields that are separated from each other by a comma. Information about different actions is separated by a semicolon. The following subfields are present:

Action
Character that indicates if information is about adding (A), changing (C), or deleting (D) the attribute.

DATETIME
10 characters when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last handled the attribute

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry for the Special attribute might be:

```
A,09181/0917,IBMUSER,00;D,09181/0920,IBMUSER,00
```

$C4RAFRM
This field is used to retain information about the FROM class and profile of dataset and general resource profiles. The field has five subfields that are separated from each other by a comma. The following subfields are present:

Class
Profile class name that was used to create a new profile. This field represents the FCLASS field value of the ADDSD/RDEFINE command.

Profile-Name
Profile that was used as a model profile to create a new profile.

Action
A character that indicates that the profile is created by using the FROM(F) keyword.

DATETIME
10 characters that show when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character ID of the userid that created the profile by using an explicit model profile.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
XFACILIT,C4RUSER.TEST.**,F,20140/1304,C4RUSER,00
```

$C4RAAUD

This field is used to retain information about the audit setting of dataset and general resource profiles. The field has five subfields that are separated from each other by a comma. Information for success and failure auditing is kept in two entries, separated by a semicolon. The following subfields are present:

SUCCESS or FAILURES
The audit setting for successful or failed access to the resource.

Action
Character that indicates if auditing was added (A), changed (M), or removed (D).

DATETIME
10 characters that show when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this audit setting.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
SUCCESS,D,19214/0841,C4RTEST,00
```

$C4RAGAU

This field is used to retain information about the globalaudit setting of dataset and general resource profiles. The field has five subfields that are separated from each other by a comma. Information for success and failure auditing is kept in two entries, separated by a semicolon. The following subfields are present:

SUCCESS or FAILURES
The globalaudit setting for successful or failed access to the resource.

Action
Character that indicates if auditing was added (A), changed (M), or removed (D).

DATETIME
10 characters that show when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this audit setting.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
FAILURES,M,19214/0938,IBMUSER,00
```

$C4RCLAU

This field is used to retain information about the class authorization (CLAUTH) of users. Only the last 64 changes are retained in the profile. The field has five subfields that are separated from each other by a comma. Information for different resource classes is separated by a semicolon.The following subfields are present:

Class
The class for which the CLAUTH was added or removed.

UN or DD
Two placeholder characters that are used to indicate if the CLAUTH was added or removed.

DATETIME
10 characters that show when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this connect.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
USER,UN,19211/1004,IBMUSER,00
```

$C4RCONN
This field is used to retain information about the connection of users to groups. It is kept in the user profile. Only the last 64 changes are retained in the profile. The field has five subfields that are separated from each other by a comma. Information about different connect groups is separated by a semicolon. The following subfields are present:

Group
The group to which the user is connected.

Auth and UACC
These characters represent the Authority in the group and the UACC for new data sets when the user is logged on using this GROUP as the current connect group. The Authority can be U (use), R (create), C (connect), or J (join). The UACC can be N (none), E (execute), R (read), U (update), C (control), or A (alter).

DATETIME
10 characters that show when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this connect.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
SYS1,JR,09245/0545,C4RTEST,08
```

$C4RC*att*
This field is used to retain information about the group attributes of users. It is kept in the user profile. Only the last 64 changes are retained in the profile. The field has five subfields that are separated from each other by a comma. Information about different connect groups is separated by a semicolon. The following subfields are present:

Group
The group to which this attribute applies.

Action
Character that indicates if information is about adding (A) or deleting (D) the attribute.

DATETIME
10 characters when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this connect.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
SYS1,A,09245/0550,C4RTEST,00;SYS1,D,09245/0555,C4RTEST,00
```

$C4RPACL
This field is used to retain information about the access list of data sets and general resource profiles. Only the last 64 changes are retained in the profile. The field has five subfields that are separated from each other by a comma. Information about different users/groups in the access list is separated by a semicolon. The following subfields are present:

User ID or Class/Profile
The access list entry, which can be a RACF user ID, group ID, an asterisk, class/profile, or the special value &RACUID.

Access level
Character for the access level granted: N(one), E(xecute), R(ead), U(pdate), C(opy), A(lter), D(elete), or F(rom).

The F character indicates that the access list of the profile is copied from the profile that is mentioned in the FROM(F) keyword.

DATETIME
10 characters when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last changed this access list entry

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
IBMUSER,R,09245/0545,C4RTEST,00;DATASET/C4RUSER.TEST.**,F,20156/0036,C4RTEST,00
```

$C4RRMEM
This field is used to retain information about the member list for profiles in a grouping resource class. Only the last 64 changes are retained in the profile. The field has four subfields that are separated from each other by a comma. Information about different members is separated by a semicolon. The following subfields are present:

Member
The member name. This name usually has the format of a profile in the corresponding member (non-grouping) class.

Action
Character that indicates if information is about adding (A) or deleting (D) the member.

DATETIME
10 characters when the command was issued. The format is *yyddd/hhmm*.

User ID
Maximum eight-character userid that last added or removed this member.

RC
Two-digit maximum return code of the RACF command or the pre-command and post-command.

An example entry might be:

```
'SYS1.LINKLIB'//NOPADCHK,A,09249/1419,C4RTEST,00
```

or

```
TEST.CEMT,A,09249/1421,C4RTEST,00;TEST.CEDA,A,09249/1421,C4RTEST,00
```

Using the zSecure Admin Command Logger

Aside from the Command Audit Trail (CAT) that the previous section describes, and the command auditing functions that the next section describes, zSecure Command Verifier also provides the option to write the complete command text to the zSecure Admin Command Logger (CKXLOG).

The CAT has as advantage that it records the essential part of a profile change in the profile itself. There is no need to scan a large amount of data to quickly find who changed the profile and when that was done. The SMF support, as the next section describes, allows a separate record of the command as the user enters it, and the command as it is modified and approved by the zSecure Command Verifier policy profiles. The zSecure Admin Command Logger function is a mix between these two options. The complete command, resulting from the application of the policy profiles, is written to a log stream. Depending on

the record retention period, the log stream might be large, and you must scan many records to locate the relevant ones. The log stream contains all the commands, while the CAT only contains information about the latest change. The CKXLOG log stream is smaller than the full SMF files.

The zSecure Admin Command Logger logs commands depending on the component that passes the command and the user's access to this component. Switch profiles of the form CKX.CKXLOG.LOG.*component-name* control how commands are logged. The *component-name* for Command Verifier is C4RMAIN. For additional information about these switch profiles, see *zSecure CARLa-Driven Components Installation and Deployment Guide*. The zSecure Admin Command Logger uses these switch profiles independently of the Command Verifier policy profiles that are described in this section. To successfully log a RACF command, the terminal user must have the appropriate access to the Command Verifier =CKXLOG policy profile, and to the CKXLOG switch profile.

The record in the CKXLOG log stream might contain information about the origin of the RACF command: For commands routed through RRSF, the terminal user's node and user IDs are included. If the command was issued from a batch job, the submitting node and user IDs are included. The record also contains the RACF return code from the command. Reporting about the RACF commands can be done either by browsing records offloaded from the log stream, or using zSecure Admin. zSecure Admin provides the CKXLOG newlist and the CR menu option in the ISPF interface. For more information about installing the required CKXLOG started task, see *zSecure CARLa-Driven Components Installation and Deployment Guide*. For more information about reporting through zSecure Admin see *zSecure Admin and Audit for RACF User Reference Manual*.

zSecure Command Verifier logs the RACF commands through CKXLOG after the commands are executed. For example, if RACF already rejected a command because of invalid syntax, the command is not passed to zSecure Command Verifier and the command is not logged. This in contrast to the zSecure Admin interface, which logs the commands before they are executed.

If the CKXLOGID command was issued before the RACF command, the record in the CKXLOG log stream is annotated with the specified ticket identification and description. If the CKXLOGID command was not issued, or if the ticket information expired, a warning message might be issued to inform the terminal user that no ticket information was added. This message can be suppressed by allowing UPDATE access to the policy profile. For more information about the CKXLOGID command see *zSecure Admin and Audit for RACF User Reference Manual*.

Recording RACF commands to the zSecure Admin Command Logger (CKXLOG) is controlled through profiles of the following format:

```
C4R.command.=CKXLOG
```

This profile specifies that the RACF command, as optionally modified and approved by the zSecure Command Verifier policy profiles, is recorded in the CKXLOG log stream. If the command is rejected, zSecure Command Verifier does not record the command.

The *command* is the non-abbreviated RACF command as the terminal user issues it. The qualifier =CKXLOG in the policy profile cannot be covered by generic characters. It must be present in the exact form as shown.

It is expected that many installations define a generic policy profile with UACC(READ) to cover all RACF commands and separate policy profiles for the RACF LIST commands with UACC(NONE):

Profile	**Universal access**
C4R.*.=CKXLOG	READ
C4R.LIST*.=CKXLOG	NONE
C4R.RLIST.=CKXLOG	NONE

The following access rules apply:

No Profile Found
This control is not implemented. The command is not logged through CKXLOG.

NONE
This control is not active for the terminal user. The command is not logged through CKXLOG.

READ
The command as optionally modified and approved by zSecure Command Verifier is logged through CKXLOG. If no ticket information is present, or if the CKXLOG server is not active, a warning message is issued.

UPDATE
The command as optionally modified and approved by zSecure Command Verifier is logged through CKXLOG. Warning messages about missing ticket information or an inactive CKXLOG server are suppressed.

CONTROL
Same as UPDATE.

Policy profiles for command auditing

To enable the installation auditor to activate detail auditing of commands, zSecure Command Verifier contains several extra RACROUTE REQUEST=AUTH requests. These requests result in the writing of SMF records for successful access to special audit-only policy profiles. Only successful access is recorded and failed access to these policy profiles is not used. The access to the audit-only policy profiles is not used to control use of the RACF commands or keywords.

The following types of audit-only profiles are supported:

- Policy profiles used to record the entire RACF command in the LOGSTR in the SMF record.
- Policy profiles used to record the use of group-level attributes like group-special.

These types are described in the following sections.

Policy profiles for auditing entire commands

The audit-only policy profiles for entire commands can be defined in the XFACILIT resource class. The profiles contain the type of data that you want audited and the RACF command. Auditing is being done through an Audit-only RACROUTE REQUEST=AUTH for a specific policy profile in the XFACILIT class. After zSecure Command Verifier has parsed the RACF command, the complete command is recorded as extra data in the SMF records.

The following profiles are used:

- A profile for the unmodified command as issued by the terminal user
- A profile for the command as passed to RACF for execution
- A profile that is used to record a possible error message

You can use **`AUDIT(SUCCESS)`** in combination with the UACC and Access List to control which users must be audited under which circumstances. The access to the profile is not used for policy decisions by zSecure Command Verifier in any way. The preferred setting for full auditing of all commands is **`UACC(READ)`** and **`AUDIT(ALL(READ))`**.

- `C4R.PREAUD.`*COMMAND* specifies whether the command string entered by the terminal user must be audited. The complete command string is available in the LOGSTRING of the generated SMF record.
- `C4R.PSTAUD.`*COMMAND* specifies whether the command string after zSecure Command Verifier processing must be audited. The complete command string is available in the LOGSTRING of the generated SMF record.
- `C4R.ERRMSG.`*COMMAND* records the error or warning message that is issued by zSecure Command Verifier. It can be found in the LOGSTRING of the SMF record that is generated for this profile.

Only successful access to these profiles is recorded. By selecting the access list, you can control which commands, issued by which user, are recorded. Only the users who have access are traced through SMF. Users who do not have access are not traced.

The definition of these command auditing profiles and their access list (or UACC) has relevance only for the auditing of these commands. These profiles are not used to control execution of the command or of any keywords.

In summary, for the auditor to activate auditing, the following actions must be completed.

- Define a profile in the RACF XFACILIT resource class; for example, C4R.PREAUD.ADDUSER
- Set UACC and Access List (ACL) at READ or higher for those users you want to audit.
- Set AUDIT(SUCCESS(READ)) if you want to audit command usage.
- Setting AUDIT(FAIL(....)) is not effective because zSecure Command Verifier does not support failed access auditing for these profiles.

The XFACILIT profiles that are being used are formed like C4R.PREAUD.*command*. The qualifiers might be set as follows:

The first qualifier:

C4R
Fixed prefix to indicate that these profiles are related to the zSecure Command Verifier.

The second qualifier:

PREAUD
For the command as entered by the terminal user.

PSTAUD
For the command after it is modified and approved by the policy routines.

ERRMSG
The error message if the command is rejected by the policy routines.

The third qualifier:

command
Variable part to indicate the command that is being audited. It is the full unabbreviated RACF command as entered by the terminal-user.

It is possible to use generic profiles. If the installation auditors want to audit all commands, they define a profile C4R.PREAUD.* with **UACC(READ)** and **AUDIT(SUCCESS(READ))**. It generates standard RACF audit records for access to these profiles. The audit LOGSTR is the command as issued by the user or modified as specified through policies. Unfortunately, if the command is longer than 255 characters, only the first 255 are shown.

An example of a profile that is being used for audit specification is shown. It indicates that the **ADDUSER** command must be audited for all users before inspection and possible modification by the policy routines. When ADDUSER is issued by user IBMUSER, it must not be audited at all.

```
C4R.PREAUD.ADDUSER   UACC(READ) AUDIT(SUCCESS(READ))
                     IBMUSER(NONE)
```

Sensitive fields are not present in the audit trail. For instance, if a system administrator issues a command to reset the password of a user ID, the new password value is not present in the audit string. For other sensitive fields, like session keys and PassTicket encryption keys, the same suppression of sensitive information occurs.

Policy profiles for the use of group-special

The audit-only policy profiles for the use of (group-)special can also be defined in the XFACILIT resource class. The profiles have the special qualifier USESCOPE to indicate that they are intended to record the use of a certain RACF administrative scope.

You can use AUDIT(SUCCESS) in combination with the UACC and Access List to control which users must be audited under which circumstances. The access to the profile is not used for policy decisions by zSecure Command Verifier in any way. The preferred setting for full auditing of all commands is UACC(READ) and AUDIT(ALL(READ)). The following profile is used:

- C4R.USESCOPE.*group*

  The qualifier *group* represents the lowest group in the RACF group-tree that grants group-special authority over the target profile in the command. If the terminal user has system-special, the fixed value =SYSTEM is used.

  All qualifiers of these policy profiles can be represented by generic characters.

Only successful access to these profiles is recorded. By selecting the access list, you can control which commands, issued by which user, are recorded. Only the users who have access are traced through SMF. Users who do not have access are not traced.

Policies where the use of an administrative scope is recorded through these USESCOPE profiles are indicated as such in the remainder of this documentation.

Example zSecure audit reports

Using zSecure Audit, it is possible to generate reports about the RACF commands that are entered before and after the zSecure Command Verifier policy routines have evaluated and possibly modified the RACF commands.

You might generate such a report from the zSecure Audit interactive interface. Select **EV.R**, specify resource class XFACILIT and resource C4R.**. The following screen is an example of the input you can use.

```
   Menu         Options       Info    Commands     Setup
 ------------------------------------------------------------------------------
                  zSecure Suite - Events - Resource Selection
 Command ===> _________________________________________________   _ start panel

 Show records that fit all of the following criteria:
 Resource  . . . . . C4R.**_____________________________________
 Class . . . . . . . XFACILIT     (class or EGN mask)
 Profile/rule/permit ___________________________________________________
 System  . . . . . . ____         (system name or EGN mask)

 Advanced selection criteria
 _  Date and time         _  Further resource selection

 Output/run options
 /  Include detail          _  Summarize               _  Specify scope
 _  Output in print format     Customize title            Send as e-mail
      Run in background        Sort
differently
```

Figure 8. Input for generating a zSecure Command Verifier audit report

It is also possible to create a zSecure Audit custom report for zSecure Command Verifier. The following example, which is also present in member C4RCNA00 in SC4RSMP, can be used as a custom display when you use zSecure Audit version 2.5.0. Most of the commands are related to the information displayed on the reports. The following line is the selection criteria for the RACF commands before zSecure Command Verifier policy processing.

```
S CLASS=(XFACILIT) PROFILE=(C4R.PREAUD.**)
```

The following line is the selection criteria for the RACF commands after zSecure Command Verifier policy processing.

```
S CLASS=(XFACILIT) PROFILE=(C4R.PSTAUD.**, C4R.ERRMSG.**)
```

The remaining zSecure Audit statements provide detailed information for the layout of the report. An example is the definition of a variable as a substring of the XFACILIT profile. This substring is the RACF

command that is being issued by the terminal-user. The example shown results in a combined report of all RACF commands before and after zSecure Command Verifier policy processing. If you remove the MERGELIST/ENDMERGE statements, you obtain three separate reports.

An example of the output from the combined report is shown. The terminal-user was not authorized to specify the OPERATIONS keyword. It is removed from the RACF command during policy profile processing.

```
1S M F   R E C O R D   L I S T I N G    3May07 01:45 to 13May07 22:36
 RACF Commands processed by Command Verifier

 Date      Time               Resource

 08Dec2001 23:49 Before PIER   ALTUSER
 System ID                     SYS1 Fri 11 May 2007 23:49
 RACF userid/ACF2 logonid      BCSCGB2
 User name                     GUUS SECONDARY ID
 SAF profile key               C4R.PREAUD.*
 SAF resource name             C4R.PREAUD.ALTUSER
 RACF Command                  ALTUSER  BCSCGB3 OPER

 08Dec2001 23:49 After  PIER   ALTUSER
 System ID                     SYS1 Fri 11 May 2007 23:49
 RACF userid/ACF2 logonid      BCSCGB2
 User name                     GUUS SECONDARY ID
 SAF profile key               C4R.PSTAUD.*
 SAF resource name             C4R.PSTAUD.ALTUSER
 RACF Command                  ALTUSER  BCSCGB3
```

Figure 9. zSecure Audit report: RACF commands before and after zSecure Command Verifier policy processing

Regular access recording through SMF

Aside from auditing the entire command that is provided by the profiles that are shown, zSecure Command Verifier also audits individual keywords. zSecure Command Verifier creates successful and failed access records for the profile that was used in the decision process.

For example, if the profile C4R.USER.ID.CRMB* allows the definition of the new user CRMBTST, a successful access event is recorded against this profile. If a policy profile denied setting a field to the specified value, a violation event is recorded against the policy profile.

The creation of these types of SMF records is controlled by the standard RACF audit settings for the profile. So, if you use the RACF default, only failed access is audited. However, you might decide to change it into AUDIT(ALL(READ)). That way, both successful and failed access attempts are recorded.

In general, zSecure Command Verifier creates an SMF event only if the policy profile was used in the decision process. For example, there are several policy profiles for the naming convention of a new user ID. If a new user ID is rejected by one policy profile, but accepted by another, only the one that allowed the name of the new user ID is recorded through SMF. The other policy profile, that might or might not have allowed creation of the user ID, is not recorded through SMF.

An example can help understand this process. Suppose that here are two user ID naming convention policy profiles. The first one states that a new user ID must have the same three starting characters as the terminal user. The second one states that if the first 3 characters are C4R, the new user ID is allowed.

```
C4R.USER.ID.&RACUID(3)        UACC(UPDATE)
C4R.USER.ID.C4R*              UACC(UPDATE)
```

Now suppose that user IBMUSER attempts to define a new `userid`:

```
ADDUSER C4RTEST DFLTGRP(C4R) OWNER(C4R)
```

The first policy profile does not apply, since the terminal user (IBMUSER) does not match the target user (C4RTEST). The second profile does apply because the target USERID starts with C4R. In this case, zSecure Command Verifier records successful access to the second profile as follows:

```
Resource:      C4R.USER.ID.C4RTEST
Profile:       C4R.USER.ID.C4R*
Access:        UPDATE
User           IBMUSER
```

The resource name that is used for the access verification, and thus the SMF record, usually does contain the value for the field as specified by the terminal user. For instance, when the **PERMIT** command to add IBMUSER to an Access List as issued by C4RTEST is allowed,

```
PERMIT 'SYS1.PARMLIB' ID(IBMUSER) AC(UPDATE)
```

a successful access event is created, such as

```
Resource:      C4R.DATASET.ACL.IBMUSER.UPDATE.SYS1.PARMLIB
Profile:       C4R.*.ACL.*.UPDATE.**
Access:        UPDATE
User:          C4RTEST
```

Access to the Policy Profiles can be reported by using any standard SMF reporting tool, like IBM Security zSecure Audit, or IRRADU00.

Chapter 5. Policy profiles

In zSecure Command Verifier, you can define the installation policies using policy profiles in the XFACILIT resource class.

zSecure Command Verifier internally issues a RACROUTE REQUEST=LIST. A resource class that is RACLISTed requires a refresh before any changes to profiles become effective. You must always issue a **SETROPTS RACLIST**(*class*) **REFRESH** command after you complete changes to the zSecure Command Verifier policy profiles.

If the zSecure Command Verifier resource class is shown as GLOBAL RACLIST ONLY in the SETROPTS output, some releases of RACF do not warn the administrator about the requirement to issue a **SETROPTS RACLIST**(*class*) **REFRESH**. Although not necessary for zSecure Command Verifier performance, you can still issue a **SETROPTS RACLIST**(*class*) command to load all relevant zSecure Command Verifier policy profiles into a dataspace. If you did so, most RACF commands issue the warning message that profiles must be refreshed to be effective.

Attention: Do not define a top generic profile like C4R.** or **. If you define a generic profile, either of the following situations occurs, depending on the specified access:

- All RACF commands for which no specific policy profiles were defined are rejected.
- All zSecure Command Verifier controls are bypassed.

Policy profile syntax

The policy profiles used by zSecure Command Verifier must be defined in the XFACILIT resource class. Alternatively, you can specify another RACF resource class for use in zSecure Command Verifier.

In general, the profiles that are used for policy specification have four qualifiers. The first qualifier is always C4R to indicate that these profiles are for zSecure Command Verifier. The second qualifier indicates the type of profile to which this control applies. Examples of the second qualifier are USER, GROUP, DATASET, and TCICSTRN. The third qualifier is used as an indicator of the function or field that is controlled. Examples are ID, OWNER, NOTIFY, ATTR, and UACC. The fourth qualifier can specify a value for the particular function or field. Examples are READ, JOIN, and *groupname*. Some types of policy profiles also support more qualifiers to specify the profile to which this policy applies. The additional qualifiers are primarily intended to allow for exceptions to the rules. Two example policy rules that fit the general pattern are shown in the following list.

- C4R.DATASET.UACC.READ.SYS1.**

  This profile controls the authority to set the UACC for data sets that match the pattern SYS1.**. The only UACC value that is explicitly controlled by this policy profile is READ.

- C4R.USER.DFLTGRP.SYS1.**

  This profile controls the authority to select the group SYS1 as DFLTGRP. The ** at the end of the policy profile indicates that it applies to all user IDs.

Additionally, special values for certain qualifiers are implemented in these profiles. Use /SCOPE to refer to target profiles like users, groups, or data sets that are *not in scope*. The /SCOPE policy profiles control the authorizations to handle the *not in scope* profiles. Other special qualifiers are indicated by an = sign. These qualifiers are used to describe *equal to* type of policies. Examples of these types of special qualifiers are shown in the following list.

- C4R.USER.PASSWORD.=DFLTGRP

  The profile controls the authority to set the password equal to the default group DFLTGRP of the user.

- C4R.USER.=OWNER.IBM*

  This profile specifies that the owner of a user ID, if it matches the pattern IBM*, must be equal to a certain value. The value is specified in the **APPLDATA** field of the policy profile.

If a policy profile contains a special qualifier that starts with a special character, like a slash "/", equals "=", or similar, that entire qualifier must be present in the policy profile. Generic characters cannot be used to represent this qualifier. Other qualifiers in the policy profile can be covered by generics. For example, the authority to set the password equal to the default group of the user is controlled by the policy:

```
C4R.USER.PASSWORD.=DFLTGRP
```

This policy can be covered by the following policy profiles:

```
C4R.USER.*.=DFLTGRP
C4R.**.=DFLTGRP
C4R.*.PASS*.=DFLTGRP
```

The policy is not covered by any of the following profiles:

```
C4R.**
C4R.USER.PASSWORD.*
C4R.USER.*.=DFLT*
```

There are some exceptions to this rule. These exceptions are mentioned in the detailed description of the policy profiles.

The policy profiles that are used in zSecure Command Verifier fall into two different categories. The first one is the general one used throughout the product. It describes the *result* of a command, and not the command itself. The policy profile has qualifiers that describe the target profile and field, like **DATASET.ACL**, and do not contain any reference to the actual command used to change the ACL, which is the **PERMIT** command. The second type of profile is the one focused around a particular command or command keyword. These policy profiles are, for instance, used to temporarily grant system-special during the execution of the **LISTUSER** command. They contain the actual command, like **ALTUSER**, as a qualifier in the policy profile.

The field-value policy profiles are used to allow, disallow, or force the adding, changing, or setting of particular fields to the specified value for a certain profile. The command-related policy profiles do not allow control over fields or values, but provide only the functionality for the entire command.

In general, if you do not define a profile, zSecure Command Verifier behaves as if there is no specific policy and defers the authorization decision to RACF. Standard RACF processing is followed, as if zSecure Command Verifier were not implemented. If a policy profile exists, the access level, from access list and UACC, is usually interpreted as follows:

No profile found
: The policy rule is not implemented.

NONE
: The terminal user does not meet the requirements as described by the policy rule. Most often, the command is rejected. For Mandatory Value policy profiles, see “Mandatory and default value policy profiles” on page 62. The mandatory value is not applied.

READ
: Same as NONE. Also, in many situations, READ access is sufficient to remove an attribute, or specify an initial value.

UPDATE
: The terminal user does meet all the requirements as described by the policy rule. The command continues.

CONTROL
: The policy rule does not apply to this terminal user.

Read the specific descriptions for each profile for a description how this general usage applies to a specific policy rule.

Avoid warning mode

zSecure Command Verifier does not support the use of warning mode on policy profiles. Access decisions are based on the standard access list (ACL) and the universal access (UACC).

If you use warning mode on zSecure Command Verifier policy profiles, the results can be confusing. For example, if warning mode is enabled, you receive ICH408I messages that show `WARNING: INSUFFICIENT AUTHORITY - TEMPORARY ACCESS ALLOWED`.
However, access to the policy is not granted and processing ends.

The reason for the confusing messages is that, at the end of processing, after zSecure Command Verifier determined the required actions, a RACF audit-only request is used to create the appropriate audit trail through SMF. The audit-only request causes the ICH408I `WARNING` message. zSecure Command Verifier ignores the response of RACF to this audit-only request.

Avoid Global Access Checking (GAC)

zSecure Command Verifier does not support the use of Global Access Checking (GAC) for policy profiles. Access decisions are based on the standard access list (ACL) and the universal access (UACC).

For most policy decisions, the GAC-table is completely ignored. However, the GAC-table does process the audit-only request that creates the audit trail through SMF. If a GAC-table entry applies, auditing of keywords and parameters that are allowed by a policy might be suppressed.

RACFVARS profiles

You can specify RACFVARS in policy profiles so that conventional characters that do not fit the required pattern can be used. RACFVARS profiles can also specify complex patterns for user ID naming conventions.

In some policy profiles, and in some APPLDATA values, zSecure Command Verifier uses special values, like =RACUID, =RACGPID, =USERID, and =GROUP. The function of =RACUID is like the function of the RACF built-in variable &RACUID. However, because RACF treats the ampersand (&) as a special character, it cannot be used in general resource profiles like the zSecure Command Verifier policy profiles. Therefore, an equals sign (=) is used instead. In general, the following four special values have the following meaning:

=RACUID
The `userid` of the terminal user that issues the command.

=RACGPID
The list of connect groups of the terminal user issues the command. It is different from the use of &RACGPID by RACF. RACF uses it to represent only the current connect group, while zSecure Command Verifier uses it for all connect groups.

=USERID
The RACF USERID specified in the command.

=GROUP
The RACF GROUP specified in the command.

In addition to these default variables, you can also use RACFVARS in most policy profiles. You can use these variables in all places where regular generic characters are allowed, but where conventional generic characters (%, *, and.**) do not fit the required pattern. An example of conventional use of a RACF variable is shown in the following example. For user IDs, only two default groups can be used. You can define two profiles, but it is also possible to define one policy profile, and use a RACF variable to specify the exact values.

```
RDEFINE XFACILIT C4R.USER.DFLTGRP.DEPTA.*
RDEFINE XFACILIT C4R.USER.DFLTGRP.DEPTS.*

or

RDEFINE RACFVARS &DFLTGRP ADDMEM(DEPTA, DEPTS)
RDEFINE XFACILIT C4R.USER.DFLTGRP.&DFLTGRP*
```

Another example of the use of RACFVARS is the specification of more complex patterns for user ID naming conventions. Suppose that an installation uses the following naming conventions:

- The first character is an S, T, U, V, or a W.
- If the first character is an S, it is normally followed by 3 digits. However, if the third digit is an 8 or 9, an extra digit is used for a total of 4 digits.
- If the first character is a T, U, V, or W, it is always followed by 4 digits.

This naming convention clearly shows historical growth. In the past, there were a limited number of S-users. When more `userid` values were needed, two free digits were used to signal the use of an extra digit. The following list of correct and incorrect `userid` values illustrates the naming convention rules.

```
correct           incorrect
S000              S0002     (S plus 4 digits)
S784              S003H     (non-numeric 5th char)
S0082             S128      (3rd digit is 8, but no 4th digit)
S9194             SAHJ      (non-numerics)
U3425             U10255    (5 digits)
U9865             X0126     (illegal 1st char)
W2314             W813      (W plus 3 digits)
```

Figure 10. User ID naming convention example

To enforce this naming convention, zSecure Command Verifier policy profiles that use RACFVARS can be used. The first step consists of recognizing the different characters that are being used and defining RACFVARS for these different types.

```
RDEFINE RACFVARS &S ADDMEM(S)                    Special character
RDEFINE RACFVARS &F ADDMEM(T U V W)              First characters
RDEFINE RACFVARS &N ADDMEM(0 1 2 3 4 5 6 7 8 9)  Normal digits
RDEFINE RACFVARS &X ADDMEM(8 9)                  eXtension digits
RDEFINE RACFVARS &Y ADDMEM(0 1 2 3 4 5 6 7)      non-extension digits
```

The next step is to use these variables to define the three valid patterns.

```
&S&N&N&Y                 S plus three digits (non-extension)
&S&N&N&X&N               S plus four digits (extension)
&F&N&N&N&N               T,U,V,W plus four digits
```

The patterns are designed such that for each user ID, only one pattern applies. For instance, the definitions of &X and &Y do not overlap, and thus there is no ambiguity about which of the first two patterns apply. If the first pattern was &S&N&N&N, an ambiguity would arise when the third digit is an eight.

The last step is to use three patterns in a set of zSecure Command Verifier policy profiles:

```
C4R.USER.ID.*            UACC(NONE)
C4R.USER.ID.&S&N&N&Y     UACC(NONE)  UPDATE(RACFADM)
C4R.USER.ID.&S&N&N&X&N   UACC(NONE)  UPDATE(RACFADM)
C4R.USER.ID.&F&N&N&N&N   UACC(NONE)  UPDATE(RACFADM)
```

The first profile ensures that user IDs outside the naming convention cannot be created. The next three profiles allow RACFADM to create user IDs according to any of the three patterns. Remember to issue a **SETROPTS REFRESH RACLIST** for both the RACFVARS and the XFACILIT class. The RACFVARS class must be RACLISTed and REFRESHed before the XFACILIT class.

The preceding example shows how RACFVARS can be used in policy profiles. It clearly shows the benefits when you define patterns for naming conventions. In these cases, only a few policy profiles suffice to implement fairly complex conventions.

Policy profile selection

Use these guidelines to decide which general policies you want to implement, and whether you want to permit exceptions to the general policies.

Every RACF command and keyword has these basic control items:

- Field or attribute that is affected
- Terminal user who issues the command to set the field or attribute
- Profile that is affected

The following example illustrates these control items:

```
IBMUSER:    ALTUSER CRMAHJB DFLTGRP(SYS1)
```

The following list contains the control items.

- *DFLTGRP* selection
- Terminal user *IBMUSER*
- Object user *CRMAHJB*

In general, the design of the policies is based on the result of a command, and not on the command itself. So, the policies try to control the value of the attributes, independent of how such a value is set. For instance, for the **DFLTGRP**, it does not matter if the value is set during creation of the user ID, or afterward through a change of the user ID. The net result is what is controlled. It is also one of the main arguments why most policies do not take System-SPECIAL or similar authorizations into consideration.

If you want to implement a policy for the **DFLTGRP**, and you do not want any changes to the **DFLTGRP** of any user, you can implement a profile like:

```
C4R.USER.DFLTGRP.**   UACC(NONE)   ACL(empty)
```

If you want to make exceptions to the general rule, you must decide what type of exceptions you want to make. If you want an exception that is based on the terminal user who issued the command, you must modify the ACL and grant the terminal user, or one of its groups, with UPDATE access. If you want to make an exception that is based on the **DFLTGRP** itself, you can define a profile that contains the name of the **DFLTGRP**:

```
C4R.USER.DFLTGRP.SYS1.**
```

Aside from the direct control over the DFLTGRP, you can also verify other aspects of the **DFLTGRP**. For more information about the Additional Policy profiles for the **DFLTGRP**, see “Additional policy controls for the default group” on page 80.

Summary

In summary, you can define several profiles for **DFLTGRP** control.

```
C4R.USER.DFLTGRP.**          UACC(???)   ACL(empty)
C4R.USER.DFLTGRP./SCOPE.**   UACC(???)   ACL(empty)
C4R.USER.DFLTGRP./OWNER.**   UACC(???)   ACL(empty)
```

The first profile controls the authority to specify the DFLTGRP. The other profiles are examples of the Additional Policy profiles for the DFLTGRP. They control the authority to change the DFLTGRP to a group outside the group-scope of the terminal user, and the authority to make the DFLTGRP different from the OWNER of the USER profile.

Other policy profiles can be evaluated in a similar way.

General functions

Aside from the preceding result-related policy profiles, zSecure Command Verifier also uses some general profiles.

An example profile in this area is the profile that determines whether terminal users with FIELD access level authority can run commands for all RACF profiles or only for profiles in their regular scope.

Attention: Do not define a top generic profile like C4R.** or **. If you define such a top generic profile, the either of the following situations occurs, depending on the specified access:

- All RACF commands for which no specific policy profiles were defined are rejected.
- All zSecure Command Verifier controls are bypassed.

User exemption, violation suppression, error handling

General profiles are used to specify whether certain users are exempt from all zSecure Command Verifier policy rules, and what action zSecure Command Verifier must take in case of policy violations and other error situations.

The following list contains the general function profiles.

- C4R.EXEMPT

  This profile controls whether certain users are exempt from policy enforcement. If the terminal user has sufficient access, no further verification of any policy is done. It is not possible to detect if the command violated any policy, or if the exemption was needed for successful execution of the command. Auditing this event can be achieved by using the following command:

  ```
  RALT XFACILIT C4R.EXEMPT AUDIT(SUCCESS(UPDATE))
  ```

  The LOGSTR for the access event to this policy profile contains the command as entered.

  In many situations, it is preferable to use the C4R.ERROR.CONTINUE policy profile. The main advantage is that all policy profiles are examined, and that possible policy violations can be recorded through the log string of the access event to the C4R.ERRMSG.command auditing profile.

  The following access rules apply for the C4R.EXEMPT policy profile:

  No profile found
  No users are exempt from policy enforcement.

  NONE
  This terminal user is not exempt from policy enforcement.

  READ
  Same as NONE.

  UPDATE
  The user is not subject to any policy rule verification or enforcement. No audit trail of the various command keywords and options is created.

  CONTROL
  Same as UPDATE.

- C4R.SUPPRESS

  This profile controls whether zSecure Command Verifier must attempt to suppress keywords and parameter values that are a violation of the specified policy. If suppression is not feasible, the command is rejected anyway. This situation can be the case where suppression of the command leads to incorrect commands, or leads to commands that violate the policy. An example of such a situation can be found when a terminal user tries to explicitly set the password of another user to the DFLTGRP of that other user. Suppression of the new password value can result in the very situation that the policy is trying to avoid. In a situation like this one, zSecure Command Verifier fails the entire command, regardless of the access to the C4R.SUPPRESS policy profile. The following access rules apply.

No profile found
Keyword suppression is not to be attempted. Any policy violation results in failure of the command.

NONE
Keyword suppression is not to be attempted for this terminal user.

READ
Same as NONE.

UPDATE
When possible, keywords and parameter values that violate a policy are suppressed. As noted before, this suppression might not always be possible.

CONTROL
Same as UPDATE.

- `C4R.ERROR.CONTINUE`

  This policy profile is used to specify how errors during interpretation of policy profiles and policy violations are handled. An example policy profile error is the use of a circular definition of the OWNER and DFLTGRP for new user IDs. An example of a policy violation is the specification of a not-allowed owner for a new user ID. UPDATE access to this profile does bypass the normal processing of policy violations. ICH408I messages are still generated, but the regular zSecure Command Verifier violation messages are suppressed. The main use of this policy is as policy override during an implementation period, and as emergency authority against incorrect policy definitions. The following access rules apply:

  No Profile Found
  This control is not implemented. The command is rejected if any of the keywords are not acceptable.

  NONE
  The command is rejected if any error or policy violation occurs.

  READ
  Same as NONE.

  UPDATE
  The command is always allowed to continue, regardless of errors or policy violations. Policies that add or change keywords are executed.

  CONTROL
  Same as UPDATE.

  This policy need not be effective when the terminal user specified a list of profiles. In such situations, the only option available to the program might be to terminate the entire command.

  Also, some policy rules might require the command to be split into multiple commands. Splitting a command is not supported. The commands cannot be run correctly and must therefore be rejected.

Message control

Some profiles control whether certain warning and information messages are issued.

The following list contains these profiles:

- `C4R.DEBUG`

  This profile is now deprecated. Instead, use the `C4R.=MSG.CMD` profile.

- `C4R.=MSG.CMD`

  This profile controls whether message C4R913I is issued to show the command that is passed to RACF after Command Verifier policy processing. The terminal user needs at least READ access to the profile.

  The RACF command as shown might differ slightly from the command as entered. For instance, the command itself is always shown in its primary form, and not as one of the possible aliases.

  No Profile Found
  This control is not implemented. The command is not displayed before execution.

NONE
The command is not displayed before execution.

READ
The RACF command as approved or modified by zSecure Command Verifier is displayed through message C4R913I before execution.

UPDATE
Same as READ.

CONTROL
Same as READ.

- C4R.=MSG.SUPPRESSED

  This profile controls whether message **C4R899W** is issued when a keyword or parameter value is suppressed. Before version 1.12, these messages were automatically issued if the C4R.SUPPRESS policy profile was enabled, but now messages are issued only if this profile is also in effect.

  No Profile Found
  This control is not implemented. **C4R899W** messages are not issued.

  NONE
  C4R899W messages are not issued.

  READ
  Message **C4R899W** is issued when keyword suppression occurs.

  UPDATE
  Same as READ.

  CONTROL
  Same as READ.

- C4R.=MSG.MANDATORY

  This profile controls whether message **C4R898W** is issued when a zSecure Command Verifier policy overrides a mandatory keyword or parameter value of a user-specified keyword or parameter.

  No Profile Found
  This control is not implemented. **C4R898W** messages are not issued.

  NONE
  C4R898W messages are not issued.

  READ
  Message **C4R898W** is issued when a mandatory keyword or parameter value override occurs.

  UPDATE
  Same as READ.

  CONTROL
  Same as READ.

- C4R.=MSG.DEFAULTS

  This profile controls whether message **C4R897W** is issued when a zSecure Command Verifier policy supplies a default keyword or parameter value to complete the user-specified command.

  No Profile Found
  This control is not implemented. **C4R897W** messages are not issued.

  NONE
  C4R897W messages are not issued.

  READ
  Message **C4R897W** is issued when a default keyword or parameter value is supplied.

  UPDATE
  Same as READ.

  CONTROL
  Same as READ.

Site-specific policy messages

zSecure Command Verifier allows site-specific messages to be issued in addition to the standard messages that are included in the product. This section describes this function and how to specify such messages.

Whenever zSecure Command Verifier detects a policy violation, a specific message is issued. For example, when an administrator issues an ALTUSER command with options that are not allowed, like

```
ALTUSER userx SPECIAL
```

a message is issued that is similar to

```
C4R480E Special attribute not allowed, command terminated
```

Aside from the fixed C4R480E message in this example, it is also possible to define text for an additional C4R914I message that provides site-specific information about the policy; for example:

```
C4R914I SPECIAL Attribute cannot be delegated according to company standard S278-01
```

zSecure Command Verifier defines the message number but a zSecure Command Verifier administrator defines the message text. The C4R914I message is issued before the standard zSecure Command Verifier message. If no message text is defined, message C4R914I is not issued.

In addition to the C4R messages, an SMF record is written for auditing purposes. The SMF record includes the policy profile that prevented the command. RACF also shows the violation in message ICH408I; for example:

```
ICH408I USER(BCSCADM ) GROUP(BCSC    ) NAME(BCSC ADMINISTRATOR  )
  C4R.USER.ATTR.SPECIAL.USERS.USERX CL(XFACILIT)
  INSUFFICIENT ACCESS AUTHORITY
  FROM C4R.USER.ATTR.SPECIAL.USERS.* (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

In this example, the policy profile that controlled the setting of the SPECIAL attribute was a generic one. The site-specific text for the C4R914I message can be defined in the policy profile that is shown in the SMF record and ICH408I message. Using this approach, it is possible to specify unique messages for all policy profiles that your organization uses.

The text for the message is stored in the USRDATA of the policy profile. zSecure Command Verifier uses several USRDATA entries for retaining the Command Audit Trail (CAT). The site message text uses the USRDATA entry named $C4RMSGT. This USRDATA entry is independent and does not interact with those that are used for the Command Audit Trail. There are currently two methods to store the site message text in the USRDATA entry:

- Use the CKGRACF command, which is part of zSecure Admin.
- Use the RALTER command on the policy profile, using a specific prefix for the installation data.

In other sections that follow, these two methods are described in detail.

The USRDATA field is named $C4RMSGT. The name starts with a dollar sign. It is based on the US-international code page. If you are using a different code page, you must use the character representing x'5B'.

It is possible to use variables in the message text. These variables are replaced by various keywords or parameters that are present in the RACF command. Not all variables are present for all commands. If a variable has no value, it is replaced by an empty string, without any additional blank characters. The following variables are supported:

&PROFILE
: Contains the profile that is used in the command. If the command specifies multiple profiles, this variable contains the single profile that violates the policy. This is different from the &PROFILE variable that is used in the command replacement function.

&CLASS
Contains the RACF class name that relates to the command. For DATASET or general resources, it has the value that identifies the resource class of the profile. For user-related commands, it has the value USER and for group-related commands, it has the value GROUP. For CONNECT commands, the value is USER.

&USER
Contains the user ID that is used in the command. It can be the user ID in the ADDUSER or ALTUSER command. It can also be the user ID that is specified on a CONNECT command. If the command affects multiple user IDs, this variable contains the single ID that violates the policy. For some commands, the value of &USER is identical to the value of &PROFILE.

&GROUP
Contains the group that is used in the command. It can be the group in the ADDGROUP or ALTGROUP command. It can also be the group that is specified on a CONNECT command. If the command affects multiple groups, this variable contains the single group that violates the policy. For some commands, the value of &GROUP is identical to the value of &PROFILE.

&ACLID
Contains the ID that is specified on a PERMIT command. If the command affects multiple IDs, this variable contains the single ID that violates the policy. This is different from the &ACLID variable that is used in the command replacement function. The special value =STAR is used to represent granting access to ID(*). This variable has a value only for an access-related policy profile.

&ACLACC
Contains the access level that is granted by the ACCESS keyword of the PERMIT command. In addition to the regular access levels, the value DELETE represents that an ACL-entry is to be removed. For commands that set the UACC, the variable shows the UACC value. This variable has a value only for an access-related policy profile.

Setting message text using CKGRACF

The site message text can be set using the CKGRACF command.

The required command has the following format:

```
CKGRACF USRDATA class profile SET $C4RMSGT(message-text)
```

class
The zSecure Command Verifier policy profile resource class.

profile
The name of the (generic) policy profile.

message-text
The text to be used for the C4R914I message.

You must enclose the entire *message-text* between quotation marks. If you want the message text to be issued in mixed case, the quoted string must be followed by the lowercase letter c. The following shows an example command:

```
ckgracf usrdata XFACILIT 'C4R.CONNECT.ID.**'c
   set $C4RMSGT('See Corporate Instruction CI278-01'c)
```

You must have sufficient authorization to run the CKGRACF command. Two types of authorization are required. You must have update authority on CKG.CMD.USRDATA and on one of the scoping profiles, like

```
 CKG.USRDATA.scope-level.XFACILIT.$C4RMSGT
```

If *scope-level* is SCP, you also need sufficient access to the applicable group-tree-based CKG.SCP.** profile. For more information about the CKGRACF command and the required authorizations, see "CKGRACF Command Language" in the *zSecure Admin and Audit for RACF User Reference Manual*.

Instead of entering the CKGRACF command manually, you can also use the MU line command in the zSecure Admin resource profile overview. A panel similar to the following is presented:

```
------------------------------------------------------------------------------
                          zSecure Suite - Manage USERDATA
 Option ===>

 1  List         List specified entry
 2  Add          Add an entry with the specified value
 3  Set          Set value/flag for the Entry name
 4  Delete       Delete the Entry Name entirely

 Class . . . . . . . XFACILIT
 Profile . . . . . . C4R.CONNECT.ID.**
 Entry name  . . . . ________
 Entry flag  . . . . 00
 Entry value . . . . ____________________________________________________________
_______________________________________________________________________________
_______________________________________________________________________________
_____________________________________
 Reason
 ______________________________________________________________________________
```

Figure 11. Manage USERDATA

For the **Entry name**, specify $C4RMSGT, and for the **Entry value**, specify the message text within quotation marks, followed by a lowercase letter c. To set the value, use option **3** (Set). If you want to delete a site-message, you can fill in the fields and use option **4** (Delete). For this application, do not use option **2** (Add).

If the policy profile already has a value for $C4RMSGT, you can also use the S line command on the detail display entry for the USRDATA. A panel is displayed, where you can overtype the current message text.

For deleting an existing message, you can use the D line command in the detail display of the policy profile.

Setting message text using RALTER

The site message text can be set using the RALTER command.

The disadvantage of using the RALTER command is that RACF translates the site message text to uppercase. Using this method works only if zSecure Command Verifier is active. If the resource class is the zSecure Command Verifier policy profile resource class, zSecure Command Verifier inspects the value that is specified for the installation data. If the specified installation data starts with $C4RMSGT=, the text following the equals sign is stored in the USRDATA named $C4RMSGT. The existing value for the installation data is not modified. The following is an example of the command:

```
RALTER XFACILIT C4R.CONNECT.ID.**
   DATA('$C4RMSGT=See Corporate Instruction CI278-01')
```

The installation data starts with the name of the USRDATA entry. Only the character string $C4RMSGT= is currently recognized. The first character ($) is the character representing x'5B'.

If the value following the equals sign is the special value DELETE, the existing site message text in the $C4RMSGT USRDATA field is removed. The installation data itself is not modified.

If the prefix of the specified installation data field does not exactly match $C4RMSGT=, the value is not recognized as a zSecure Command Verifier site message text. The entire specified value is stored in the installation data field of the policy profile.

Temporary system-level attributes

zSecure Command Verifier provides a facility to use System-SPECIAL or System-AUDITOR authorization for a particular RACF command, even if the terminal user does not have this attribute.

Two types of policy profiles control this function. The first one unconditionally grants the system-level attribute for the duration of the command. The second policy profile grants the system-level attribute only if all relevant keywords in the RACF command are explicitly covered by policy profiles. If a keyword or parameter is used that is not explicitly authorized, the command runs without the temporary system-level attribute.

Unconditional temporary system-level attributes

Use the C4R.*command*.=SPECIAL and C4R.*command*.=AUDITOR profiles to implement unconditional temporary system-level attributes for a RACF command.

- C4R.*command.*=SPECIAL
- C4R.*command.*=AUDITOR

Users with UPDATE access to these profiles have RACF System-SPECIAL or System-AUDITOR authorization during the running of the command and during the running of potential **PRE-** and **POST-** commands. The qualifiers =SPECIAL and =AUDITOR in the policy profiles cannot be covered by generic characters. They must be present in the exact form shown. The following access rules apply:

No Profile Found
: This function is not implemented. The command runs with the regular authorization of the terminal user.

NONE
: The system-level attribute is not assigned. The command runs with the regular authorization of the terminal user.

READ
: Same as NONE.

UPDATE
: The command runs with the temporary system-level attribute. Possible **PRE-** and **POST-** commands are also run with this authorization.

CONTROL
: Same as UPDATE.

Controlled temporary system-level attributes

The difference between the Controlled Temporary system-level attributes and the Unconditional ones is the requirement that all keywords and parameters must be covered by applicable zSecure Command Verifier policy profiles. If one such profile is absent, the Controlled Temporary attribute policy does not apply, and temporary System-SPECIAL or System-AUDITOR is not granted.

The Controlled Temporary attributes policy gives an installation more granular control over which functions must run with system-level authorization and those functions that do not. However, designing correct controls for effective use of Controlled Temporary attributes is rather complex because the definition of individual policy profiles now has a dual effect: It determines whether a keyword or parameter is authorized to be used, and it determines whether temporary system-level authorization must be granted. The easiest method to handle this duality is to treat the definition of policy profiles as a way to specify which keywords and parameters can be present without affecting the temporary system-level authorization. For example, you might define policies by using the following strategy:

- Do not define any policy to control keywords and parameters that you do not care about; for example, the ADSP attribute in user profiles.
- Define appropriate policy profiles to prevent the use of certain keywords and parameters; for example, the OWNER of a user profile.
- Define a Controlled Temporary Special or Controlled Temporary Auditor policy profile for selected commands.

Temporary attributes are now applied only for those commands that are allowed, and for which a policy profile exists. So, if somebody wants to change the ADSP attribute and there is no policy profile for setting this attribute, the command is passed to RACF, which in turn can accept or reject the command. The commands are not run with temporary SPECIAL or temporary AUDITOR because there is no applicable policy profile for the ADSP attribute. If somebody wants to change the OWNER of a user profile, zSecure Command Verifier can accept or reject the command, based on the access to the OWNER policy profile. If the specified OWNER value is allowed, the command can run with temporary SPECIAL or temporary AUDITOR, since there is a policy profile. If the specified OWNER is not allowed, the command is rejected, and it does not matter if it would have run with or without any temporary authorization.

The major issue to watch when you use the Controlled Temporary system-level attribute policy profiles is the exact list of keywords and parameters that are now authorized, not just in the original context of the permanent RACF authorizations, but also in the context of the temporary authorization.

When you grant Controlled Temporary Special or Auditor, you must carefully verify whether all currently authorized changes to profiles must indeed run with SPECIAL or AUDITOR authorization. A strategy that can help prevent unintended side effects is to never define a policy profile that explicitly allows the same function as allowed in the absence of the policy profile.

The following policy profiles can be used to implement Controlled Temporary system-level attributes. The access levels that are supported are explained in detail.

- C4R.*command.*=CTLSPEC
- C4R.*command.*=CTLAUD

Users with UPDATE access to one of these profiles have RACF `System-SPECIAL` or `System-AUDITOR` during the execution of the command and during execution of potential **PRE-** and **POST-** commands. The policy applies only if *all* keywords and parameters are controlled by a zSecure Command Verifier policy profile. The qualifiers =CTLSPEC and =CTLAUD in the policy profiles cannot be covered by generic characters. They must be present in the exact form shown.

You can use this policy even if the command itself is not dependent on the system-level attribute. In that situation, the system-level attribute might still have an effect on the authorization for the optional PRE- and POST- commands. You can, for example, use this to set the UAUDIT attribute for a user when the CONNECT command is issued for a sensitive group.

The following access rules apply:

No Profile Found
: This function is not implemented. The command runs with the regular authorization of the terminal user.

NONE
: The system-level attribute is not assigned. The command runs with the regular authorization of the terminal user.

READ
: Same as NONE.

UPDATE
: The command runs with temporary `System-SPECIAL` or `System-AUDITOR` authorization. Possible **PRE-** and **POST-** commands are also run with this authorization.

CONTROL
: Same as UPDATE.

The Controlled Temporary system-level attribute policy profiles do not require that *all* possible zSecure Command Verifier policy profiles are defined. Only those policy profiles that actually determine the authority to use a particular keyword or parameter are required. For example, when you want to grant READ access to user USRXYZ for data set MYHLQ.TEST.DSN, you do not need a policy profile that allows the use of user IDs instead of GROUPs on access lists, which is the `ACL./GROUP` policy profile. The only controlling profile that is needed in this example is the one for the following resource:

```
C4R.DATASET.ACL.USRXYZ.READ.MYHLQ.TEST.DSN
```

This resource might be controlled by the following policy profile:

```
C4R.*.ACL.USRXYZ.READ.**
```

The following list shows all policy profiles that are used to determine whether Controlled Temporary system-level attributes are applied. In this list, generics are used to denote one or more qualifiers that can have multiple values.

```
C4R.USER.ID.*              C4R.USER.DELETE.*          C4R.USER.DFLTGRP.*
C4R.USER.OWNER.*           C4R.USER.ATTR.*            C4R.USER.MFA.*
C4R.USER.PASSWORD.*        C4R.USER.PWINT.*           C4R.USER.PWEXP.*
C4R.USER.NAME.*            C4R.USER.INSTDATA.*        C4R.USER.CLAUTH.*
C4R.USER.SECLABEL.*        C4R.USER.SECLEVEL.*        C4R.USER.CATEGORY.*
C4R.USER.MODEL.*           C4R.USER.WHEN.*            C4R.USER.segment.*
C4R.USER.segment./SCOPE
C4R.GROUP.ID.*             C4R.GROUP.DELETE.*         C4R.GROUP.SUPGRP.*
C4R.GROUP.OWNER.*          C4R.GROUP.ATTR.*           C4R.GROUP.INSTDATA.*
C4R.GROUP.MODEL.*          C4R.GROUP.segment.*        C4R.GROUP.segment./SCOPE
C4R.CONNECT.*              C4R.REMOVE.*               C4R.CONNECT.OWNER.*
C4R.CONNECT.AUTH.*         C4R.CONNECT.UACC.*         C4R.CONNECT.ATTR.*
C4R.class.ID.*             C4R.class.OWNER.*          C4R.class.UACC.*
C4R.class.ACL.*            C4R.class.CONDACL.*        C4R.class.VOLUME.*
C4R.class.UNIT.*           C4R.class.RACFIND.*        C4R.class.TYPE.*
C4R.class.ATTR.*           C4R.class.INSTDATA.*       C4R.class.NOTIFY.*
C4R.class.APPLDATA.*       C4R.class.SECLABEL.*       C4R.class.CATEGORY.*
C4R.class.SECLEVEL.*       C4R.class.LEVEL.*          C4R.class.RETPD.*
C4R.class.segment.*        C4R.class.segment./SCOPE
```

Figure 12. Policy profiles used to determine whether Controlled Temporary system-level attributes can be assigned

Profiles for managing multifactor authentication (MFA) data

RACF implemented support for multifactor authentication through new function APARs to both RACF and SAF. The required data can be added to USER profiles and to general resource profiles in the MFADEF resource class.

In the USER profiles, the relevant data is kept in several MFA-related fields in the BASE segment. Although technically, the MFA data in the USER profile is not kept in a separate segment, zSecure Command Verifier handles the information as if an independent segment is being used. That means that for the USER MFA data, policy profiles are used that are similar to the existing policy profiles for segments. The MFA data in USER profiles is thus covered by two types of policy profiles:

- Segment management policy profiles, as described in "Profiles for controlling management of non-base segments" on page 52
- MFA related fields, as described in "Policy profiles for USER MFA data management" on page 101

The MFA information for the profiles in the MFADEF general resource class is kept in the MFA and the MFPOLICY segments. The only function that RACF provides for the MFA segment is adding or removing the segment. The contents of the MFA segment cannot be managed using RACF commands. Consequently, the only policy that Command Verifier provides is one for managing the presence of the MFA segment. In contrast, the contents of the MFPOLICY segment is managed using RACF commands. Command Verifier provides policy profiles to control managing the presence of the MFPOLICY segment and the setting of keywords and parameters. For more information, see "Profiles for controlling management of non-base segments" on page 52 and "Policy profiles for MFPOLICY segment management" on page 180.

Profiles for controlling management of non-base segments

RACF allows management of information in non-base segments, such as the OMVS and TSO segments to all System-SPECIAL users and to all users with sufficient access to profiles in the FIELD class. The latter method is often referred to by the term *Field Level Access Checking*.

In some situations, it might be desirable to restrict management of these types of segments even further. Although MFA data in the USER profile is technically not in a separate segment, zSecure Command Verifier handles the information as if it is contained in a segment. In the remainder of this section, the variable segment also applies to the MFA data in the USER profile. To allow control over the non-base segment, zSecure Command Verifier implements three types of profiles.

- C4R.*class.segment*.=RACUID

  This policy profile is used to control the authority to manage your own segment information. The effect of this policy profile is similar to placing &RACUID on the access list of the corresponding profile in the FIELD class.

- C4R.*class.segment*

  This policy profile controls the authority to manage segment information for user profiles other than your own.

- C4R.*class.segment.*/SCOPE

  This policy profile can be used to control the scope of authority for management of segment information.

Either the first or the second profile is used to determine the authority of the terminal user to manage the non-base segment. For users without System-SPECIAL, the third profile can be used as well to reduce the scope of control. If the terminal user must be able to display their own TSO information, the following two profiles must be in place:

```
XFACILIT     C4R.USER.TSO.=RACUID           userid(READ)
FIELD        USER.TSO.**                    &racuid(READ);
```

For allowing the same terminal user to display the TSO information of other users the following two profiles must be in place:

```
XFACILIT     C4R.USER.TSO                   userid(READ)
FIELD        USER.TSO.**                    userid(READ)
```

In these scenarios, both the zSecure Command Verifier and the field profile must be in place. The implementations can also be mixed, as in the following example.

```
XFACILIT     C4R.USER.TSO.=RACUID           uacc(NONE) userid(READ)
XFACILIT     C4R.USER.TSO                   uacc(NONE)
FIELD        USER.TSO.**                    uacc(READ)
```

In this situation, the terminal user can manage the TSO segment of the user's own user profile according to the first zSecure Command Verifier profile. This command is also authorized by the field profile. This field profile also allows displaying the TSO segment of all other users. However, the second zSecure Command Verifier policy profile that is shown prevents that.

Attention: See “Policy profiles for USS segment management” on page 201 for important information about how to restrict certain field value assignments in non-base segments. Without appropriate profiles, providing UPDATE access from field level access can create undesirable effects.

The following section describes the profiles and access levels in detail.

- C4R.*class.segment*.=RACUID

  This profile is used if the terminal user tries to display or change the *segment* in their own user profile. Because the =RACUID qualifier refers to the terminal user itself, the policy profile is only applicable for the USER *class*. If the profile does not exist, or does not allow access, authorization verification continues with the general profile for all users that are described in the next entry (C4R.*class.segment*). You cannot use generic characters to cover the =RACUID qualifier in the policy profile; it must be present in the exact form shown.

  Use care when you define a generic value for the segment name because the resulting policy profile might also match the authority to change your own password or password phrase. For more information about the policy profiles for passwords and password phrases, see “Policy profiles for user password and phrase management” on page 95.

  The following access rules apply:

  No Profile Found
  : This control is not implemented. zSecure Command Verifier does not control access to *segment*. RACF controls access to the *segment* according to the definitions in the field class.

  NONE
  : The terminal user cannot access the *segment* information of their own user ID. However, this restriction can be overruled by the general segment access policy profile. This restriction is independent of the definition of profiles in the field class.

READ
The terminal user can display the *segment* information of the user. This ability is also subject to the appropriate access to the profiles defined in the field class.

UPDATE
The terminal user can update the *segment* information of the user. This ability is also subject to the appropriate access to the profiles defined in the field class.

CONTROL
Same as UPDATE.

- C4R.*class.segment*

This profile is used if the terminal user tries to display or change the *segment* information for any user profile. This profile is also used if the terminal user tried to access the *segment* of the user ID, but is not authorized by the profile that is mentioned before. The following access rules apply:

No Profile Found
This control is not implemented. zSecure Command Verifier does not control access to *segment*. RACF controls access to the *segment* according to the definitions in the field class.

NONE
The terminal user cannot access the *segment* information of the target user ID. This restriction is independent of the definition of profiles in the field class.

READ
The terminal user can display the *segment* information. This ability is also subject to the appropriate access to the profiles defined in the field class. If the target user ID is outside the `Group-SPECIAL` scope of the terminal user, or the terminal user does not have any `Group-SPECIAL` attribute, the /SCOPE policy profile that is described in the following section applies.

UPDATE
The terminal user can update the *segment* information. This ability is also subject to the appropriate access to the profiles defined in the field class. If the target user ID is outside the `Group-SPECIAL` scope of the terminal user, or the terminal user does not have any `Group-SPECIAL` attribute, the /SCOPE policy profile that is described in the following section applies.

CONTROL
Same as UPDATE.

Currently, the following values are supported for the qualifier *segment* in the preceding profiles:

USER
CICS, DFP, LANGUAGE, NETVIEV, OMVS, OPERPARM, TSO, WORKATTR, OVM, DCE, NDS, LNOTES, KERB, PROXY, EIM, CSDATA, MFA

GROUP
DFP, OMVS, OVM, TME, CSDATA

DATASET
DFP, TME, CSDATA

General Resource
SESSION, DLFDATA, SSIGNON, STDATA, SVFMR, TME, KERB, PROXY, EIM, CDTINFO, ICTX, CFDEF, ICSF, SIGVER, MFA, MFPOLICY, IDTPARMS, JES, CSDATA

Scoping rules to manage segments

In RACF, access to profiles in the field class controls access to the fields in the non-base segments of all profiles. You cannot allow a decentralized administrator to manage, for example, the TSO segments of only those users that fall within the `Group-SPECIAL` scope of the administrator.

Starting with z/OS 2.3.0, RACF provides a method to add a check for the authority of the terminal user to the base segment before granting access to the non-base segments. Presence and access to the profile FLAC.SKIP.BASECHECK in the FIELD class determine if access to the base segment is used. This authority applies to all non-base segments and does not provide granularity per segment type. The zSecure Command Verifier scoping policies that this section describes provide segment scoping based on the

segment-type. Also, the zSecure Command Verifier scoping policies are based only on System-SPECIAL and Group-SPECIAL attributes and ignore direct ownership of the target profiles.

zSecure Command Verifier provides a facility that allows enforcing the same scoping rules as used for the normal RACF (is BASE) segment, for the non-base segments. Using the /SCOPE profile, it is possible to restrict these users to just the profiles that are within their BASE-segment scope.

zSecure Command Verifier does not replace the RACF access control to the non-base segment information. If the decentralized administrator does not have access through field level access checking, the administrator still cannot view or modify non-base segments. Full implementation of *Scoping of Segment Management* requires that all decentralized administrators who must maintain the non-base segments of their profiles have access to the corresponding field profiles.

Although technically, the MFA data in the USER profile is not kept in a separate segment, zSecure Command Verifier handles the MFA data in the USER profile as if it is contained in a segment. The /SCOPE policy profiles as described in this section can be used to limit management of MFA data in USER profiles to users with group-SPECIAL or system-SPECIAL. Access to profiles in the FIELD class is not required.

zSecure Command Verifier does not consider direct ownership of the target profile in this scoping rule. Only group-SPECIAL is used for determining the scope of control.

Terminal users with system-SPECIAL authorization are exempt from this control because all profiles in the system are considered to be in their scope.

For terminal users with group-SPECIAL or system-SPECIAL, use of their administrative authority over the base segment of the target profile is recorded through the audit-only policy profile

- C4R.USESCOPE.*group*

Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the base segment of the target profile in the command. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

Attention: See “Policy profiles for USS segment management” on page 201 for important information about how to restrict certain field value assignments in non-base segments. Without appropriate profiles, activation of Segment Management Scoping and providing UPDATE access from field level access to decentralized administrators can create undesirable effects.

- C4R.*class.segment.*/SCOPE

  The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The following access rules apply:

  No Profile Found
  This control is not implemented. Standard RACF rules apply. The non-base segments of all profiles can be accessed according to the definitions in the field class.

  NONE
  The terminal user cannot access any non-base segment outside the standard RACF scope. For profiles within the scope, the access level to the respective field class profiles controls if the fields can be displayed or modified.

  READ
  The terminal user can display the authorized non-base segments of all profiles in the system. For profiles outside the scope, only list commands are allowed. For profiles within the scope, all commands are allowed. The access level to the respective field class profiles controls if the fields can be displayed or modified.

  UPDATE
  The terminal user can modify the authorized non-base segments of all profiles in the system. The profiles in the field class still control if a field is accessible for display or modify to a particular terminal user.

  CONTROL
  Same as UPDATE.

RACF command replacement

zSecure Command Verifier provides a way to replace commands with other commands by a combined add/replace approach.

The first step is to specify a pre-command or a post-command. The second step is to specify whether the original commands must be run, maybe stripped of some keywords, or not. It can be controlled by three profiles. In the pre-command and post-command, several fields from the original RACF command can be referenced by variables. For instance, the target class and profile can be specified by &CLASS and &PROFILE.

The specified pre- and post-commands run with the same authority as the original RACF command. If temporary special or auditor authorization is specified for the original RACF command, the pre- and post-commands also run with temporary special or auditor. This also applies to controlled temporary attributes. It is the responsibility of the Command Verifier Policy administrator to specify pre- and post-commands that are appropriate for the environment in which they run.

Note: Currently this function is only available for the following commands and keywords:

Table 7. Commands and keywords supported by the Command/Keyword Replace Function.

Command	**Keyword**	**Keyword-qualification**
ALTUSER	RESUME	RESUME
ALTUSER	REVOKE	REVOKE
ALTUSER	RESUME(date) NORESUME	RESUMEDT
ALTUSER	REVOKE(date) NOREVOKE	REVOKEDT
ADDUSER ALTUSER	SPECIAL	SPECIAL
ADDUSER ALTUSER	OPERATIONS	OPERATIONS
ADDUSER ALTUSER	AUDITOR	AUDITOR
ADDUSER ALTUSER	*segment* no*segment*	*segment.action* *action*={Add | Alt | Del}
ADDUSER ALTUSER	OMVS(UID(0)) OVM(UID(0))	UID0
ADDUSER ALTUSER	OWNER(*owner*)	OWNER.owner
CONNECT	GROUP(grpname)	GROUP.*grpname*
PERMIT	CLASS(*class*)	CLASS.*class*
REMOVE	GROUP(grpname)	GROUP.*grpname*

The general form of the command replacement policy profile is:

```
C4R.command.function.keyword-qualification
```

The *command* is the non-abbreviated RACF command issued by the terminal user. The *function* indicates which part of the command replacement feature is controlled by this policy profile. Possible values for *function* are **=PRECMD**, **=PSTCMD**, and **=REPLACE**. These are used to specify the PRE-command and the POST-command and to indicate if and how the original RACF command is issued.

The possible values for the *keyword-qualifier* are dependent on the command:

- For setting attributes through the **ADDUSER** or **ALTUSER** command, the *keyword-qualifier* consists of only a single qualifier. Examples are the REVOKE, RESUME, and SPECIAL qualifiers.

- When managing the owner of users, the keyword-qualifier consists of the fixed value OWNER, followed by the specified value for the new owner.
- When managing user segments, the *keyword-qualifier* consists of two qualifiers. The first is the name of the segment, and the second is an action qualifier. The action qualifier can be ADD, ALT, or DEL.
- When managing user-to-group connects through the **CONNECT** or **REMOVE** command, the *keyword-qualifier* consists of the fixed value GROUP, followed by the name of the group used in the command.
- For changing the access list through the **PERMIT** command, the *keyword-qualifier* consists of two qualifiers. The first is the fixed value CLASS, and the second is the resource class name.

The special qualifier =PRECMD, =PSTCMD, or =REPLACE must be explicitly coded in the policy profile. It cannot be matched by generic characters. Other qualifiers in these policy profiles like the command or the resource class can be described by generic characters.

The following list contains some sample policy profiles.

```
C4R.*.=PRECMD.SPECIAL
C4R.ALTUSER.=PRECMD.REVOKE
C4R.ALTUSER.=PSTCMD.TSO.ADD
C4R.A*.=PRECMD.*.A*
C4R.PERMIT.=PSTCMD.CLASS.DATASET
```

See the following list for the detailed description of the profiles and the supported access levels.

- C4R.*command*.=PRECMD.*keyword-qualification*

  This profile specifies the command that must be run before the original RACF command. The pre-command is specified by the APPLDATA of the profile. The most common use of this profile is to replace the **ALTUSER RESUME** command by a **CKGRACF RESUME** command.

  If more than one keyword matches an =PRECMD profile, any of the profiles can be used to specify the pre-command. The profile that is used by zSecure Command Verifier is unpredictable.

  The qualifier =PRECMD in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  If the pre-command fails during execution, the original, or modified RACF command is suppressed. This way, dependent actions in the modified RACF command are only run if the prerequisite action from the pre-command is completed.

  The following access rules apply:

  No Profile Found
  This control is not implemented. No pre-command is issued.

  NONE
  The pre-command that is specified in this profile is not run for this terminal user.

  READ
  The pre-command that is defined by the APPLDATA is run before the original RACF command.

  UPDATE
  Same as READ.

  CONTROL
  Same as UPDATE.

- C4R.*command*.=REPLACE.*keyword-qualification*

  This profile specifies whether the original keyword must be kept or suppressed or if the entire RACF command must be suppressed. If the pre-command fails, the original RACF command is not run. This case is independent of the definition of the =REPLACE profile.

  If the *keyword* is present in the command, the action is controlled by the access rules that are specified in the following list. If more than one keyword matches an =REPLACE profile, all of these profiles can be used to suppress keywords or the entire command.

For the **CONNECT** and **REMOVE** commands, the only supported keyword qualification is for the group. Suppression of the GROUP keyword is not effective because, in the absence of the GROUP keyword, RACF automatically uses the terminal user's current connect group for the command. The resulting command does not have the intended effect. For this reason, the **CONNECT** and **REMOVE** commands do not support suppression.

The qualifier =REPLACE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

The following access rules apply:

No Profile Found
This control is not implemented. The keyword is not removed.

NONE
The keyword suppress is not done for this terminal user.

READ
The keyword is suppressed. This suppression can result in a command without any effective keywords. For the **CONNECT** and **REMOVE** commands, the effect of READ is the same as NONE: the keyword is not suppressed.

UPDATE
The entire command is suppressed. This suppression can result in error flags that are being presented to the terminal user, indicating that the command failed.

CONTROL
Same as UPDATE.

- C4R.*command*.=PSTCMD.*keyword-qualification*

  This profile specifies the command that must be run after the original RACF command. The post-command is specified by the APPLDATA of the profile. In the command, the target class and profile can be specified by &CLASS and &PROFILE.

  If more than one keyword matches an =PSTCMD profile, any of the profiles can be used to specify the post-command. The profile that is used by zSecure Command Verifier is unpredictable.

  The qualifier =PSTCMD in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  The following access rules apply:

  No Profile Found
  This control is not implemented. No post-command is issued.

  NONE
  The post-command that is specified in this profile is not run for this terminal user.

  READ
  The post-command that is defined by the APPLDATA is run after the original RACF command. If the original RACF command issues a warning message, the post-command is suppressed. This access level can be useful for some RACF commands like **ALTUSER** and **ALTGROUP** that issue only a warning message, even if the command fails completely.

  UPDATE
  The post-command that is defined by the APPLDATA is run after the original RACF command. If the original RACF command failed with an error message or an abend, the post-command is suppressed.

  CONTROL
  Same as UPDATE.

The APPLDATA of =PRECMD and =PSTCMD profiles can be used to specify the command that is to be run before and after the original RACF command. The Command Verifier policy can specify multiple commands, that are separated by a semicolon (;), up to 255 characters. Command Verifier executes each command in the order that it is provided before moving on to the next command. If a command fails, all the following commands are not executed. Because of the way that RACF handles the **APPLDATA** field,

the value that is entered is folded to uppercase. In the specified command string, variables can be used to refer to parts of the original RACF command. Variables are prefixed by an ampersand (&) sign. The following variables are supported:

&CLASS
Represents the CLASS of the PROFILE. For the **ALTUSER** command, this value is USER. For the **PERMIT** command, the value is DATASET or the general resource class specified.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&CLASS ---> FACILITY

ALTUSER IBMUSER REVOKE
&CLASS ---> USER
```

&PROFILE
Represents the PROFILE. For the **ALTUSER** command, it is the affected user ID. For the **PERMIT** command, it is the fully qualified data set name or the general resource profile name.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&PROFILE ---> STGADMIN.**

ALTUSER IBMUSER REVOKE
&PROFILE ---> IBMUSER
```

&PROFILE(1)
Represents one PROFILE. For the **ALTUSER** command, it is one of the affected user IDs. For the **PERMIT** command, it is one of the fully qualified data set names or general resource profile names. Which profile is used is unpredictable.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&PROFILE(1) ---> STGADMIN.**

ALTUSER (IBMUSER) REVOKE
&PROFILE(1) ---> IBMUSER

ALTUSER (IBMUSER, C4RTEST) REVOKE
&PROFILE(1) ---> C4RTEST (maybe)
```

&SEGMENT
Represents the list of USER SEGMENTs that are being managed in this command.

```
ALTUSER IBMUSER TSO OMVS(UID(0))
&SEGMENT ---> TSO OMVS
```

&SEGMENT(1)
Represents one of the USER SEGMENTs that are being managed in this command. Which SEGMENT is used is unpredictable.

```
ALTUSER IBMUSER TSO OMVS(UID(0))
&SEGMENT(1) ---> OMVS (maybe)
```

&RACUID
Represents the user ID of the terminal user that is issuing the command.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&RACUID ---> CRMAHJB (maybe)

ALTUSER IBMUSER REVOKE
&RACUID ---> CRMAHJB (maybe)
```

&RACGPID
Represents the current connect GROUP of the terminal user that is issuing the command.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&RACGPID ---> CRMA (maybe)

ALTUSER IBMUSER REVOKE
&RACGPID ---> CRMA (maybe)
```

&DATE
Represents the current date in Julian format (YY.DDD). The Julian date is the same format as used by RACF in the LISTUSER output.

```
ALTUSER IBMUSER REVOKE
&DATE ---> 04.060 (maybe)
```

&TIME
Represents the current time in 24 hour format (HH:MM:SS). This time format is the same as used by RACF in the LISTUSER output.

```
ALTUSER IBMUSER REVOKE
&TIME ---> 08:17:31 (maybe)
```

&SYSID
Represents the SMF System Identifier of the current system. This variable is the four character string that is specified by SMFPARMxx in parmlib. It is the same value that can be used in the conditional access list of PROGRAM profiles.

```
ALTUSER IBMUSER REVOKE
&SYSID ---> IDFX (maybe)
```

&ACLID
Represents the list of IDs of both user and GROUPs specified in the ID keyword of the **PERMIT** command. The list can consist of a single value, or a blank separated list. Leading and trailing blanks are not included.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&ACLID ---> IBMUSER C4RTEST
```

&ACLID(1)
Represents one of the IDs of both user and GROUPs specified in the ID keyword of the **PERMIT** command. Which one of the IDs is used is not predictable.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER,C4RTEST) ACCESS(READ)
&ACLID(1); ---> C4RTEST (maybe)
```

&ACLACC
Represents the access level that is granted by the ACCESS keyword of the **PERMIT** command. In addition to the regular access levels, the value DELETE represents that an ACL-entry is to be removed.

It is also possible to substitute by using a substring of the ACCESS level. This substitution can be specified by a single digit between parenthesis immediately following the string &ACLACC. Only a single digit from 1 to 8 is allowed, and the total substring specification must consist of exactly 3 characters. Any other format is treated as a regular character string.

```
PERMIT STGADMIN.** CLASS(FACILITY) ID(IBMUSER C4RTEST) ACCESS(UPDATE)
&ACLACC    ---> UPDATE
&ACLACC(3); ---> UPD
```

Example 1

In this example, two profiles replace the **ALTUSER RESUME** command with the zSecure Admin Resume function.

```
XFACILIT:   C4R.ALTUSER.=PRECMD.RESUME
  UACC:     UPDATE
  APPLDATA: 'CKGRACF &class &profile RESUME'

XFACILIT:   C4R.ALTUSER.=REPLACE.RESUME
  UACC:     READ
```

By using the two profiles, the following substitution takes place:

```
Input:   ALTUSER userid PASSWORD(password) RESUME
Precmd:  CKGRACF USER userid RESUME
Maincmd: ALTUSER userid PASSWORD(password)
```

Note: When you use the substitution, verify that necessary files for CKGRACF （SYSTERM）are available.

Example 2

Another example of command replacement can be done for the REVOKE keyword. This keyword can be replaced by a **CKGRACF DISABLE**. This **DISABLE** can be undone only by **CKGRACF ENABLE** commands.

If the **RESUME** function is translated into a **CKGRACF RESUME**, most resume attempts fail because of the **DISABLE** schedule. The REVOKE can be converted by definition of profiles:

```
XFACILIT:   C4R.ALTUSER.=PRECMD.REVOKE
  UACC:     UPDATE
  APPLDATA: 'CKGRACF &class &profile SCHEDULE GRPADMIN DISABLE TODAY'

XFACILIT:   C4R.ALTUSER.=REPLACE.REVOKE
  UACC:     UPDATE
```

By using the two profiles, the following substitution takes place:

```
Input:   ALTUSER userid REVOKE
Precmd:  CKGRACF USER userid SCHEDULE GRPADMIN DISABLE TODAY
Maincmd: none
```

See the *IBM Security zSecure Admin and Audit for RACF: User Reference Manual* for detailed documentation of the **CKGRACF** command and the required authorization to manage Revoke/Resume schedules.

Example 3

In this example, a precommand is used to **ENABLE** a user on **ALTUSER RESUME** when the **REVOKE** is converted with **CKGRACF DISABLE**.

In the scenario where a **REVOKE** is converted, as described in “Example 2” on page 61, it is possible to automatically **ENABLE** the user on an **ALTUSER RESUME**. In this case, the suggested approach is to use a pre-command to attempt to **ENABLE** the user. If a **CKGRACF ENABLE** command is issued, **CKGRACF** determines whether other schedules prevent the user from being resumed. If not, **CKGRACF** automatically resumes the user at the **ENABLE** date (=today). The **RESUME** can be converted by definition of profiles:

```
XFACILIT:   C4R.ALTUSER.=PRECMD.RESUME
  UACC:     UPDATE
  APPLDATA: 'CKGRACF &class &profile SCHEDULE GRPADMIN ENABLE TODAY'

XFACILIT:   C4R.ALTUSER.=REPLACE.RESUME
  UACC:     READ
```

By using the two profiles, the following substitution takes place:

```
Input:   ALTUSER userid PASSWORD(password) RESUME
Precmd:  CKGRACF USER userid SCHEDULE GRPADMIN ENABLE TODAY
Maincmd: ALTUSER userid PASSWORD(password)
```

See the *IBM Security zSecure Admin and Audit for RACF: User Reference Manual* for detailed documentation of the **CKGRACF** command and the required authorization to manage Revoke/Resume schedules.

Example 4

In this example, some **PERMIT** commands are replaced by a **CONNECT** to the appropriate group.

The name of the group is derived from the access level. The current implementation allows only ACCESS to be truncated. No provision is made to create **REMOVE** commands if the specified ACCESS is DELETE.

```
XFACILIT:   C4R.PERMIT.=PRECMD.CLASS.SDSF
  UACC:     UPDATE
  APPLDATA: 'CONNECT &ACLID GROUP(SDSF#ACLACC(1))'

XFACILIT:   C4R.PERMIT.=REPLACE.CLASS.SDSF
  UACC:     UPDATE
```

By using the profiles, the following substitution takes place:

```
Input:   PERMIT profile CLASS(SDSF) ID(IBMUSER) ACCESS(READ)
Precmd:  CONNECT IBMUSER GROUP(SDSF#R)
Maincmd: none
```

Group-Special authorization restriction

Use these guidelines to understand the scope of zSecure Command Verifier policy profiles.

In this version, zSecure Command Verifier recognizes only the system-wide and group-related SPECIAL attributes for command authorizations. zSecure Command Verifier ignores all other command authorization methods, such as group operations, group connect authorizations JOIN, CONNECT, CREATE, and direct ownership, where the terminal user is the owner of the affected RACF profiles. For the **RDEFINE** and **ADDUSER** command, zSecure Command Verifier recognizes the **CLAUTH** attribute, if the command otherwise conforms to the specified policy.

Mandatory and default value policy profiles

In zSecure Command Verifier, you can use profiles that enforce a specific value for a keyword. The value overrides anything that is specified by the terminal-user.

These profiles are called *Mandatory Value* policy profiles. They can be used only when the RACF command requires the keyword or uses a default value. This restriction means that for most keywords, the Mandatory Value policy profiles can be used only for create or add command types such as **ADDUSER**. They all have a third qualifier that starts with =, for example, =DFLTGRP. These profiles are examples of Mandatory Value policy profiles:

- C4R.DATASET.=UACC.SYS1.LINKLIB

  This profile specifies a mandatory value for the UACC of the SYS1.LINKLIB data set. The value for the UACC is specified in the **APPLDATA** field of the policy profile.
- C4R.USER.=OWNER.IBM*

  This profile specifies that the OWNER of a user ID, if it matches the pattern IBM*, must be equal to a certain value. The value is specified in the **APPLDATA** field of the policy profile.

When Mandatory Value policy profiles are present, they override any value that the terminal user specified. So, in the second example (C4R.USER.=OWNER.IBM*), if the terminal user enters the command:

```
ADDUSER IBMTEST OWNER(CMDVFY)
```

and the **APPLDATA** of the Mandatory Value policy profile contains the value SYS1, the actual command that is passed to RACF is:

```
ADDUSER IBMTEST OWNER(SYS1)
```

The terminal user must have sufficient RACF authorization to create the User profile. If access is insufficient, RACF issues the usual error message.

Profiles can be used to provide a default value in case the terminal user did not specify a value. These profiles are called the *Default Value* policy profiles. Again, these profiles can be used only or those situations when RACF needs a value, or defaults to a specific value. If the RACF action is to leave an existing value unmodified, the profile is not used. They are used for create or add types of commands. The third qualifier for these profiles starts with /, for example /OWNER. If Mandatory Value policy profiles are present, they pre-empt the Default policy profiles. For example, see the following Default policy profile:

```
C4R.USER./OWNER.IBM*
```

If this profile is present, its APPLDATA value is never used. The Mandatory Value policy profile provides a value, and the Default value is never needed.

The use of Mandatory and Default Value policy profiles can sometimes interfere with the possibility to define or modify multiple profiles in a single command. Most RACF commands allow manipulation of multiple profiles in a single command.

```
ADDUSER (AHJBTST, IBMTEST) OWNER(CMDVFY)
```

By using the same Mandatory Value policy profile, the actual command that is required for the second user ID looks like the following sample:

```
ADDUSER IBMTEST OWNER(SYS1)
```

However, the new value for the OWNER, might be unacceptable for the AHJBTST user ID. In a single RACF command, it is not possible to specify two different OWNERs. Because the conflict cannot be resolved, the entire command is rejected. To avoid these kinds of situations:

- Your installation must specify non-conflicting policies for all profiles that are likely to be handled by a single RACF command. For example, if the Mandatory Value policy profile applied to all user IDs, C4R.USER./OWNER.**, there is no conflict.
- The terminal user must split RACF commands such that they act upon a single profile, or that only profiles with matching policy profiles are grouped into a single RACF command.

The details of the preceding profiles are described in the following sections, together with the verification process for the terminal user specified values of keywords.

SETROPTS-related profiles

The general design of the zSecure Command Verifier policy profiles is centered around the result of the command on a specific profile. However, some commands do not explicitly manage a profile or range of profiles. The most obvious example is the SETROPTS command that acts on RACF settings rather than any profile in the RACF database.

To control the keywords and parameters on the SETROPTS command, a pseudo resource class is used: RACF. The regular profile-related policy profiles consist of four qualifiers, as described in "Policy profile syntax" on page 39.

Because the SETROPTS command has many options, the keywords to manage these options are split into broad categories. The resulting policy profiles have therefore the form:

```
C4R.RACF.category.field.value
```

The following categories are currently used for the RACF options:

LIST
: This category is only used to describe the **SETROPTS LIST** command. Only one profile is implemented in this category.

OPTION
: This category is used for general RACF options, like ERASE, ADDCREATOR and GRPLIST.

AUDIT
This category is used for all audit-related RACF settings, like SAUDIT and CMDVIOL. The LOGOPTIONS setting is not part of this category. Log options are set per class, and are therefore part of the *class* category.

JES
This category is used for JES-related settings.

USER
This category is used for USER and Password options, like the InActive interval and The Password History.

MLS
This category is used for all options that are related to the Implementation of Multi-Level Security.

class
All *class* related settings are categorized per class. This way, one policy profile can be used to control all class-related settings for a particular resource class.

For many options and audit settings, the *value* qualifier in the policy profiles is unused. The seven tables on the following pages summarize per category all RACF options that can currently be controlled by zSecure Command Verifier policy profiles.

One important observation about these policy profiles is the absence of a separate policy profile to describe the REFRESH keyword. The REFRESH keyword is treated as a modifier on the CLASS-related keywords. See the discussion of all CLASS-related profiles.

Because of the complexity of the **SETROPTS** command, the general zSecure Command Verifier policy profiles for error and authority failure suppression are not implemented. If zSecure Command Verifier detects insufficient authorization, it rejects the entire command, regardless of the terminal authority of the user to the C4R.SUPPRESS and C4R.ERROR.CONTINUE profiles.

An example implementation of the SETROPTS-related profiles is shown here:

```
C4R.RACF.AUDIT.**              UACC(NONE) SYSAUDIT(UPDATE)
C4R.RACF.USER.**               UACC(NONE)
C4R.RACF.OPTION.**             UACC(NONE)
C4R.RACF.JES.**                UACC(NONE)
C4R.RACF.XFACILIT.**           UACC(NONE) CMVFYADM(UPDATE)
C4R.RACF.%CICS*.**             UACC(NONE) CICSADM(UPDATE)
C4R.RACF.PROGRAM.*             UACC(NONE) SPROGK(UPDATE)
C4R.RACF.*.RACLIST             UACC(READ)
C4R.RACF.**                    UACC(NONE)
```

Sometimes, it is necessary to give product administrators the System-SPECIAL or System-AUDITOR attribute to be able to fully manage all aspects of the required resource classes. Also, in many organizations, the central user administrator is given System-SPECIAL to manage all profiles for all users and groups. To limit the authority of such people, you can implement profiles like the ones that are shown in the preceding example. Basically, you exclude managing the RACF system-wide settings from their scope of control. The profiles in the preceding example have as a direct effect:

- C4R.RACF.AUDIT.**

  Only people in the SYSAUDIT group can change audit settings. If other people outside that group have the System-Auditor attribute (for instance, because you want them to be able to see various auditing settings), they still cannot modify any of the RACF global audit settings.

- C4R.RACF.USER.**

  Nobody can modify the System Password rules and options.

- C4R.RACF.XFACILIT.**

  Only the zSecure Command Verifier administrators can change the settings of the XFACILIT resource class. This class includes `classact` and `refresh` in-storage profiles.

- C4R.RACF.PROGRAM.*

Only certain people in the Systems Programming department can change the SETROPTS settings for PROGRAM control, including the **REFRESH** of the in-storage profiles.

- C4R.RACF.*.RACLIST

  All System-SPECIAL people, and all others that have sufficient RACF authorization, can **REFRESH** RACLISTed resource classes. RACF permits people with CLAUTH or Group-SPECIAL to **REFRESH** those resource classes.

- C4R.RACF.**

  All remaining SETROPTS keywords and parameters are restricted from all users. If you must modify one of these options, somebody in the CMVFYADM group must define a matching profile, provide access, and issue a REFRESH of the XFACILIT resource class. When you implement SETROPTS controls, you must ensure that at least one person has authority to manage the XFACILIT class.

Table 8 on page 65 shows the policy profile that is used to control the **SETROPTS LIST** command. The information is provided in a separate table so that it can be retrieved more easily.

Table 8. Profiles used for verification of SETROPTS LIST authority. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.

Keyword	Value	Profile
LIST	N/A	C4R.RACF.LIST

Table 9 on page 65 describes all the policy profiles that are used for general RACF Options. These options are set only one time for a certain system, and never changed afterward.

Table 9. Profiles used for verification of RACF options. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.

Keyword	Value	Profile
(NO)ADDCREATOR	N/A	C4R.RACF.OPTION.ADDCREATOR
(NO)ADSP	N/A	C4R.RACF.OPTION.ADSP
CATDSNS	*mode*	C4R.RACF.OPTION.CATDSNS.*mode* *mode* ={ FAILURES, WARNING }
NOCATDSNS	N/A	C4R.RACF.OPTION.CATDSNS.FAILURES C4R.RACF.OPTION.CATDSNS.WARNING
(NO)EGN	N/A	C4R.RACF.OPTION.EGN
ERASE	*type*	C4R.RACF.OPTION.ERASE.*type* *type* = { PROFILE, SECLEVEL, ALL }
(NO)GENERICOWNER ENHANCEDGENERICOWNER	N/A	C4R.RACF.OPTION.GENERICOWNER
(NO)GRPLIST	N/A	C4R.RACF.OPTION.GRPLIST
KERBLVL	*level*	C4R.RACF.OPTION.KERBLVL
PROTECTALL	*mode*	C4R.RACF.OPTION.PROTECTALL.*mode* *mode* = { FAILURES, WARNING }
NOPROTECTALL	N/A	C4R.RACF.OPTION.PROTECTALL.FAILURES C4R.RACF.OPTION.PROTECTALL.WARNING
(NO)REALDSN	N/A	C4R.RACF.OPTION.REALDSN
RETPD	*period*	C4R.RACF.OPTION.RETPD

Table 9. Profiles used for verification of RACF options. The entries in this table reflect the SETROPTS keywords that are used to set a particular option. *(continued)*		
Keyword	**Value**	**Profile**
SESSIONINTERVAL NOSESSIONINTERVAL	*interval* N/A	C4R.RACF.OPTION.SESSIONINTERVAL
(NO)TAPEDSN	N/A	C4R.RACF.OPTION.TAPEDSN
TERMINAL	*access*	C4R.RACF.OPTION.TERMINAL.*access*
RVARYPW	SWITCH(*password*)	C4R.RACF.OPTION.RVARYPW.SWITCH
RVARYPW	STATUS(*password*)	C4R.RACF.OPTION.RVARYPW.STATUS

The following table describes all the policy profiles that are used for the non-class-specific auditing options. These options are already restricted to people with the System-AUDITOR attribute. However, you must define these profiles if you assigned this attribute to people so that they can see the auditing settings.

Table 10. Profiles used for verification of RACF auditing settings. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.		
Keyword	**Value**	**Profile**
(NO)APPLAUDIT	N/A	C4R.RACF.AUDIT.APPLAUDIT
(NO)CMDVIOL	N/A	C4R.RACF.AUDIT.CMDVIOL
(NO)INITSTATS	N/A	C4R.RACF.AUDIT.INITSTATS
(NO)OPERAUDIT	N/A	C4R.RACF.AUDIT.OPERAUDIT
(NO)SAUDIT	N/A	C4R.RACF.AUDIT.SAUDIT
(NO)SECLABELAUDIT	N/A	C4R.RACF.AUDIT.SECLABELAUDIT
SECLEVELAUDIT	*seclevel*	C4R.RACF.AUDIT.SECLEVELAUDIT.*seclevel*
NOSECLEVELAUDIT	N/A	C4R.RACF.AUDIT.SECLEVELAUDIT

The next table describes all the policy profiles that are used for the JES-related setting. Usually, these options are only set one time, and need never be changed.

Table 11. Profiles used for verification of JES-related settings. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.		
Keyword	**Value**	**Profile**
(NO)BATCHALLRACF	N/A	C4R.RACF.JES.BATCHALLRACF
(NO)EARLYVERIFY	N/A	C4R.RACF.JES.EARLYVERIFY
(NO)XBMALLRACF	N/A	C4R.RACF.JES.XBMALLRACF
NJEUSERID	*userid*	C4R.RACF.JES.NJEUSERID.*userid*
UNDEFINEDUSER	*userid*	C4R.RACF.JES.UNDEFINEDUSER.*userid*

The following table describes all the policy profiles that are used for the USER and PASSWORD-related setting.

Table 12. Profiles used for verification of USER-related settings. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.

Keyword	Value	Profile
(NO)INACTIVE	*days*	C4R.RACF.USER.INACTIVE
PASSWORD	ALGORITHM(KDFAES) NOALGORITHM	C4R.RACF.USER.PASSWORD.ALGORITHM
PASSWORD	HISTORY(*count*)	C4R.RACF.USER.PASSWORD.HISTORY
PASSWORD	INTERVAL(*period*)	C4R.RACF.USER.PASSWORD.INTERVAL
PASSWORD	MINCHANGE(*period*)	C4R.RACF.USER.PASSWORD.MINCHANGE
PASSWORD	(NO)MIXEDCASE	C4R.RACF.USER.PASSWORD.MIXEDCASE
PASSWORD	REVOKE(*count*)	C4R.RACF.USER.PASSWORD.REVOKE
PASSWORD	RULEn(*rule-spec*) NORULEn NORULES	C4R.RACF.USER.PASSWORD.RULES
PASSWORD	(NO)SPECIALCHARS	C4R.RACF.USER.PASSWORD.SPECIALCHARS
PASSWORD	WARNING(*period*)	C4R.RACF.USER.PASSWORD.WARNING

The next table describes all the policy profiles that are used for control of the Multi-Level Security-related settings. Unless you are implementing Multi-Level Security, these options must not be modified.

Table 13. Profiles used for verification of MLS-related settings. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.

Keyword	Value	Profile
(NO)COMPATMODE	N/A	C4R.RACF.MLS.COMPATMODE
MLACTIVE	*mode*	C4R.RACF.MLS.MLACTIVE.*mode* *mode* = {FAILURES, WARNING }
NOMLACTIVE	N/A	C4R.RACF.MLS.MLACTIVE.FAILURES C4R.RACF.MLS.MLACTIVE.WARNING
MLS	*mode*	C4R.RACF.MLS.MLS.*mode* *mode* = { FAILURES, WARNING }
NOMLS	N/A	C4R.RACF.MLS..FAILURES C4R.RACF.MLS.WARNING
(NO)MLSTABLE	N/A	C4R.RACF.MLS.MLSTABLE
MLFSOBJ	*mode*	C4R.RACF.MLS.MLFSOBJ
MLIPCOBJ	*mode*	C4R.RACF.MLS.MLIPCOBJ
(NO)MLNAMES	N/A	C4R.RACF.MLS.MLNAMES
(NO)MLQUIET	N/A	C4R.RACF.MLS.MLQUIET
(NO)SECLABEL CONTROL	N/A	C4R.RACF.MLS.SECLABELCONTROL
(NO)SECLBYSYSTEM	N/A	C4R.RACF.MLS.SECLBYSYSTEM

The following table describes all the policy profiles that are used for the class-specific options. Usually these options are set frequently by many different people. The policy profiles in this category also describe the authorization to REFRESH in-storage profiles.

Table 14. Profiles used for verification of class-specific settings. The entries in this table reflect the SETROPTS keywords that are used to set a particular option.

Keyword	Value	Profile
(NO)AUDIT	*class*	C4R.RACF.*class*.AUDIT
(NO)CLASSACT	*class*	C4R.RACF.*class*.CLASSACT
(NO)GENCMD	*class*	C4R.RACF.*class*.GENCMD
(NO)GENERIC	*class*	C4R.RACF.*class*.GENERIC
(NO)GENLIST	*class*	C4R.RACF.*class*.GENLIST
(NO)GLOBAL	*class*	C4R.RACF.*class*.GLOBAL
(NO)RACLIST	*class*	C4R.RACF.*class*.RACLIST
(NO)STATISTICS	*class*	C4R.RACF.*class*.STATISTICS
(NO)WHEN	*class*	C4R.RACF.*class*.WHEN
LOGOPTIONS	*condition(class)*	C4R.RACF.*class*.LOGOPTIONS.*condition* *condition* = { ALWAYS, NEVER, SUCCESSES, FAILURES, DEFAULT }

The following list describes the profiles in detail, and shows the required access.

- C4R.RACF.LIST

  Specifies the authority to issue the **SETROPTS LIST** command. The following access rules apply.

  No profile found
  Only the regular RACF authority must be used to determine the authority of the terminal user to **LIST** the current RACF settings.

  NONE
  The terminal user is not authorized to **LIST** the current RACF settings.

  READ
  If the terminal user has sufficient RACF authorization, the current RACF settings can be listed.

  UPDATE
  Same as READ.

  CONTROL
  Same as READ.

- C4R.RACF.*category.keywords.values*

  The access requirements for most policy profiles are as follows. See the following descriptions for more notes about the use of the policy profiles.

  No profile found
  This control is not implemented. Only the regular RACF authority must be used.

  NONE
  The terminal user is not authorized to modify the RACF setting in hand.

  READ
  Same as NONE.

  UPDATE
  If the terminal user has sufficient RACF authorization, the RACF setting can be modified.

CONTROL
Same as UPDATE.

- C4R.RACF.OPTION.CATDSNS.*mode*

  Specifies the authority to modify the settings for the CATDSNS option. If the CATDSDN option is used without a *mode* parameter, RACF defaults to FAILURES mode. If the NOCATDSNS option is used, zSecure Command Verifier does not check for the current *mode* but requires access to both *modes*. For most environments, use generics (".**") for the last qualifier (*mode*).

- C4R.RACF.OPTION.ERASE.*mode*

  Specifies the authority to modify the options for the ERASE on scratch setting. If ERASE is used without a subparameter, RACF uses the ERASE settings for individual data set profiles. In zSecure Command Verifier this case is described by the *mode* PROFILE. This mode is also used for the NOERASE setting. The other ERASE settings are described by *modes* SECLEVEL and ALL. The SECLEVEL policy profile does not include the actual *seclevel* specified in the command. It also describes the use of the NOSECLEVEL option. For most situations, use generics (".**") for the last qualifier.

- C4R.RACF.OPTION.GENERICOWNER

  This policy profile is used to control setting and removing the GENERICOWNER and ENHANCEDGENERICOWNER options.

- C4R.RACF.OPTION.KERBLVL

  The actual *level* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.OPTION.PROTECTALL.*mode*

  Specifies the authority to modify the settings for the PROTECTALL option. If the PROTECTALL option is used without a *mode* parameter, RACF defaults to FAILURES mode. If the NOPROTECTALL option is used, zSecure Command Verifier does not check for the current *mode* but requires access to both *modes*. For most environments, use generics (".**") for the last qualifier (*mode*).

- C4R.RACF.OPTION.RETPD

  The actual default retention period that is specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.OPTION.SESSIONINTERVAL

  Specifies the authority to modify the settings for the SESSIONINTERVAL option. This profile is used both for the NOSESSIONINTERVAL and the SESSIONINTERVAL setting.

  The actual session *interval* specified in the command, is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.OPTION.RVARYPW.*action*

  This policy profile describes the authority to set the RVARY passwords. RACF supports a separate password for both the SWITCH and STATUS *action*s. The actual RVARY passwords that are specified in the command are not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.AUDIT.SECLEVELAUDIT.*level*

  Specifies the authority to modify the settings for the SECLEVELAUDIT option. When you set the preceding SECLEVEL which auditing must be done, the *level* is included as the last qualifier of the zSecure Command Verifier policy profile. When you disable SECLEVELAUDIT, this *level* qualifier is not used. For most environments, use generics (".**") for this last qualifier (*level*).

- C4R.RACF.USER.INACTIVE

  The actual INACTIVE *days* specified in the command are not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.ALGORITHM

  This policy profile controls selecting the password encryption algorithm. The name of the selected ALGORITHM (KDFAES) is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.HISTORY

  The actual HISTORY *count* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.INTERVAL

  The actual INTERVAL *period* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.MINCHNAGE

  The actual MINCHANGE *period* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.MIXEDCASE

  This policy profile controls the setting for the `mixedcase` option for user passwords.

- C4R.RACF.USER.PASSWORD.REVOKE

  The actual REVOKE *count* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.USER.PASSWORD.RULES

  This single policy profile is used to describe all changes to any RACF password rule. The policy profile is also used when you disable any or all password rules. The current version of zSecure Command Verifier does not provide support for the actual password rule content.

- C4R.RACF.USER.PASSWORD.SPECIALCHARS

  This policy profile controls setting the option to allow additional special characters in user passwords.

- C4R.RACF.USER.PASSWORD.WARNING

  The actual WARNING *period* specified in the command is not represented in the zSecure Command Verifier policy profile.

- C4R.RACF.MLS.MLACTIVE.*mode*

  Specifies the authority to modify the settings for the MLACTIVE option. If the MLACTIVE option is used without a *mode* parameter, RACF defaults to WARNING mode. If the NOMLACTIVE option is used, zSecure Command Verifier does not check for the current *mode* but requires access to both *modes*. For most environments, use generics (".**") for the last qualifier (*mode*).

- C4R.RACF.MLS.MLS.*mode*

  Specifies the authority to modify the settings for the MLS option. If the MLS option is used without a *mode*parameter, RACF defaults to WARNING mode. If the NOMLS option is used, zSecure Command Verifier does not check for the current *mode* but requires access to both *modes*. For most environments, use generics (".**") for the last qualifier (*mode*).

- C4R.RACF.MLS.MLFSOBJ

  Specifies the authority to modify the mode for MLFSOBJ processing. Both modes (ACTIVE and INACTIVE) are described by the same zSecure Command Verifier policy profile.

- C4R.RACF.MLS.MLIPCOBJ

  Specifies the authority to modify the mode for MLIPCOBJ processing. Both modes (ACTIVE and INACTIVE) are described by the same zSecure Command Verifier policy profile.

- C4R.RACF.*class.function*

  These profiles are used to describe the authority to activate and deactivate class-related options, and to REFRESH in-storage profile. The *function* can be any of the functions that are shown in the preceding table. WHEN applies only to the PROGRAM class.

  The access requirements for these policy profiles differ from the access requirements for most other policy profiles. The READ access level is significant and provides the authority to REFRESH in-storage profiles. It is only used for the listed *function*s.

Often, you can use generics (`.**`) for the last qualifier. For the LOGOPTIONS, this qualifier includes the *condition* when audit records are created. The extra qualifier allows easier delegation to the designated people.

No profile found
This control is not implemented. Only the regular RACF authority must be used.

NONE
The terminal user is not authorized to activate, deactivate, or refresh *function* for the *class*.

READ
The terminal user is authorized to REFRESH in-storage profiles for the *class*. This case applies to the GENERIC, GENLIST, GLOBAL, RACLIST, and WHEN *function*s. For all other *function*s, this access level has the same effect as access NONE. This access level does not permit use of any of the listed *function*s without the REFRESH keyword.

UPDATE
The terminal user is authorized for the *function* for the *class*. This setting applies only if the user has sufficient RACF authorization to perform the *function*.

CONTROL
Same as UPDATE.

Profiles for managing user IDs

The profiles in the following sections are used to manage commands that are related to user IDs.

All the possible keywords and the corresponding profiles are split in several categories. The first group of profiles describes naming conventions for a new `userid` and the place in the RACF group hierarchy for new or existing `userids`. Subsequent sections describe the Connections of users to groups and the attributes and authorizations of users.

Conventions for naming user IDs

Many installations have user ID naming conventions to indicate to which department an ID belongs. zSecure Command Verifier implements several of these naming conventions. These rules are only applied to the ADDUSER command for creating new User profiles.

Table 15 on page 71 summarizes the profiles that control the `userid` itself. Table 16 on page 75 and Table 17 on page 75 describe mandatory values and default values for some keywords. Table 18 on page 75 describes the profiles for verifying the values that are specified by the terminal user.

Table 15. Profiles used for verification of the RACF user ID. The entries in this table reflect the keywords that describe the name of new and deleted USERIDs.

Command	Keyword	Profile
ADDUSER	*userid*	C4R.USER.ID.=RACUID(n)
ADDUSER	*userid*	C4R.USER.ID.=RACGPID(n)
ADDUSER	*userid*	C4R.USER.ID.*userid*
DELUSER	*userid*	C4R.USER.DELETE.*userid*

The profiles in this table describe new userids that are created. For the `userid` itself, zSecure Command Verifier provides controls to enforce the naming conventions. The authority to change existing user IDs is not controlled by naming conventions. This authorization is already sufficiently restricted by the normal RACF scoping rules. The authority to delete users is also controlled by the normal RACF ownership rules; however, an extra control is needed. Therefore, another name-based rule is used to implement this control. To define new `userids`, the terminal user still needs CLAUTH(USER) plus at least one group-related authorization like JOIN, the group-SPECIAL attribute, or direct ownership.

The user ID-based controls enforce naming conventions for new IDs. This first set of profiles controls the userid for the user. These profiles are intended to specify which userids can be defined. In general, only one of these profiles is used to specify your naming convention. More generic profiles must be used to block the definition of new userids that do not follow your naming convention. Exceptions can be implemented by the definition of more specific discrete or generic profiles. The following example shows the implementation of these profiles.

```
C4R.USER.ID.=RACUID(4)        UACC(UPDATE)
C4R.USER.ID.TEST*             UACC(NONE) IBMUSER(UPDATE)
C4R.USER.ID.*                 UACC(NONE)
```

These profiles ensure that no new userids can be defined unless the first 4 characters of the new userid are the same as the first 4 characters of the terminal user who defines the ID. An exception is userids that start with TEST. These user IDs can be defined by the terminal user IBMUSER, and also, according to the first profile, by all terminal users that have a userid starting with TEST. The third profile is required to stop definition of new userids outside the specified naming convention. Without the third profile, almost any userid is accepted, either explicitly by the first or second profile or implicitly by the absence of a matching profile.

- C4R.USER.ID.=RACUID(n)

  Specifies a special generic policy for the new userid. The =RACUID stands for the userid of the terminal user. If the substring(=RACUID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the userids.

  This profile is a discrete profile. Only the single digit between parenthesis is variable. It must be specified as a value in the range 1-8. It is not possible to use a true generic profile.

  The following access rules apply.

  No profile found
  The userid of the terminal user is not used as naming convention for new userids. Verification continues with the =RACGPID*(n)* profile.

  NONE
  The new userid is not allowed. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The new userid is accepted.

  CONTROL
  Same as UPDATE.

- C4R.USER.ID.=RACGPID(n)

  Specifies a special generic policy for the new userid. The =RACGPID stands for the list of groups to which the terminal user is connected. All the groups of the user are used, independent of the list of group access checking setting. This profile is used only if =RACUID(*n*) profile is not present or does not match. If the substring(=RACGPID,1,*n*) matches, this profile is used in preference to other profiles described in the following paragraph, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the userids.

  This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The groups of the terminal user are not used as naming convention for new userids. Verification continues with profile C4R.USER.ID.userid.

  NONE
  The new userid is not allowed. The command is rejected.

READ
Same as NONE.

UPDATE
The new userid is accepted.

CONTROL
Same as UPDATE.

- C4R.USER.ID.*userid*

  Specifies which new *userid*s can be created by the terminal user. This profile is only used for the ADDUSER command if both =RACUID(*n*) and =RACGPID(*n*) are absent or do not match. This rule can be covered by a generic profile.

  No profile found
  No naming convention is enforced for new user IDs.

  NONE
  The specified userid is not allowed. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  Permission to create the specified userid.

  CONTROL
  Same as UPDATE.

Deletion of existing users

Use the **C4R.USER.DELETE.***userid* profile described in this topic to control the authority to delete existing user IDs.

The authority to delete User profiles is normally controlled by some form of ownership (either direct or within the scope of a group-SPECIAL attribute) and by system-SPECIAL authorization. Some organizations want to keep strict control over the authority to delete existing users. Most often, it is because these organizations implemented extra procedures, like saving or renaming data sets or interaction with non-RACF information. The policy profile that is described in this section puts more constraints on the authorization to delete user IDs. This profile is not verified if RACF already rejected deletion of the group because of syntax errors or insufficient authority.

Deleting user IDs can also be controlled through the =NOCHANGE policy for user IDs. If a DELETE policy allows deleting the ID, the =NOCHANGE policy profile can still reject the command.

- C4R.USER.DELETE.*userid*

  This profile can be used to control which user ID in scope can be deleted. When using generic profiles, deletion of user IDs can also be completely prevented. Only the terminal users who have access through this profile are allowed to delete these user IDs. This control reduces the normal RACF delete authorization.

  No profile found
  The control is not implemented. No additional restrictions on deleting the specified userid.

  NONE
  The userid cannot be deleted. The command is rejected.

  READ
  The userid can be deleted only if the terminal user has the system special attribute.

  UPDATE
  The userid can be deleted.

  CONTROL.
  Same as UPDATE.

Prevent all actions against user profiles

Use the C4R.USER.=NOCHANGE.*owner.userid* profile described in this topic to prevent all changes or actions against a user ID.

Aside from preventing the deletion of user profiles as described in the previous section, zSecure Command Verifier also provides an option to completely prevent any action against a selected user or range of users. This can be done through the definition of a =NOCHANGE policy profile for the user ID. When a =NOCHANGE policy profile has been defined, the following changes are prevented unless the terminal user has sufficient access:

- Changing the attributes of the user ID.
- Setting or changing the password, phrase, or interval for the user ID.
- Connecting or removing the user to or from any group.
- Deleting the user ID.
- Granting or removing direct access of the user ID.

These changes can also be controlled through individual policy profiles. The advantage of the =NOCHANGE policy profile is that a single policy profile can be used to control all actions related to the user ID. The =NOCHANGE policy profile can be used to effectively lock or freeze the current definition of the user ID. The qualifier =NOCHANGE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The =NOCHANGE policy profile for users has the following format:

- C4R.USER.=NOCHANGE.*owner.userid*

  This policy profile can be used to prevent all changes or actions against a user ID. Actions that are controlled are changing or deleting the user ID, connecting the ID to or removing the ID from a group, and granting or removing direct access of the user ID to a data set or general resource. If the policy profile has been defined, only terminal users who have access through this profile are allowed to change these user IDs. The following access levels are supported:

  No Profile Found
  This control is not implemented. Modification of the target user ID is not prevented.

  NONE
  The terminal user is not authorized to perform any actions against the target user ID.

  READ
  Same as NONE.

  UPDATE
  The terminal user can modify the target user ID, provided that the target user ID is within the regular RACF scope of the terminal user.

  CONTROL
  Same as UPDATE.

Placement of new IDs in the RACF group hierarchy

When a userid is created according to the preceding profiles, additional rules can apply to the placement of the new ID in the RACF Group hierarchy.

zSecure Command Verifier provides the following types of profiles to control this aspect:

- The mandatory value profiles enforce a specific owner and default group for the new `userid`.
- The default profiles provide default values if the terminal user does not specify a value.
- The last set of profiles verifies that the values that the terminal user specifies are acceptable.

The following information describes how these profiles are used together and which keywords can be suppressed or added.

For Mandatory Value profiles, the third qualifier consists of an equals sign (=), followed by the keyword. So for the DFLTGRP, the profile has the qualifier =DFLTGRP. Table 16 on page 75, describes the Mandatory Value profiles.

Table 16. Mandatory Value policy profiles for RACF user ID place-related command/keywords. The entries in this table reflect the keywords that describe the Mandatory Value place of new USERIDs.

Command	Keyword	Profile
ADDUSER	*userid*	C4R.USER.=DFLTGRP.*userid*
ADDUSER	*userid*	C4R.USER.=OWNER.*userid*

Table 17 on page 75 describes the Default profiles that are used if the terminal user did not specify any keywords that control the place in the RACF Group hierarchy. For Default profiles, the third qualifier consists of a forward slash, followed by the keyword. So for the DFLTGRP, the policy profile has /DFLTGRP.

Table 17. Profiles used for Default values of RACF user ID place-related command/keywords. The entries in this table reflect the default values for keywords that describe the Default Place of new USERIDs.

Command	Keyword	Profile
ADDUSER	*userid*	C4R.USER./DFLTGRP.*userid*
ADDUSER	*userid*	C4R.USER./OWNER.*userid*

Table 18 on page 75 describes the profiles that are used to verify acceptability of the terminal user-specified values. The table summarizes which profile is used for which keyword or function.

Table 18. Profiles used for verification of the RACF user ID. The entries in this table reflect the keywords that are specified by the terminal user to describe the name and place of new or changed user IDs.

Command	Keyword	Profile
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP.=RACUID(n)
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP.=RACGPID(n)
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP.=USERID(n)
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP.*group.userid*
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP./SCOPE.*group.userid*
ADDUSER ALTUSER	DFLTGRP	C4R.USER.DFLTGRP./OWNER.*group.userid*
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER.=RACUID(n)
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER.=RACGPID(n)
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER.=USERID(n)
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER.*owner.userid*
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER./SCOPE.*owner.userid*
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER./GROUP.*owner.userid*
ADDUSER ALTUSER	OWNER	C4R.USER.OWNER./DFLTGRP.*owner.userid*

Policy profiles selection for the default group

Use these guidelines to implement policy profiles for the default group.

Aside from the name of a new user ID, there are two other important aspects for defining new users or changing existing users:

- The place of the ID in the RACF hierarchy (the OWNER).
- The default group of the ID (DFLTGRP).

The default group as such is not exceptional in any way. It is only important when you define a user because it controls the authorization to create the user. In RACF, the terminal user must either have JOIN authority in that group, the group must be within the scope of a group-SPECIAL attribute, or the terminal user must own the group. zSecure Command Verifier implements some additional controls on the default group. To define new User profiles, the terminal user also needs system special or CLAUTH in the User class. The next paragraphs describe how the zSecure Command Verifier profiles from the preceding tables are used.

The first set of profiles controls the default group DFLTGRP of the new userid for the **ADDUSER** command. zSecure Command Verifier does not use the Mandatory or Default Value profiles for the OWNER and DFLTGRP on the **ALTUSER** command. Because the **ALTUSER** command does not force these existing values to change, it is not necessary enforce a specific value.

When you define a new user profile, zSecure Command Verifier also verifies the authorization to CONNECT the new user to the specified DFLTGRP. The specification of a GROUP as DFLTGRP during the creation of a new user results in an automatic CONNECT of the userid to the GROUP. The required authorization is verified independently. See "CONNECT management" on page 130 for details.

Mandatory and default value policy profiles for the DFLTGRP

Use these policy profiles to specify the mandatory and default values for the DFLTGRP of a new userid. These profiles are only used for the **ADDUSER** command.

- C4R.USER.=DFLTGRP.*userid*

  This profile is used to specify a mandatory value for the DFLTGRP of every newly defined userid. It is only used for the ADDUSER command. The DFLTGRP that is used, is obtained from the APPLDATA field in the profile. This value is used to override any terminal user specified value, or added to the command if the terminal user did not specify a value. The DFLTGRP value that is obtained by this Mandatory Value profile is not subject to other DFLTGRP-related policy profiles.

  The value *userid* represents the affected user. This value allows the specification of exceptions to the general rule. Only the most specific profile is used by zSecure Command Verifier. Generic profiles can be used to specify the DFLTGRP for users.

  The qualifier =DFLTGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  The control is not active for the terminal user. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command. If this process does not yield a valid group, the current connect group of the terminal user is used instead.

  UPDATE
  Same as READ.

  CONTROL
  The control is not active for the terminal user. No mandatory value is enforced. If the terminal user specified a value for the group, it is used. If no value was specified, RACF uses the current group of the terminal user.

  Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. However, access NONE indicates that the facility as described by the policy is unavailable to the terminal user. For the Mandatory Value profiles, the odd situation then occurs that access NONE has the same net result as access CONTROL.

The values that are accepted for the **APPLDATA** field are shown in the following list. The terminal user still needs sufficient authority in the assigned DFLTGRP to define new users. This authorization is not verified in zSecure Command Verifier. Insufficient authority can result in failure of the command by RACF.

BLANK
Indicates that RACF default processing must be used. That is, RACF uses the current group of the terminal user.

userid
This entry is not valid. Because this entry is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current group of the terminal user.

group
This group is inserted. If the terminal user does not have sufficient access to this group, the command is rejected by RACF.

=OWNER
Reflects the OWNER as specified (or defaulted) by the OWNER keyword on the command. This value might also be an OWNER value as inserted by zSecure Command Verifier. If the OWNER resolves to the special value =DFLTGRP (indicating the default group), the command is rejected.

=MYOWNER
Reflects the OWNER of the terminal user. This value must be a group. All other situations are considered an error. Because this case is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current group of the terminal user.

=USERID(n)
Reflects the first *n* characters of the new USERID itself. This value must be a GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to allow definition of the `userid` through =RACGPID(*n*) in C4R.USER.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to permit definition. In all other situations, the APPLDATA value =RACGPID is considered an error, and the current group of the terminal user is used instead.

After zSecure Command Verifier processes this profile and determines the mandatory value for the DFLTGRP, it verifies the authorizations for the specified connection. See “CONNECT management” on page 130 for all user-to-group connections.

- C4R.USER./DFLTGRP.*userid*

This profile is used to specify a default value for the DFLTGRP in case the terminal user did not specify a DFLTGRP on the ADDUSER command. If the preceding Mandatory Value policy profile is used to provide a value, the /DFLTGRP profile is not used.

The DFLTGRP that is used, is obtained from the **APPLDATA** field in the profile. The DFLTGRP value that is obtained by this Default Value profile is not subject to other DFLTGRP-related policy profiles.

The qualifier /DFLTGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied. Using the default value that is normally provided by RACF is also not acceptable and the command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the DFLTGRP.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied. The current group of the terminal user is used by RACF.

The values that are accepted for the **APPLDATA** field are shown in the following list. The terminal user still needs sufficient authority in the assigned DFLTGRP to define new users. Insufficient authority can result in failure of the command.

BLANK
Indicates that RACF default processing must be used. The current group of the terminal user is used.

userid
This entry is not valid. Because this case is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current group of the terminal user.

group
The group is inserted.

=OWNER
Reflects the OWNER as specified (or defaulted) by the OWNER keyword on the command. This value can be an OWNER value as inserted by zSecure Command Verifier. If the OWNER resolves to the special value =DFLTGRP (indicating the default group), the command is rejected.

=MYOWNER
Reflects the owner of the terminal user and this value must be a group. All other situations are considered an error. Because this error is not caused by incorrect entry by the terminal user, the command is permitted to continue by using the current group of the terminal user.

=USERID(n)
Reflects the first *n* characters of the new USERID itself. This value must be a group. All other situations are considered an error, and the current group of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to permit definition of the `userid` through `=RACGPID(n)` in C4R.USER.ID.=RACGPID(n). This value is only used if `=RACGPID(n)` was used to permit definition. In all other situations, the APPLDATA value =RACGPID is considered an error, and the current group of the terminal user is used instead.

After zSecure Command Verifier processes this profile and determines the Default value for the DFLTGRP, it verifies the authorizations for the specified connection. See "CONNECT management" on page 130 for all user-to-group connections.

Verification of the default group

The profiles in this topic are used to control the selection of the default group for new and existing users.

These profiles are used to verify the specification of the DFLTGRP by the terminal user. Restrictions on the selection of a default group through the **ALTUSER** command, are probably not relevant to most RACF processing. The user can still select any of its groups as the current group during logon processing. Only the specification of the default value is controlled by the **ALTUSER** RACF command.

When you add a user to the system through the **ADDUSER** command, a second check is performed for the DFLTGRP. The selection of a DFLTGRP has an immediate result that the new user is also connected to the specified group. Therefore, the authorization to connect the user to the specified group is also verified. The same applies for the group-authorizations. For more information about the user-to-group connections and authorization, see "CONNECT management" on page 130.

- `C4R.USER.DFLTGRP.=RACUID(n)`

  Specifies a special generic policy for the DFLTGRP in ADDUSER and ALTUSER commands. The =RACUID stands for the USERID of the terminal user. If the `substring(=RACUID,1,n)` matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the user ID values.

This profile is a discrete profile. Only the single digit between parenthesis is variable, and must be specified from 1 to 8. It is not possible to use a true generic profile.

No profile found
The user ID of the terminal user is not used as naming convention or restriction for the DFLTGRP.

NONE
The specified DFLTGRP is not allowed. This decision can be overruled by authorization to profile *group.userid* (see C4R.USER.DFLTGRP.*group.userid*).

READ
Same as NONE.

UPDATE
The specified DFLTGRP is accepted.

CONTROL
Same as UPDATE

- C4R.USER.DFLTGRP.=RACGPID(*n*)

Specifies a special generic policy for the DFLTGRP in ADDUSER and ALTUSER commands. The =RACGPID stands for the list of groups to which the terminal user is connected. All the user groups are used, independent of the setting of "list of group access checking". This profile is used only if the preceding =RACUID(*n*) profile is not present or does not match. If the substring(=RACGPID,1,*n*) matches, this profile is used in preference to other profiles described later in the list, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the USERID values.

This profile is a discrete profile. Only the single digit between parenthesis is variable, and must be specified from 1 to 8. It is not possible to use a true generic profile.

No profile found
The current group of the terminal user is not used as naming convention or restriction for the DFLTGRP.

NONE
The specified DFLTGRP is not allowed. This decision can be overruled by authorization to profile *group.userid* (see C4R.USER.DFLTGRP.*group.userid*).

READ
Same as NONE.

UPDATE
The specified DFLTGRP is accepted.

CONTROL
Same as UPDATE

- C4R.USER.DFLTGRP.=USERID(*n*)

Specifies a special generic policy for the DFLTGRP in ADDUSER and ALTUSER commands. The =USERID stands for the user ID that is being defined or changed. If the substring(=USERID,1,*n*) matches, this profile is used in preference to other generic profiles, independent of the value of *n*. This profile is used only if =RACUID(*n*) and =RACGPID(*n*) are not present or do not match.

This profile is a discrete profile. Only the single digit between parenthesis is variable. It must be specified as a value in the range 1-8. It is not possible to use a true generic profile.

No profile found
The first *n* characters of userid are not used as a restriction on the DFLTGRP for the user.

NONE
The specified DFLTGRP is not allowed. This decision can be overruled by authorization to profile *group.userid* (see C4R.USER.DFLTGRP.*group.userid*).

READ
Same as NONE.

UPDATE
The specified DFLTGRP is accepted.

CONTROL
Same as UPDATE

If any of the preceding three profiles allow the selected DFLTGRP, the next profile is skipped. Processing continues with the /SCOPE and /OWNER policies that are described in "Additional policy controls for the default group" on page 80. If the preceding profiles did not authorize the use of a certain DFLTGRP, the next profile is used as alternative authorization method.

- C4R.USER.DFLTGRP.*group.userid*

  This profile is used independently of the three rules that are defined earlier. It can be used to specify exceptions to the generic name-based policies. It controls whether *group* can be used as DFLTGRP for the new *userid*. For existing IDs, the profile specifies which of the groups for the user can be selected as the DFLTGRP on the ALTUSER command.

  In most situations, you specify *userid* through a generic. Explicit profiles can be used to define exceptions for certain userids.

  This profile is not used if any of the previous three profiles already allowed the use of the specified DFLTGRP.

  No profile found
  The control is not implemented. No name-based policy is enforced.

  NONE
  The specified DFLTGRP is not allowed.

  READ
  Same as NONE.

  UPDATE
  The groupname can be used.

  CONTROL
  Same as UPDATE.

Additional policy controls for the default group

The next profiles are used to define general restrictions on the default group (DFLTGRP):

1. The first one (C4R.USER.DFLTGRP./SCOPE.*group.userid*) restricts DFLTGRP to be within the scope of a group-SPECIAL attribute. It effectively disables JOIN authorization and direct ownership of a GROUP as a means to permit creation of new User profiles. As normal users usually do not have group-SPECIAL, all changes to the DFLTGRP are considered outside their scope. This profile also effectively disallows normal users to change their DFLTGRP. Each user can still specify any of its groups as the current group during the logon process.
2. The second profile (C4R.USER.DFLTGRP./OWNER.*group.userid*) compares the DFLTGRP against the OWNER of the USERID. It can be used to enforce a match, but it also allows exceptions to this general rule.

- C4R.USER.DFLTGRP./SCOPE.*group.userid*

  This profile is used to specify that the default group of new users must be within the scope of group-SPECIAL. It also controls which of the existing groups can be selected as the default group. The main purpose of this profile is to prevent decentralized administrators from changing the DFLTGRP to a group that they do not control.

  The variables *userid* and *group* represent the affected User profile and its new DFLTGRP. This enables specification of exceptions to the general rule. zSecure Command Verifier uses the most specific profile.

  The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

For terminal users with group-special or system-special, specification of a new default group for the user is recorded through the audit-only policy profile

– C4R.USESCOPE.*group*

Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the specified default group for the user. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

The following access levels are supported for the /SCOPE policy profile:

No profile found
The control is not implemented.

NONE
Only groups within the scope of the terminal user can be specified as DFLTGRP on both the **ADDUSER** and **ALTUSER** commands. If any other GROUP is specified, the command is rejected.

READ
Same as NONE.

UPDATE
Groups outside the scope of the terminal user can be used on both the **ADDUSER** and **ALTUSER** command. If the terminal user does not have sufficient authority in the specified group, the command is rejected by RACF.

CONTROL
This policy is not in effect for the terminal user.

- C4R.USER.DFLTGRP./OWNER.*group.userid*

Specifies that the DFLTGRP of new users must be the same as the OWNER of the userid. Users need access to this profile to specify anything but the owner as the value for the DFLTGRP.

For existing users, it restricts the selection of the DFLTGRP through the **ALTUSER** command to the group that is the OWNER of the user profile. If the OWNER is changed concurrently in the same **ALTUSER** command, the new DFLTGRP is verified against the new OWNER.

For new userids, the use of C4R.USER.=DFLTGRP.*userid*, described previously, is preferred. This Mandatory Value policy profile overlays any value that is specified by the terminal user. The current /OWNER profile requires the terminal user to specify the correct value. If the Mandatory Value policy profile is used, the current profile is skipped. The main purpose of the /OWNER profile is to permit certain users to be exempt from the DFLTGRP=OWNER requirement.

The variables *userid* and *group* represent the affected User profile and its new DFLTGRP. These variables permit specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented.

NONE
The DFLTGRP for the user must be the same as the OWNER of the user ID.

READ
Same as NONE

UPDATE
The terminal user is authorized to specify a value for the DFLTGRP that is different from the current or new OWNER of the user ID.

CONTROL
This policy is not in effect for the terminal user.

Policy profiles for the owner

The other piece of information that describes a newly defined user ID is the OWNER. The profiles in this section are used to control the specification of the owner.

These profiles apply both to the **ADDUSER** and **ALTUSER** commands. In general, the processing for these profiles assumes that the policy of your installation is to use GROUPs as OWNER. The last profile that is described in "Mandatory and default value profiles for the OWNER" on page 82 /GROUP provides a control that can be used to indicate whether your installation wants to enforce such a policy or not. Again, the description is split into several sets of profiles. The first specifies a mandatory or default value for the owner. The second set of profiles describes controls on a specified value for the owner. The final set of three profiles describes general policies that can be used for the OWNER of user IDs.

Mandatory and default value profiles for the OWNER

Use these policy profiles to specify the mandatory and default values for the OWNER of a new user ID. These profiles are used only for the **ADDUSER** command.

- C4R.USER.=OWNER.*userid*

  This profile is used to specify a mandatory (overriding) value for the OWNER of the newly defined user ID. It is only used during **ADDUSER** processing. The OWNER value that is obtained from this Mandatory Value profile is not subject to more OWNER-related policy profiles.

  The qualifier =OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  No action. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command. If the process yields an ID that is not valid, or a non-existing entry, the current group of the terminal user is used instead.

  UPDATE
  Same as READ

  CONTROL
  The control is not active for the terminal user. No mandatory value is supplied. The value for the OWNER as specified by the terminal user is used in the command.

  Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. However, access NONE indicates that the facility as described by the policy is unavailable to the terminal user. For the Mandatory Value profiles, the odd situation can then occur that access NONE has the same net result as access CONTROL.

  The values that are accepted for the **APPLDATA** field are given as follows. The OWNER can be a user ID or GROUP.

  BLANK
  The specified value of the new OWNER is suppressed, and replaced by the user ID of the terminal user. This value is the default value that RACF uses if no OWNER was specified. Depending on the access level to the /GROUP profile, zSecure Command Verifier allows use of the terminal user as the new OWNER.

  userid
  Depending on the access level to the /GROUP profile, the *userid* is inserted as the owner of the new user ID.

  group
  The specified GROUP is used as OWNER of the new user ID.

=DFLTGRP
Represents the default group DFLTGRP as specified or defaulted on the command. If this value resolves to the special value =OWNER, which represents the OWNER that is being determined, the command fails.

=MYOWNER
Reflects the OWNER of the terminal user. If this value is a GROUP, the value is used as the OWNER of the new user ID. If this value is a user ID, further processing is dependent on the access level that the terminal user must the /GROUP profile.

=USERID(n)
Reflects the first *n* characters of the new user ID itself. This value must be a user ID or GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to allow definition of the user ID in C4R.USER.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) is used to allow definition. In all other situations, the value =RACGPID is considered an error, and the current GROUP of the terminal user is used instead.

- C4R.USER./OWNER.*userid*

This profile is used to specify a default value for the OWNER of the newly defined user ID profile. It is only used during **ADDUSER** processing. The OWNER that is to be used as the default value is obtained from the **APPLDATA** field in the profile. The OWNER value that is obtained through this Default Value profile is not subject to more OWNER-related policy profiles. If the preceding =OWNER profile is used to provide a value, the /OWNER profile is not used.

The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied. This case results in RACF providing a default for the OWNER =the terminal user itself.

NONE
No action. No default value is supplied. zSecure Command Verifier does not allow RACF to provide a value for the OWNER. The command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the OWNER.

READ
The **APPLDATA** field is extracted and used for the command. If the process yields an ID that is not valid, or a non-existing entry, the current group of the terminal user is used instead.

UPDATE
Same as READ

CONTROL
The control is not active for the terminal user. No default value is supplied. Because the terminal user did not specify a value for the OWNER, RACF makes the terminal user the OWNER of the new profile.

The values that are accepted for the **APPLDATA** field are given in the following list. The specified OWNER can be a user ID or GROUP.

BLANK
Depending on the access level to the /GROUP profile, the terminal user can become the OWNER of the new profile.

userid
Depending on the access level to the /GROUP profile, the specified user ID is inserted as the OWNER of the new user ID.

group
The specified GROUP is used as OWNER of the new user ID.

=DFLTGRP
Reflects the default group DFLTGRP as specified or defaulted on the command. If this value resolves to the special value =OWNER, indicating the OWNER of the new profile, the command is rejected. See the description at =DFLTGRP for details.

=MYOWNER
Reflects the owner of the terminal user. If this value is a GROUP, the value is used as the OWNER of the new user ID. If this value is a user ID, further processing is dependent on the access level that the terminal user must the /GROUP profile.

=USERID(n)
Reflects the first *n* characters of the new user ID itself. This value must be a user ID or GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to permit definition of the USERID in C4R.USER.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to permit definition. In all other situations, the value =RACGPID is considered an error, and the current GROUP of the terminal user is used instead.

Verification of the specified owner

The following set of profiles is used when a new OWNER is specified in the **ADDUSER** or **ALTUSER** command.

RACF itself does not impose any constraints on the value of the new owner. The new owner must be an existing user ID or existing GROUP only. Aside from this restriction, all values are allowed. This set of profiles can be used to restrict the choice of new OWNERs. If the use of the specified OWNER is not accepted by any of these general policy rules, the explicit profile in the subsequent section is used.

- C4R.USER.OWNER.=RACUID(*n*)

  This profile specifies a special generic policy for the OWNER in ADDUSER and ALTUSER commands. The =RACUID stands for the userid of the terminal user. If the substring=(RACUID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the userids.

  This profile is a discrete profile. Only the single digit between parentheses is variable; it must be specified as a value in the range 1-8. It is not possible to use a true generic profile.

  If the OWNER specified by the terminal user is accepted, processing continues with the additional verifications described like /SCOPE and /GROUP.

  No profile found
  The terminal user ID of the user is not used as naming convention or restriction for the OWNER.

  NONE
  The specified OWNER is not allowed. The command is rejected. This decision can be overruled by authorization to profile *owner.userid*.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted.

  CONTROL
  Same as UPDATE.

- C4R.USER.OWNER.=RACGPID(*n*)

  This profile specifies a special generic policy for the OWNER in **ADDUSER** and **ALTUSER** commands. The =RACGPID stands for the list of groups to which the terminal user is connected. All the groups of the user are used, independent of the setting of "list of group access checking". If the substring=(RACGPID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. It is only used if =RACUID(*n*) is not present or does not match. If you defined multiple of these profiles, only the one with the lowest value for *n* is used.

This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

If the OWNER specified by the terminal user is accepted, processing continues with the additional verifications described like /SCOPE and /GROUP.

No profile found
The terminal GROUPs of the user are not used as naming convention or restriction for the OWNER.

NONE
The specified OWNER is not allowed. The command is rejected. This decision can be overruled by authorization to profile *owner.userid*.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE

- C4R.USER.OWNER.=USERID(*n*)

This profile specifies a special generic policy for the OWNER in ADDUSER and ALTUSER commands. The special value =USERID represents the affected user profile itself. This profile can be used to enforce a naming convention that states that the first *n* characters of a user ID must match the first *n* characters of its owner.

The =USERID stands for the userid in the command. If the substring=(USERID,1,*n*) matches the specified OWNER, this profile is used in preference to other generic profiles, independent of the value of *n*. It is only used if =RACUID(*n*) and =RACGPID(*n*) are not present or do not match. If you defined multiple of these profiles, only the one with the lowest value for *n* is used.

This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

If the OWNER specified by the terminal user is accepted, processing continues with the additional verifications described like /SCOPE and /GROUP.

No profile found
The target user ID itself is not used as naming convention or restriction for the OWNER.

NONE
The specified OWNER is not allowed. The command is rejected. This decision can be overruled by authorization to profile *owner.userid* described as following.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

If any of the above three profiles allow the specified OWNER, the next profile rule is skipped. Processing continues with the /SCOPE, /GROUP, and /DFLTGRP policies that are described as following. If the preceding profiles did not authorize the use of a certain OWNER, the next profile is used as alternative authorization method.

- C4R.USER.OWNER.*owner.userid*

The primary purpose of this control is to specify a policy if none of the general policies that were previously described applies. The variable *owner* represents the new OWNER of the *userid*. This case allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The OWNER as verified by this policy profile is still subjected to the additional policies /SCOPE, /GROUP, and /DFLTGRP.

No profile found
This control is not implemented.

NONE
The command is rejected.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

Additional policy controls for the owner

Aside from the profiles that are intended to enforce a naming convention, it is also possible to implement a policy that is based on the existing RACF group hierarchy.

The following profiles allow specification of general rules for the new OWNER. By using more specific (or fully qualified) profiles, you can specify that some users or groups are exempt from such a restriction.

The three profile rules are used as an extra set of policies. If the specified OWNER is accepted by any of the rules above, it is verified against the three policies. If it fails any of these policies below, the command is rejected.

- C4R.USER.OWNER./SCOPE.*owner.userid*

  This profile is used to control if the new OWNER as specified by the terminal user must be within the scope of a group-SPECIAL attribute. This case applies both for the **ADDUSER** command and the **ALTUSER** command. This profile can prevent the terminal user from "giving away" user ID profiles that are within the scope of a group-SPECIAL attribute.

  The variables *userid* and *owner* represent the affected User profile and its new OWNER. This step allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  Specifying a user ID as the new owner is always considered to be outside the administrative scope of the terminal user.

  If the profile is within scope of a group-SPECIAL authorization, the use of this authorization is recorded by the profile

  – C4R.USESCOPE.*group*

  Successful access with UPDATE authority to this profile is recorded through SMF. In this policy profile, the qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authorization over the specified owner. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

  The following access levels are supported for the /SCOPE policy profile:

  No profile found
  The terminal user group-SPECIAL scope is not used to control the new OWNER of user profiles.

  NONE
  If the specified new OWNER is outside the scope of a group-SPECIAL attribute of the terminal user, the command is rejected.

  READ
  Same as NONE.

UPDATE
The specified OWNER is accepted, irrespective of the scope of the terminal user.

CONTROL
Same as UPDATE.

- C4R.USER.OWNER./GROUP.*owner.userid*

The profile is used to control if the specified OWNER must be a RACF GROUP or not. This profile is verified independently of the other preceding profiles. If either the =OWNER or the /OWNER profiles are used, this policy rule is bypassed.

The variables *userid* and *owner* represent the affected USERID and its new OWNER. This step permits specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /GROUP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. The specified OWNER can be a GROUP or a user ID.

NONE
If the specified owner is an existing RACF group, the command is accepted. In all other situations, the command is rejected.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted even if it does not represent an existing group. If the specified OWNER is not a valid entry, the command is rejected by RACF.

CONTROL
Same as UPDATE.

- C4R.USER.OWNER./DFLTGRP.*owner.userid*

This profile is used to control if the OWNER as specified by the terminal user must be the same as the DFLTGRP of the user ID. This case applies both for the **ADDUSER** command and the **ALTUSER** command.

The values *userid* and *owner* represent the affected USERID and its new OWNER. This step permits specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /DFLTGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. The specified OWNER can be different from the current DFLTGRP.

NONE
The specified new OWNER must be the same as the current or new DFLTGRP.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted, irrespective of the value of the DFLTGRP.

CONTROL
Same as UPDATE.

Implementation of a new user policy

Use the guidelines in this scenario to implement policy profiles in the specification of new user IDs.

The previous sections describe the profiles that are used in the decision process for the user ID and the place in the RACF group hierarchy. These profiles allow great flexibility in specification of the user IDs that

a terminal user is allowed to create. Use the following scenario to describe the steps that are required to implement a new user policy:

- Central administrators can define all users.
- De-centralized administrators can define users only for their own department.
- Departments can be recognized by the RACF group structure (ownership).
- All user profiles must be owned by a RACF group, according to the departmental structure.
- A user ID naming convention is used where the first 3 characters of the `userid` are the same as the first 3 characters of the department name.

For the preceding organization, the following profiles can be implemented:

c4r.user.id.* uacc(none) sysadmin(update)
This profile ensures that only system administrators are allowed to define new user profiles outside the regular naming conventions.

c4r.user.id.=racuid(3) uacc(update)
This profile allows all decentralized administrators to define new users that have as first 3 characters the same characters as the decentralized administrator. Only those decentralized administrators who have `CLAUTH(USER)` and the group-SPECIAL attribute are allowed to define new users.

Note: The implementation of this policy through the `=RACGPID(3)` profile is not as effective. All the groups of the terminal user are used as naming convention. It is not guaranteed that the terminal user is not connected to a functional group of another department, which has a different prefix.

c4r.user.delete. uacc(none) sysadmin(update)**
This profile ensures that only the central system administrators are allowed to delete existing users.

c4r.user.=dfltgrp. uacc(update) sysadmin(control) appldata('=myowner')**
This profile specifies that independent of what any decentralized administrator specifies, the newly defined `userid` is always connected to the GROUP that owns the decentralized administrator. Central system administrators must specify a `DFLTGRP` because this control does not apply to them. However, see the next profile.

c4r.user./dfltgrp. uacc(none) sysadmin(update) appldata('USERS')**
If the central system administrator does not specify a DFLTGRP for new users, the user is assigned to the group called USERS.

c4r.user.=owner. uacc(update) sysadmin(control) appldata('=myowner')**
This profile ensures that the OWNER of the new user ID profile is the same as the OWNER of the decentralized administrator. Again, this control does not apply to the central system administrators. The next profile is especially defined for their usage.

c4r.user./owner. uacc(none) sysadmin(update) appldata('=dfltgrp')**
The use of `=DFLTGRP` as the `APPLDATA` value ensures that if no value is specified for the OWNER, the OWNER is completed by zSecure Command Verifier to be the same as the `DFLTGRP` for the new user ID.

Implementation of an existing user policy

Use the guidelines in this scenario to implement policy profiles in the specification of existing user IDs.

Continuing with the scenario used in the New User policy example in “Implementation of a new user policy” on page 87, you can also set up a policy to handle existing users. For this example, extend the previously defined New User policy with some additional rules:

- Central administrators can modify all users.
- Central administrators can specify any user or group as owner.
- De-centralized administrators can change the owner within their own department only.
- De-centralized administrators can return existing users to the `not-in-a-department` pool.
- The `not-in-a-department` pool is implemented through the RACF group HOLDING.

This example does not describe the profiles that are needed to connect users to a group or to remove them, or how to change the user authorizations and attributes. The next section shows the profiles that are required to control the **CONNECT** and **REMOVE** commands. It is assumed in this case that the user IDs are somehow connected to the RACF GROUP HOLDING.

For the preceding organization the following profiles can be implemented.

c4r.user.dfltgrp./scope. uacc(none) sysadmin(control)**
This profile ensures that only system administrators are allowed to change the default group to all values. The decentralized administrators can specify only groups that are within their scope of control. Because this /SCOPE profile is defined, normal users can no longer permanently change their own default groups. They can still select their current connect GROUP during logon.

c4r.user.owner./scope. uacc(none) sysadmin(control)**
This profile ensures that only system administrators have unrestricted authorization to change the OWNER of existing users. Decentralized administrators can change the OWNER only within their scope. They cannot give away any of their user IDs. Normal users cannot change the OWNER of any user IDs that they own because they do not have group-SPECIAL: everything is outside their scope.

c4r.user.dfltgrp.HOLDING.* uacc(update)
This profile identifies the RACF GROUP HOLDING as an exceptional group. All users in the system can select the RACF GROUP HOLDING as their default group if they are already connected to the GROUP.

c4r.user.owner.HOLDING.* uacc(control)
This profile identifies the RACF GROUP HOLDING as an exceptional group. It allows all decentralized administrators to transfer existing users from their current OWNER to the HOLDING group.

Policy profiles for user attributes and authorizations

This section describes the controls that can be implemented for user attributes and authorizations.

Similar attributes and authorizations also exist for GROUP connections. Some of the keywords that are available on the **ADDUSER** and **ALTUSER** command apply to the GROUP connections for the DFLTGRP or the specified GROUP. For the description of the CONNECT attributes and authorizations, see "CONNECT management" on page 130. The user/system level keywords are summarized in the following table.

Table 19. Profiles used for RACF attributes. The entries in this table reflect the keywords that are specified on the **ADDUSER** and **ALTUSER** commands

Command	Keyword	Profile
ADDUSER	N/A	C4R.USER.=ATTR.*owner.userid*
ADDUSER ALTUSER	SPECIAL	C4R.USER.ATTR.SPECIAL.*owner.userid*
ADDUSER ALTUSER	OPERATIONS	C4R.USER.ATTR.OPERATIONS.*owner.userid*
ADDUSER ALTUSER	AUDITOR	C4R.USER.ATTR.AUDITOR.*owner.userid*
ADDUSER ALTUSER	ROAUDIT	C4R.USER.ATTR.ROAUDIT.*owner.userid*
ADDUSER ALTUSER	RESTRICTED	C4R.USER.ATTR.RESTRICTED.*owner.userid*
ALTUSER	UAUDIT	C4R.USER.ATTR.UAUDIT.*owner.userid*
ADDUSER ALTUSER	ADSP	C4R.USER.ATTR.ADSP.*owner.userid*
ADDUSER ALTUSER	GRPACC	C4R.USER.ATTR.GRPACC.*owner.userid*
ADDUSER ALTUSER	NOPASSWORD NOPHRASE	C4R.USER.ATTR.PROTECTED.*owner.userid*
ADDUSER ALTUSER	OIDCARD	C4R.USER.ATTR.OIDCARD.*owner.userid*
ALTUSER	REVOKE	C4R.USER.ATTR.REVOKE.*owner.userid*
ALTUSER	RESUME	C4R.USER.ATTR.RESUME.*owner.userid*

Table 19. Profiles used for RACF attributes. The entries in this table reflect the keywords that are specified on the **ADDUSER** and **ALTUSER** commands *(continued)*		
Command	**Keyword**	**Profile**
ALTUSER	REVOKE(*date*) NOREVOKE	C4R.USER.ATTR.REVOKEDT.*owner.userid*
ALTUSER	RESUME(*date*) NORESUME	C4R.USER.ATTR.RESUMEDT.*owner.userid*

Mandatory value profiles for user attributes

By using the mandatory value policy profile for user attributes, an installation can specify that new users must always have certain attributes, irrespective of the keywords that are used on the **ADDUSER** command.

The most obvious use for this function is setting the NOADSP and NOGRPACC values. The standard policy profiles can be used to prevent a terminal user from specifying the value ADSP or GRPACC. If they accidentally specify such a value, the command can be rejected. Use of the Mandatory Value policy profile allows effectively ignoring any non-acceptable value. The Mandatory Attribute policy profile and the applicable access level is described in the following list.

The qualifier =ATTR in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

- C4R.USER.=ATTR.*owner.userid*

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The mandatory attributes do not apply for the terminal user.

  READ
  The APPLDATA of the Mandatory Value policy profile is used as the list of attributes for the new user.

  UPDATE
  Same as READ.

  CONTROL
  The control is not active for the terminal user. No mandatory value is supplied. The attributes as specified by the terminal user are used in the command.

Note:

1. In contrast to other mandatory value policy profiles, the attributes assigned through the =ATTR policy profile are verified against the regular attribute policy profiles. For example, if the =ATTR policy assigns the OPERATIONS attribute, the terminal user must also have access to the matching C4R.USER.=ATTR.OPERATIONS.*owner.userid* policy profile.
2. The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value profiles, these profiles lead to the odd situation that access NONE has the same net result as access CONTROL.

The **APPLDATA** field of the Mandatory Value policy profile specifies a list of user attributes. The following user attributes are recognized.

- SPECIAL and NOSPECIAL
- OPERATIONS and NOOPERATIONS
- AUDITOR and NOAUDITOR
- ROAUDIT and NOROAUDIT
- PASSWORD and NOPASSWORD

- RESTRICTED and NORESTRICTED
- OIDCARD and NOOIDCARD
- ADSP and NOADSP
- GRPACC and NOGRPACC

It is not possible to use abbreviations for the attributes. If multiple attributes must be assigned, the individual attributes must be separated by a single comma without any intervening blanks. For example:

```
NOADSP,NOGRPACC
```

User attributes and access level descriptions

Use these guidelines and policy profile attributes to set the access levels that control which keywords and values can be used.

In general, the access level that is required is UPDATE to give the attribute or READ to take away the attribute. For the **ADDUSER** commands, zSecure Command Verifier does not check the default value that is used by RACF. However, the non-default value is checked by using a method similar to the checking done for the **ALTUSER** command.

- C4R.USER.ATTR.SPECIAL.*owner.userid*
- C4R.USER.ATTR.OPERATIONS.*owner.userid*
- C4R.USER.ATTR.AUDITOR.*owner.userid*
- C4R.USER.ATTR.ROAUDIT.*owner.userid*
- C4R.USER.ATTR.ADSP.*owner.userid*
- C4R.USER.ATTR.GRPACC.*owner.userid*

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to specify either keyword on the **ALTUSER** command. The no-attribute keyword is allowed (defaulted) on the **ADDUSER** command.

  READ
  The terminal user is authorized to explicitly specify the no-attribute keyword on the **ALTUSER** command. This setting allows removal of these attributes.

  UPDATE
  The terminal user is authorized to specify both keywords on the **ALTUSER** command. This setting allows regular maintenance of these attributes.

  CONTROL
  The control is not implemented for the terminal user. The terminal user is authorized to specify both keywords on the **ADDUSER** and **ALTUSER** command. This setting allows regular maintenance of these attributes.

  In all the preceding situations, the terminal user needs sufficient RACF authorization to specify the keyword. For instance, for most keywords, the terminal user must have the SPECIAL attribute.

- C4R.USER.ATTR.RESTRICTED.*owner.userid*

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to specify the **(NO-)RESTRICTED** operand. The default value **NORESTRICTED** is allowed on the **ADDUSER** command.

  READ
  The terminal user is authorized to specify the RESTRICTED keyword on the **ADDUSER** and **ALTUSER** command. This setting reduces the standard access of the target user to only those resources that are explicitly authorized for use.

UPDATE
The terminal user is authorized to specify the NORESTRICTED keyword on the **ALTUSER** command. This setting allows the regular maintenance of the RESTRICTED attribute.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to specify both keywords on the **ADDUSER** and **ALTUSER** command.

- C4R.USER.ATTR.UAUDIT.*owner.userid*

The UAUDIT attribute can be seen and assigned only by a terminal user with the System-AUDITOR attribute. It results in all RACF verifications being audited through SMF.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the **(NO-)UAUDIT** operand.

Note: The **(NO-)UAUDIT** keyword is not available on the **ADDUSER** command.

READ
The terminal user is authorized to specify the NOUAUDIT keyword on the **ALTUSER** command.

UPDATE
The terminal user is authorized to specify the UAUDIT keyword on the **ALTUSER** command. This setting allows the regular maintenance of the UAUDIT attribute.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to specify both keywords on the **ALTUSER** command.

- C4R.USER.ATTR.PROTECTED.*owner.userid*

The PROTECTED attribute is effectively the status of a user without a PASSWORD or PHRASE. Technically, an OIDCARD is also relevant, but OIDCARDs are no longer used in modern systems. A user with only a password can be made protected by removing the password. With the introduction of the KDFAES password algorithm, it became possible to create *phrase-only users*. Removing the password phrase from such a user also creates a protected user.

An **ALTUSER** command with the NOPASSWORD or a NOPHRASE keyword can result in the creation of a protected user. Similarly, the use of the PASSWORD or PHRASE keyword can result in the removal of the protected status. If the **ALTUSER** command changes the protected status of the user, the PROTECTED policy in the current section is used in combination with the appropriate PASSWORD or PHRASE policies. If the protected status is not affected by the command, only the regular PASSWORD or PHRASE policies are used. See "Policy profiles for user password and phrase management" on page 95.

The policy described in this section also applies for the **ADDUSER** command if it causes the creation of a protected user. The policy can be used to enforce that new user IDs are always assigned either a password or a phrase. User IDs that are initially created as PROTECTED through the **ADDUSER** command are subject to special access level requirements for this policy. If such a user ID has never been used, removing the protected attribute requires only READ access. This is the same access level that is required to create such a protected user ID. If the user ID has been used, or if an **ALTUSER RESUME** command has been issued for the user ID, the special status is reset. In that case, regular UPDATE access to the current policy is required to remove the protected status.

The following access levels apply to changing the protected status:

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the NOPASSWORD or NOPHRASE keyword if it causes the target user to become protected. If the target user currently is protected, use of the PASSWORD or PHRASE keyword is prohibited as well.

READ
The terminal user is authorized to specify the NOPASSWORD or NOPHRASE keyword to create a protected user. If the target user currently has the protected status, but has never been used (**RACF LISTUSER** output shows LAST-ACCESS=UNKNOWN), removal of the protected status is allowed. The terminal user also needs sufficient access to the applicable PASSWORD or PHRASE policy profiles.. If the target user currently has the protected status, and has been used (**RACF LISTUSER** output shows a LAST-ACCESS date and time), removing the protected status through the PASSWORD or PHRASE keyword is prevented.

UPDATE
The terminal user is authorized to specify the NOPASSWORD or NOPHRASE keyword to create protected users. Removing the protected status through the **ALTUSER** command with the PASSWORD or PHRASE keyword is also allowed. The terminal user also needs sufficient access to the applicable PASSWORD or PHRASE policy profiles. UPDATE access is required to assign a PASSWORD or PHRASE to a user ID that was created as protected and that was subsequently used, or was subjected to an ALTUSER RESUME command.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to specify all keywords that affect the protected status of the ID.

- C4R.USER.ATTR.OIDCARD.*owner.userid*

This profile is used to control the use of the NOOIDCARD and the OIDCARD keyword. The default keyword NOOIDCARD is not checked for the **ADDUSER** command.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the **(NO-)OIDCARD** operand on the **ALTUSER** command. The NOOIDCARD keyword is allowed (defaulted) on the **ADDUSER** command.

READ
The terminal user is authorized to specify the NOOIDCARD keyword on the **ALTUSER** command to reset the OIDCARD of an existing user.

UPDATE
The terminal user is authorized to specify the OIDCARD keyword to set the oidcard of a new or existing user. This command succeeds only if the terminal user has physical access to a terminal attached magnetic card reader.

CONTROL
The control is not implemented for the terminal user.

- C4R.USER.ATTR.REVOKE.*owner.userid*

This policy profile applies only to the REVOKE attribute without a future revoke date. Management of revoke dates is controlled by the REVOKEDT policy profile.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to revoke the user. This setting applies to the REVOKE keyword without any specification of a future revoke date.

READ
The terminal user is authorized to REVOKE a userid. This setting applies to the REVOKE keyword without specification of a future revoke date.

UPDATE
Same as READ.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to revoke a userid.

- C4R.USER.ATTR.RESUME.*owner.userid*

  This policy profile applies only to the RESUME attribute without a future resume date. Management of resume dates is controlled by the RESUMEDT policy profile.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to resume the user. This setting applies RESUME keyword without specification of a future resume date.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to RESUME a `userid`. This setting applies only to an immediate RESUME without a future resume date.

  CONTROL
  The control is not implemented for the terminal user. The terminal user is authorized to resume the `userid`.

- C4R.USER.ATTR.REVOKEDT.*owner.userid*

  This policy profile applies to the REVOKE attribute with a future `revoke` date. It also applies to the use of the NOREVOKE keyword to remove existing revoke dates.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to manage `revoke` dates for the user. This setting applies to both REVOKE(*date*) and the NOREVOKE option.

  READ
  Same as NONE

  UPD
  The terminal user is allowed to manage `revoke` dates through REVOKE(*date*) or NOREVOKE.

  CONTROL
  The control is not implemented for the terminal user. The terminal user is authorized to manage the revoke dates for the `userid`.

- C4R.USER.ATTR.RESUMEDT.*owner.userid*

  This policy profile applies to the RESUME attribute with a future resume date. It also applies to the use of the NORESUME keyword to remove existing resume dates.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to manage `resume` dates for the user. This setting applies to both RESUME(*date*) and the NORESUME option.

  READ
  Same as NONE

  UPD
  The terminal user is allowed to manage `resume` dates through RESUME(*date*) or NORESUME.

  CONTROL
  The control is not implemented for the terminal user. The terminal user is authorized to manage the resume dates for the `userid`.

Policy profiles for user password and phrase management

This section summarizes all the keywords and controlling profiles that are related to a user's password and password phrase.

Although the PROTECTED attribute is also controlled by the (NO)PASSWORD and (NO)PHRASE keywords, it is described in "User attributes and access level descriptions" on page 91, together with other attributes.

Of the policy profiles described here, one policy profile is used to control the authority to set or change passwords and another one is used for setting or changing password phrases. To provide limited quality control for passwords set by an administrator, two special policy profiles are provided. The remaining password policies control the setting of the password/phrase interval and the use of the expired/noexpired keyword.

Attention: This control does not enforce any standards on the passwords as set by users when they change their password during logon.

zSecure Command Verifier also provides two policy profiles to control who can use the PWCONVERT and PWCLEAN keywords on the ALTUSER command. Because these two options are not used for regular password administration, the policy profiles are described in the general section about other user-related policy profiles. See "Other user-related policy profiles" on page 104.

The following table lists the policy profiles available to manage RACF user passwords and phrases. Although the policy profiles for interval and expiration suggest that they apply only to passwords, they apply to passwords and phrases. There are no separate policies to control the password interval and the phrase interval. Detailed descriptions for each profile in the table are provided following the table.

Table 20. Profiles used for RACF passwords. The entries in this table reflect the keywords that are specified on the **ADDUSER**, **ALTUSER**, and **PASSWORD** commands

Command	Keyword	Profile
ADDUSER ALTUSER	PASSWORD	C4R.USER.PASSWORD.*owner.userid*
ADDUSER ALTUSER	PASSWORD	C4R.USER./PASSWORD.*owner.userid*
PASSWORD	PASSWORD	C4R.USER.PASSWORD.=RACUID
ADDUSER ALTUSER	PHRASE	C4R.USER.PHRASE.*owner.userid*
PASSWORD	PHRASE	C4R.USER.PHRASE.=RACUID
ADDUSER ALTUSER	PASSWORD	C4R.USER.PASSWORD.=DFLTGRP
PASSWORD	USER(*userid*)	C4R.USER.PASSWORD.=DFLTGRP
ADDUSER ALTUSER	PASSWORD	C4R.USER.PASSWORD.=USERID
PASSWORD PHRASE	(NO)INTERVAL	C4R.USER.=PWINT.*owner.userid*
PASSWORD PHRASE	(NO)INTERVAL	C4R.USER.PWINT.*owner.userid*
ALTUSER	(NO)EXPIRED	C4R.USER.PWEXP.*owner.userid*

The following entries describe the policy profiles and access levels that are used to control the password-related functions of zSecure Command Verifier.

- C4R.USER.PASSWORD.*owner.userid*

  This policy profile controls the setting of the password by an administrator through the **ADDUSER** or **ALTUSER** command. Setting your own password through the **PASSWORD** command is controlled by the =RACUID profile. Some levels of RACF allow setting the password of another user through the **PASSWORD** command. This is controlled by the password quality profile for value =DFLTGRP.

  If the use of the (NO)PASSWORD keyword does not change the protected status of the target ID, only the current profile is used. If these keywords make the user protected, or remove the protected status,

the C4R.USER.ATTR.PROTECTED profile is used in combination with current PASSWORD policy profile. For more information about the PROTECTED policy, see "User attributes and access level descriptions" on page 91. The PASSWORD profile described here controls the authorization to manage passwords for protected and non-protected users.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the **PASSWORD** operand. When using the **ADDUSER** command, and depending on the level of RACF, this access level can result in users with a RACF default password (=DFLTGRP) or in PROTECTED users. Both can be prevented by defining adequate policies for password quality or the protected status.

READ
Same as NONE.

UPDATE
The terminal user is authorized to specify the **PASSWORD** operand on the **ADDUSER** or **ALTUSER** command to set or reset the password for an existing user. However, if the target user currently has the PROTECTED attribute, the terminal user also needs sufficient access to the applicable PROTECTED policy profile. UPDATE access to the PASSWORD policy allows for normal password maintenance.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to specify the PASSWORD keyword. If the target userid currently has the PROTECTED attribute, the terminal user also needs sufficient access to the applicable PROTECTED policy profile..

- C4R.USER./PASSWORD.*owner.userid*

This policy profile is used when the **ADDUSER** or **ALTUSER** command is used with the PASSWORD keyword, but without a value for the password. In this case, the DFLTGRP of the target user would be used as password. Depending on the level of RACF, such an **ADDUSER** command could also result in the definition of a PROTECTED user. For the **ADDUSER** command, it is possible to force the current policy to apply by using the PASSWORD keyword without a value for the password. It is also possible to automatically insert the PASSWORD keyword using the mandatory attribute policy as described in "Mandatory value profiles for user attributes" on page 90.

If the current policy applies, it is possible to automatically assign a value for the password. Using the value RANDOM for the APPLDATA instructs Command Verifier to insert a random value for the password. The generated password is always eight characters long and each character is selected from all available types:

– By default, the password characters are selected from the set consisting of the uppercase alphabetic characters, numerics, and the three national characters (@, #, and $).
– If mixed case passwords are enabled (SETROPTS PASSWORD(MIXEDCASE)), lowercase alphabetic characters can also be used.
– If special characters are enabled (SETROPTS PASSWORD(SPECIALCHARS)), the special characters as documented in the *RACF Security Administrator's Guide* can also be used.

Password rules that are specified through the SETROPTS command are mainly intended to force users to choose characters from each set or to prevent the use of common words. Command Verifier-generated passwords are truly random. Therefore, they are not guaranteed to adhere to password rules that limit the length or the choice of characters. If the installation has defined password rules, or uses a new password exit, RACF might not accept the generated random password if used in combination with the NOEXPIRE option. An example random password that violates the mixedall password rule is $%QyaFXi, because it lacks a numeric character. Forcing a numeric character would reduce the time that is needed for a brute force attack of the password by approximately a factor eight.

If the ADDUSER or ALTUSER command specifies a value for the PASSWORD, the /PASSWORD policy profile is not used.

The qualifier /PASSWORD in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. No action is performed.

NONE
No default value is supplied.

READ
The generated value for the password is inserted in the command. The password is not disclosed to the terminal user.

UPDATE
The generated value for the password is inserted in the command. A message is issued to the terminal user that shows the new password.

CONTROL
The control is not implemented for the terminal user. No default value for the password is supplied. RACF uses the DFLTGRP of the target user as the new value of the password.

The following values for APPLDATA are supported.

BLANK
This value is used to indicate that RACF default processing must be used. This can trigger other policies, like those for password quality or creation of protected users.

RANDOM
zSecure Command Verifier generates a random value for the password. The generated password is always eight characters long and selects characters from all available types.

Other
Although this value must be considered an error, processing continues as if no value for the APPLDATA was specified. This can trigger other policies, like those for password quality or creation of protected users.

- C4R.USER.PASSWORD.=RACUID

  This profile describes the authority of a user to change its own password by using the **PASSWORD** command. You cannot use generic characters to cover the =RACUID qualifier in the policy profile; it must be present in the exact form shown.

  Use care when you define a generic value for the PASSWORD qualifier because the resulting policy profile might also match the authority to change your own non-base segments. For more information about the policy profiles for non-base segments, see "Profiles for controlling management of non-base segments" on page 52.

  The following access rules apply:

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to specify the **PASSWORD** operand. This setting means that the user can change its password only during logon.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to specify the **PASSWORD** operand on the **PASSWORD** command to change its password.

  CONTROL
  The control is not implemented for the terminal user.

- C4R.USER.PHRASE.*owner.userid*

This policy profile controls the setting of the password phrase through the **ADDUSER** or **ALTUSER** command. Setting your own password phrase through the **PASSWORD** or **PHRASE** command is controlled by the =RACUID profile.

If the use of the (NO)PHRASE keyword does not change the protected status of the target ID, only the current profile is used. If these keywords make the user protected, or remove the protected status, the C4R.USER.ATTR.PROTECTED profile is used in combination with current PHRASE policy profile. For more information about the PROTECTED policy, see "User attributes and access level descriptions" on page 91. The PHRASE profile described here controls the authorization to manage phrases for protected and non-protected users.

The following access levels apply to changing the phrase:

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the **PHRASE** operand.

READ
Same as NONE.

UPDATE
The terminal user is authorized to specify the **PHRASE** operand on the **ADDUSER** or **ALTUSER** command to set or reset the phrase for an existing user. However, if the target user currently has the PROTECTED attribute, the terminal user also needs sufficient access to the applicable PROTECTED policy profile. UPDATE access to the PHRASE policy allows for normal phrase maintenance.

CONTROL
The policy is not implemented for the terminal user. The terminal user is authorized to specify the **PHRASE** keyword. If the target userid currently has the PROTECTED attribute, the terminal user also needs sufficient access to the applicable PROTECTED policy profile.

- C4R.USER.PHRASE.=RACUID

This profile describes the authority of a user to change its own password phrase by using the **PASSWORD** or **PHRASE** command. RACF does not allow adding a password phrase through the **PASSWORD** or **PHRASE** command. You can change only the value of existing password phrases. You cannot use generic characters to cover the =RACUID qualifier in the policy profile; it must be present in the exact form shown.

Use care when you define a generic value for the PHRASE qualifier because the resulting policy profile might also match the authority to change your own non-base segments. For more information about the policy profiles for non-base segments, see "Profiles for controlling management of non-base segments" on page 52.

The following access rules apply:

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify the **PHRASE** operand. This setting means that the user can change only its password phrase during logon, if and when this setting is supported by the application.

READ
Same as NONE.

UPDATE
The terminal user is authorized to specify the **PHRASE** operand on the **PASSWORD** or **PHRASE** command to change its password phrase.

CONTROL
The control is not implemented for the terminal user.

- C4R.USER.PASSWORD.=DFLTGRP

This profile is used to control the authorization to leave the password value blank at the **ADDUSER** and **ALTUSER** command. Leaving the password value blank results in RACF using the DFLTGRP of the user for the new password. Explicitly setting the PASSWORD to the DFLTGRP is also controlled by this policy.

Depending on the level of RACF, the **PASSWORD** command, when issued for another user without the INTERVAL keyword, resets the password to the default group of that user. This policy profile does also apply to that form of the **PASSWORD** command.

The qualifier =DFLTGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

Activation of the preceding /PASSWORD policy preempts this policy. Implementation of the default value policy can result in setting a value for the password. In that case, the password value no longer matches the DFLTGRP, and the current policy profile does not apply.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to use the **ADDUSER** command without explicitly specifying a value for the password. If you use the PASSWORD keyword on the **ALTUSER** command without specifying a value, the command is rejected as well.

READ
The terminal user is authorized to leave the password value blank or explicitly specify the DFLTGRP on the **ADDUSER** command. On the **ALTUSER** command, use of the PASSWORD keyword without an explicit value is not allowed.

UPDATE
The terminal user is authorized to leave the password value blank or explicitly specify the DFLTGRP on both the **ADDUSER** and the **ALTUSER** command.

CONTROL
The control is not implemented for the terminal user. A password equal to the DFLTGRP is acceptable.

- C4R.USER.PASSWORD.=USERID

This profile is used to control the authorization to specify the userid as part of the new password on the **ADDUSER**, **ALTUSER**, and **PASSWORD** commands.

The qualifier =USERID in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to use the userid as part of the value for the new password. The command is rejected.

READ
Same as NONE.

UPDATE
The terminal user is authorized to use the user ID as part of the new value for the password.

CONTROL
The control is not implemented for the terminal user. A password equal to the user ID is acceptable.

- C4R.USER.=PWINT.*owner.userid*

This policy profile can be used to enforce a particular value for the password and phrase interval for a user. The interval that is defined by this policy profile is used to override any value that is specified by the terminal user. If the **PASSWORD** or **PHRASE** command is used without the INTERVAL keyword, the interval is not changed. Although the qualifier =PWINT suggests that this policy profile applies only for the password interval, RACF uses the same interval for the password and phrase. Therefore, this policy profile also applies to both.

The qualifier =PWINT in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. No action is performed.

NONE
No action. No mandatory value is enforced.

READ
The **APPLDATA** field is retrieved and used for the new interval for the user.

UPDATE
Same as READ.

CONTROL
The control is not implemented for the terminal user. No mandatory value is enforced.

The values possible for the **APPLDATA** field are given as following.

BLANK
This value is used to indicate that the RACF SETROPTS value must be used as a default.

interval
The *interval* must be specified by 3 digits that include leading zeros. Ensure that this value is less or equal to the RACF SETROPTS value. Otherwise, the resulting command might fail.

NEVER
The password interval is set to `never`. This setting results in a password and phrase that never expire. RACF requires extra authorization to specify this value. If the terminal user lacks this authorization, the command is rejected by RACF.

other
This value is an error. The RACF SETROPTS value is used as maximum.

- `C4R.USER.PWINT.`*owner.userid*

This profile can be used to control the maximum value of the password and phrase interval. In the best fitting profile, the maximum value for the interval must be specified by the APPLDATA. The *interval* must be specified by 3 digits that include leading zeros. The terminal user specified value is compared against the value that is defined in the APPLDATA. If the value in the command is higher than the value in the profile, the command is rejected. If the terminal user has CONTROL access the defined maximum value is ignored. Although the qualifier PWINT suggests that this policy profile applies only for the password interval, RACF uses the same interval for the password and phrase. Therefore, this policy profile also applies to both.

No profile found
This control is not implemented. No action is performed.

NONE
Changing the interval is not allowed. Any value that is specified by the terminal user is rejected.

READ
Same as NONE.

UPDATE
The value from the **APPLDATA** is used as a maximum value for the interval. If the terminal user specified value is less than or equal to the defined value, the command is accepted. The interval cannot be set higher than the system-wide default.

CONTROL
The control is not implemented for the terminal user. Any terminal user specified value is accepted.

The values possible for the **APPLDATA** field are given as follows.

BLANK
This value is used to indicate that the RACF SETROPTS value must be used as a maximum.

interval
The *interval* must be specified by 3 digits that include leading zeros.

NEVER
The interval can be set to NEVER. This setting results in a password and phrase that never expire. RACF requires extra authorization for this value. It is also possible to specify an interval that is less than or equal to the SETROPTS value.

other
This value is an error. The RACF SETROPTS value is used as maximum.

- C4R.USER.PWEXP.*owner.userid*

This policy profile can be used to control usage of the EXPIRED and NOEXPIRED options on the **ALTUSER** command. RACF already restricts the NOEXPIRED option to terminal users with the system special attribute and users with UPDATE access to the IRR.PASSWORD.RESET profile. The current policy profile allows further restriction on the target user. It also controls the authority to expire a password or phrase without setting a new value of a password or phrase. Although the qualifier PWEXP suggests that this policy profile applies only for expiration of passwords, it also applies to phrases.

The following access rules apply:

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to expire the current password and phrase through use of the EXPIRED keyword on the **ALTUSER** command. If a new value for the password or phrase is specified, the default value of EXPIRED is allowed. When specifying a new value for the password or phrase, the terminal user is not authorized to specify NOEXPIRED.

READ
Same as NONE

UPDATE
The terminal user is authorized to expire the current password and phrase through use of the EXPIRED keyword without specifying a new value for the password or phrase. When specifying a new value for the password or phrase, the terminal user is authorized to specify EXPIRED as well as NOEXPIRED. This access level allows regular maintenance of password and phrases.

CONTROL
The policy is not implemented for the terminal user. This access level allows regular maintenance of password and phrases.

Policy profiles for USER MFA data management

RACF implemented support for multifactor authentication through several new functions. Part of the required data must be added to USER profiles.

In the USER profiles, the relevant data is kept in several MFA-related fields in the BASE segment. The following policy profiles are used to control management of MFA-specific fields in the USER profile:

Table 21. Policy profiles to control management of MFA-specific fields in the USER profile

Keyword	Value	Profile
(NO)PWFALLBACK	n/a	C4R.USER.MFA.PWFALLBACK.*owner.userid*
(DEL)FACTOR	*factor-name*	C4R.USER.MFA.FACTOR.ID.*factor-name.owner.userid*
(NO)ACTIVE	*factor-name*	C4R.USER.MFA.FACTOR.ACTIVE.*factor-name*
TAG	*factor-name tag-name*	C4R.USER.MFA.FACTOR.TAG.*factor-name.tag-name*
DELTAG	*factor-name tag-name*	C4R.USER.MFA.FACTOR.TAG.*factor-name.tag-name*
NOTAGS	n/a	C4R.USER.MFA.FACTOR.TAG.*factor-name*.+

Table 21. Policy profiles to control management of MFA-specific fields in the USER profile (continued)		
Keyword	**Value**	**Profile**
ADDPOLICY DELPOLICY	*policy-name*	C4R.USER.MFA.POLICY.*policy-name.owner.userid*

The profiles in the preceding table describe the policies that can be used to verify the keywords and values as entered by the terminal user. The following list shows detail information about these policies and the supported access levels.

- C4R.USER.MFA.PWFALLBACK.*owner.userid*

  This profile describes the authorization to set the PWFALLBACK attribute for the user. The PWFALLBACK attribute is used during logon if the MFA server is not available or is unable to determine the validity of an active factor. The following access levels are used:

  No profile found
  This control is not implemented. Only RACF authorization is used to control setting the PWFALLBACK or NOPWFALLBACK attribute.

  NONE
  The terminal user is not authorized to assign either PWFALLBACK or NOPWFALLBACK. The command is rejected.

  READ
  The terminal user is authorized to assign NOPWFALLBACK. This is the default value on the ALTUSER command

  UPDATE
  The terminal user is authorized to assign PWFALLBACK and NOPWFALLBACK.

  CONTROL
  Same as UPDATE.

- C4R.USER.MFA.FACTOR.ID.*factor-name.owner.userid*

  This profile describes the authorization to add an MFA factor for a user, to modify options for the specified MFA factor, or to remove an MFA factor. The keywords to set the ACTIVE status, or to modify the list of TAGs, apply to the specified factor. The following access levels are used:

  No profile found
  This control is not implemented. Only RACF authorization is used to control adding, modifying, or removing the specified factor *factor-name*.

  NONE
  The terminal user is not authorized to add, modify, or remove the specified factor *factor-name*. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to manage the specified factor *factor-name*.

  CONTROL
  Same as UPDATE.

- C4R.USER.MFA.FACTOR.ACTIVE.*factor-name*

  This profile describes the authorization to activate an MFA factor for a user. This policy profile is used in conjunction with the FACTOR.ID policy profile. The FACTOR.ID policy profile controls the authority to manage the FACTOR for a particular user. The current policy profile controls the use of the ACTIVE status of the factor.

  No profile found
  This control is not implemented. Only RACF authorization is used to control the status of the specified factor.

NONE
The terminal user is not authorized to modify the status of the specified factor *factor-name*. The command is rejected. The terminal user is also not allowed to explicitly specify the default value of NOACTIVE for the status of the factor.

READ
Same as NONE.

UPDATE
The terminal user is authorized to change the ACTIVE status of the specified factor *factor-name*.

CONTROL
Same as UPDATE.

- C4R.USER.MFA.FACTOR.TAG.*factor-name.tag-name*

  This profile describes the authorization to manage TAGs for the specified factor. This policy profile is used in conjunction with the FACTOR.ID policy profile. The FACTOR.ID policy profile controls the authority to manage the FACTOR for a particular user. The current policy profile controls the management of the TAGs of the factor. If multiple tags are set or removed in a single command, the terminal user must have sufficient authority for all tags. If the terminal user has insufficient authority to one or more tags, the entire command is rejected. The special value + (plus sign) is used for the *tag-name* to designate the use of the NOTAGS keyword.

  No profile found
  This control is not implemented. Only RACF authorization is used to control managing TAGS.

  NONE
  The terminal user is not authorized to manage the tag *tag-name* for the factor *factor-name*. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to manage the tag *tag-name* for the factor *factor-name*.

  CONTROL
  Same as UPDATE.

- C4R.USER.MFA.POLICY.*policy-name.owner.userid*

  This profile describes the authorization to add or remove an MFA policy for a user. The following access levels are used:

  No profile found
  This control is not implemented. Only RACF authorization is used to control adding, modifying, or removing the specified *policy-name*.

  NONE
  The terminal user is not authorized to add or remove the specified policy *policy-name*. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to manage the specified policy *policy-name*.

  CONTROL
  Same as UPDATE.

Other user-related policy profiles

This final section on user-related controls describes the remaining settings. The main controls are the controls for setting the name, installation data, and class authorizations (CLAUTH).

Table 22. Profiles used for user settings. The entries in this table reflect the keywords that are specified on the **ADDUSER** and **ALTUSER** commands

Command	Keyword	Profile
ADDUSER ALTUSER	ADD/DEL CATEGORY	C4R.USER.CATEGORY.*category.owner.userid*
ADDUSER ALTUSER	(NO)CLAUTH	C4R.USER.CLAUTH.*class.owner.userid*
ADDUSER ALTUSER	(NO)DATA	C4R.USER.INSTDATA.*owner.userid*
ADDUSER		C4R.USER./INSTDATA.*owner.user*
ADDUSER ALTUSER	(NO)MODEL	C4R.USER.MODEL.*owner.userid*
ADDUSER ALTUSER	NAME	C4R.USER.NAME.*owner.userid*
ADDUSER ALTUSER	(NO)SECLABEL	C4R.USER.SECLABEL.*seclabel.owner.userid*
ADDUSER ALTUSER	(NO)SECLEVEL	C4R.USER.SECLEVEL.*seclevel.owner.userid*
ADDUSER ALTUSER	(NO)WHEN	C4R.USER.WHEN.*owner.userid*
ALTUSER	PWCLEAN	C4R.USER.PWCLEAN.*owner.userid*
ALTUSER	PWCONVERT	C4R.USER.PWCONVERT.*owner.userid*

The following paragraphs describe the remaining policy profiles that are supported by zSecure Command Verifier.

At the moment, there is only limited support for SECLABEL and SECLEVEL. It is possible to control assignment of these two settings, but it is not possible to control removal of the settings.

- C4R.USER.CATEGORY.*category.owner.userid*

  This profile can be used to control assignment of security categories. Normally, RACF administrators can assign their own CATEGORY to other users in their scope. Security categories can be used as extra method of preventing access to resources. Users must have at least all the security categories that are assigned to the resource. The current profile allows control over the assignment and removal of a CATEGORY to a user.

  No profile found
  This control is not implemented. Administrators who are assigned *category* can assign and remove this CATEGORY to users within their scope.

  NONE
  Assignment and removal of the CATEGORY *category* to user *userid* is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

  READ
  System-SPECIAL users can assign and remove the *category* to this userid.

UPDATE
Administrators who are assigned *category* can assign this value to other users within their scope. They also have the authority to remove *category*.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.USER.CLAUTH.*class.owner.userid*

This profile can be used to control which users in your system can be given the authority to define new user IDs and profiles in the specified general resource classes. Normally, users with CLAUTH for a class, can pass their authorization on to other users. The current profile can be used to prevent this case. The access levels that can be used for this profile are given as follows.

No profile found
This control is not implemented. Users with CLAUTH can pass on their authorization to other users in the organization. Also, users with System-SPECIAL can assign CLAUTH for all classes to all users.

NONE
Delegating CLAUTH for CLASS *class* to user *userid* is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

READ
Terminal users can remove the CLAUTH for *class* from users within their scope.

UPDATE
Terminal users with CLAUTH for CLASS *class* can pass their authorization to user *userid*. This method is the standard RACF authorization.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.USER.INSTDATA.*owner.userid*

This profile is used to control the authorization to change installation data of a user. Normally this authorization is already restricted to the owner of the profile, and people with **group-** SPECIAL authorization. This profile implements further restrictions.

The INSTDATA policy profile can also include a reference to the format required for the installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate set of format specification policy profiles. Format specification policy profiles or short format profiles use names similar to the following name:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

Multiple format profiles can be used to specify different parts of the installation data of the RESOURCE profile. For a complete description of the format profiles, see “Installation data field format specification” on page 193.

The access levels that can be used for the INSTDATA profile are given as follows.

No profile found
This control is not implemented. All RACF authorized users can change the installation data of users within their control.

NONE
Specifying installation data is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

READ
Specifying installation data on the **ADDUSER** command is allowed. Changing the value afterward through the **ALTUSER** command is not allowed.

UPDATE
Changing the installation data is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

The optional value that is specified by **APPLDATA** is described as follows:

Format-Name
The name of the format that must be used for the installation data of the *userid* The *Format-Name* is used to locate the appropriate set of format profiles.

- C4R.USER./INSTDATA.*owner.userid*

This profile is used to control the authorization to set the default installation data of a USER. The /INSTDATA policy profile can also include a reference to the format that is required for the default installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate set of format specification policy profiles. Format specification policy profiles or short format profiles use names that are similar to the following name:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

Multiple format profiles can be used to specify different parts of the installation data of the RESOURCE profile. For a complete description of the format profiles, see "Installation data field format specification" on page 193. The following lists the access levels that can be used for the /INSTDATA profile.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied.

READ
The INSTDATA rules are extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied.

The value that is specified by APPLDATA is described as follows:

Format-Name
The name of the format that must be used for the installation data of the *userid* The *Format-Name* is used to locate the appropriate set of format profiles.

- C4R.USER.MODEL.*owner.userid*

The model data set name is used by RACF when a new data set profile that starts with this `userid` is defined. When the new data set profile is defined, the model data set name that is specified is prefixed by the `userid`. RACF allows each user to specify the name of the model profile. User data set modeling is only used if it is activated through SETROPTS. zSecure Command Verifier allows control over the authority to select which data set profile must be used as model. The C4R.*class.*TYPE.*type.profile* profile that is described in "Other policy profiles and access level descriptions" on page 187 allows control over the definition of MODEL data sets themselves.

No profile found
This control is not implemented. All users can select their own model data set. The model data set is only used of MODEL(USER) is activated in SETROPTS.

NONE
Selection of the user MODEL data set name is not allowed.

READ
The MODEL can be specified on the **ADDUSER** command. It is not possible to change it later by the **ALTUSER** command.

UPDATE
Setting, changing, and removing the MODEL specification is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.USER.NAME.*owner.userid*

This profile can be used to control changing the NAME sometimes known as the PGMRNAME of a user ID. The main application of this policy is to prevent users from changing their own **NAME** field. The access levels that can be used for this profile are given as follows.

No profile found
This control is not implemented. All users can change their own NAME. RACF administrators and users can change the NAME of all user IDs under their control.

NONE
Specifying the NAME is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

READ
Specifying a NAME on the **ADDUSER** command is allowed. Changing the NAME through the **ALTUSER** command is not allowed.

UPDATE
Changing the NAME of the user ID is accepted.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.USER.SECLABEL.*seclabel.owner.userid*

This profile can be used to control assignment of Security Labels. Normally, RACF administrators can assign their own SECLABEL to other users in their scope. This label is only the default Security Label. Users can choose their SECLABEL during LOGON process, provided they have access to the defined SECLABEL.

At the moment zSecure Command Verifier has no policy profile that controls the complete removal of a SECLABEL. Administrators can remove the assigned security label from any user within their scope.

No profile found
This control is not implemented. Administrators with access to a *seclabel* can assign this value as the default SECLABEL for users within their scope.

NONE
Assignment of the SECLABEL *seclabel* to user *userid* is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

READ
System special users can assign the *seclabel* to this userid.

UPDATE
Administrators with access to a *seclabel* can assign this value as the default SECLABEL for users within their scope.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.USER.SECLEVEL.*seclevel.owner.userid*

This profile can be used to control assignment of security levels. Normally, RACF administrators can assign SECLEVELs up to their own SECLEVEL to other users in their scope. Security levels can be used as extra method of preventing access to resources. Users must have at least the same security level as the level assigned to the resource. The zSecure Command Verifier policy profile allows control over the assignment of a SECLEVEL to a user. Only the exact name of the *seclevel* is used in the verification process. The corresponding numeric value is not evaluated. Also, assignment of another *seclevel* with a lower value is only controlled by the zSecure Command Verifier policy profile corresponding to that particular *seclevel*.

At the moment zSecure Command Verifier has no policy profile that controls the removal of a SECLEVEL. Administrators can remove the assigned security level from any user within their scope.

No profile found
This control is not implemented. Administrators who are assigned *seclevel* can assign this SECLEVEL to users within their scope.

NONE
Assignment of the SECLEVEL *seclevel* to user *userid* is not allowed. The command is rejected. This setting applies both to the **ADDUSER** and the **ALTUSER** command.

READ
System-SPECIAL users can assign the *seclevel* to this `userid`.

UPDATE
Administrators who are assigned *seclevel* can assign this value to other users within their scope.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- `C4R.USER.WHEN.`*owner.userid*

This single policy profile controls both the setting of the WHEN(DAYS) and the WHEN(TIME) specification for user IDs. These two options control which days of the week and which hours of the day a user ID can log on. This option applies only to interactive work, and only to the exact time and day of the LOGON itself.

No profile found
This control is not implemented. WHEN(DAYS) and WHEN(TIME) can be specified.

NONE
Specification of LOGON restrictions is not allowed.

READ
Same as NONE.

UPDATE
Specification and removal of LOGON restrictions is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- `C4R.USER.PWCLEAN.`*owner.userid*

This profile can be used to control cleanup of the password history for the user. The following access levels can be used for this policy profile:

No profile found
This control is not implemented. Any System-Special user can clean up the password and phrase history.

NONE
Cleanup of the password and phrase history is not allowed.

READ
Same as NONE.

UPDATE
Cleanup of the password and phrase history is allowed. RACF still requires the terminal user to have the System-Special attribute.

CONTROL
The control is not implemented for this terminal user. RACF still requires the terminal user to have the System-Special attribute.

- `C4R.USER.PWCONVERT.`*owner.userid*

This profile can be used to control conversion of the current password and history entries for the user. The following access levels can be used for this policy profile:

No profile found
This control is not implemented. Any System-Special user can convert the password and password history.

NONE
Conversion of the current password and password history is not allowed.

READ
Same as NONE.

UPDATE
Conversion of the current password and password history is allowed. RACF still requires the terminal user to have the System-Special attribute.

CONTROL
The control is not implemented for this terminal user. RACF still requires the terminal user to have the System-Special attribute.

Profiles for managing groups

Use the topics in this section to implement policy profiles to manage group-related commands.

Similar to user ID definitions, for group-related commands, several profiles are used. The next sections describe these profiles. For clarity, all the possible keywords and the corresponding profiles are split in several categories. The first section concentrates on those profiles that describe naming conventions for groups and the place in the RACF group hierarchy for new or existing groups. Subsequent sections describe the connections of users to groups and the attributes of groups.

If you want to implement naming conventions for your groups, the profiles in "Profiles to enforce naming conventions for groups" on page 109 must be used. For the place in the RACF hierarchy, the profiles in "Placement of new groups in the RACF hierarchy" on page 113 can be applied. "Policy profiles for group attributes and authorizations" on page 126 describes the group attributes and other group-related settings.

Profiles to enforce naming conventions for groups

Use these guidelines to create policy profiles to set controls for the creation of groups.

For the name and place of a new or existing group, the installation can use naming conventions that are based on the group itself. The first table summarizes the profiles that control the name of the new group. The profiles apply only to the **ADDGROUP** command to create new groups. "Mandatory and default value policy profiles for SUPGRP" on page 114 and "Mandatory and default value policy profiles for OWNER" on page 120 describe mandatory and default values for the superior group and owner. The last table describes the profiles that are used for the verification of the values that are specified by the terminal user.

Table 23. Profiles used for verification of RACF GROUP. The entries in this table reflect the keywords that describe the name of new and deleted groups.

Command	Keyword	Profile
ADDGROUP	*groupname*	C4R.GROUP.ID.=RACUID(n)
ADDGROUP	*groupname*	C4R.GROUP.ID.=RACGPID(n)
ADDGROUP	*groupname*	C4R.GROUP.ID.*group*
DELGROUP	*groupname*	C4R.GROUP.DELETE.*group*

The profiles in the preceding table are used to describe new GROUPs that can be defined. For the GROUP itself, zSecure Command Verifier provides controls that are based on the name of the new group. The authority to change groups is not controlled by name-based rules. This authorization is already sufficiently restricted by the normal RACF scoping rules. The authority to delete groups is also controlled by the normal RACF ownership rules, but an extra control is needed. This authority is implemented by another name-based rule. For defining new groups, the terminal user still needs at least one group-related authorization (join, group-SPECIAL, or direct ownership).

The *groupname*-based controls preceding impose naming conventions on the new group. This first set of profiles is used to control the name of the newly defined group. These profiles are intended to specify which groups can be defined. In general, only one of these profiles is used to specify your naming convention. Other, less specific generic profiles must be used to block the definition of new groups that do not follow your naming convention. Exceptions can then be implemented by the definition of more specific discrete or generic profiles. An example implementation of these profiles is given as follows.

```
C4R.GROUP.ID.=RACGPID(4)       UACC(UPDATE)
C4R.GROUP.ID.TEST*             UACC(NONE) IBMUSER(UPDATE)
C4R.GROUP.ID.*                 UACC(NONE)
```

These profiles ensure that no new groups can be defined, unless the first 4 characters of the new group are the same as any of the groups of the terminal user that defines the new group. An exception is made for GROUPs that start with TEST. These profiles can be defined by the user IBMUSER, and also (according to the first profile) by all users that are connected to a group that also starts with TEST. The third profile is required to stop definition of new groups outside the allowed naming convention. Without the third profile, almost any `groupname` is accepted, either explicitly by the first or second profile, or implicitly by the absence of a matching profile.

- `C4R.GROUP.ID.=RACUID(n)`

  Specifies a special generic policy for the new group. The =RACUID stands for the `userid` of the terminal user. If the `substring(=RACUID,1,`*n*`)` matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used.

  This profile is a discrete policy profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The terminal user ID is not used as naming convention for new groups.

  NONE
  The new groupname is not allowed. The command fails.

  READ
  Same as NONE.

  UPDATE
  The new groupname is accepted.

  CONTROL
  Same as UPDATE.

- `C4R.GROUP.ID.=RACGPID(n)`

  Specifies a special generic policy for the new groupname. The =RACGPID stands for the list of groups to which the terminal user is connected. All the groups of the user are used, independent of the setting of ""list of group access checking"". This profile is used only if the preceding `=RACUID(`*n*`)` profile is not present or does not match. If the `substring(=RACGPID,1,`*n*`)` matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the `userids`.

  This profile is a discrete policy profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The current group of the terminal user is not used as naming convention for new groups.

  NONE
  The new groupname is not allowed. The command fails.

  READ
  Same as NONE.

  UPDATE
  The new group is accepted.

CONTROL
Same as UPDATE.

- C4R.GROUP.ID.*group*

  Specifies which new groups can be created by the terminal user.

  No profile found
  No naming convention is enforced for new groups.

  NONE
  The specified groupname is not allowed. The command fails.

  READ
  Same as NONE.

  UPDATE
  The specified group is allowed to be created.

  CONTROL
  Same as UPDATE.

Deletion of existing groups

Use the C4R.GROUP.DELETE profiles described in this topic to control the authority to delete existing groups.

The authority to delete Group profiles is normally controlled by some form of ownership (either direct or within the scope of a group-SPECIAL attribute) and by system-SPECIAL authorization. Some organizations want to keep strict control over the authority to delete existing groups. Most often, it is because these organizations implemented extra procedures like saving or renaming data sets or interaction with non-RACF information. The policy profiles that are described in this section put more constraints on the authorization to delete groups.

Deleting groups can also be controlled through the =NOCHANGE policy for groups. If a DELETE policy allows deleting the group, the =NOCHANGE policy profile can still reject the command.

- C4R.GROUP.DELETE.*group*

  This profile is used to control which groups within scope can be deleted. When using generic profiles, deletion of groups can also be completely prevented. Only the terminal users who have access through this profile are allowed to delete these groups. This control reduces the normal delete authorization.

  This profile is not verified if RACF already rejected deletion of the group because of syntax errors or insufficient authority.

  The following rules apply for access to the policy profile:

  No profile found
  The control is not implemented. No additional restriction on the delete of the specified group.

  NONE
  The group cannot be deleted. The command is rejected.

  READ
  The group can be deleted only if the terminal user has the System-SPECIAL attribute.

  UPDATE
  The group can be deleted.

  CONTROL
  Same as UPDATE.

- C4R.GROUP.DELETE.=UNIVERSAL

  This profile is used to control the deletion of universal groups. Universal groups do not contain a list of regular users that are connected to the group. Such groups might seem to be unused or empty, while many users are still connected to the group. More steps to change these user profiles are required when you delete universal groups. To prevent accidental creation of referential problems, a policy can be

implemented that restricts the deletion of universal groups. Only terminal users with sufficient access to the policy profile are allowed to delete universal groups. The required access for system-special users is READ, while regular users require at least UPDATE access.

This policy profile is not verified if RACF already rejected deletion of the group because of insufficient authority, or because the group is not empty. RACF determines that a group is not empty if the group shows any connected user or has one or more subgroups.

The following rules apply for access to the policy profile:

No profile found
The control is not implemented. No additional restriction is applied on the deletion of universal groups.

NONE
Universal groups cannot be deleted. The **DELGROUP** command to delete universal groups is rejected.

READ
Universal groups can be deleted only if the terminal user has the system-special attribute. The **DELGROUP** command to delete universal groups is rejected for all other users.

UPDATE
Universal groups can be deleted.

CONTROL
Same as UPDATE.

Prevent all actions against group profiles

Use the C4R.GROUP.=NOCHANGE.*owner.group* profile described in this topic to prevent all changes or actions against a group.

Aside from preventing the deletion of group profiles as described in the previous section, zSecure Command Verifier also provides an option to completely prevent any action against a selected group or range of groups. This can be done through the definition of a =NOCHANGE policy profile for the group. When a =NOCHANGE policy profile has been defined, the following changes are prevented unless the terminal user has sufficient access:

- Changing the attributes of the group.
- Connecting or removing the user to or from any group.
- Deleting the group.
- Granting or removing direct access of the group.
- Defining new users with the group as DFLTGRP.

These changes can also be controlled through individual policy profiles. The advantage of the =NOCHANGE policy profile is that a single policy profile can be used to control all actions related to the group. The =NOCHANGE policy profile can be used to effectively lock or freeze the current definition of the group.

The qualifier =NOCHANGE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The =NOCHANGE policy profile for groups has the following format:

- C4R.GROUP.=NOCHANGE.*owner.group*

  This profile can be used to ensure that a group cannot be changed, deleted, added to or removed from any resource's access control list, and that no users can be connected to or removed from the group. If the policy profile has been defined, only terminal users who have access through this profile are allowed to change these groups. The following access levels are supported:

  No Profile Found
  This control is not implemented. Modification of the target group is not prevented.

  NONE
  The terminal user is not authorized to perform any actions against the target group.

READ
Same as NONE.

UPDATE
The terminal user can modify the target group, provided that it is within the regular RACF scope of the terminal user.

CONTROL
Same as UPDATE.

Placement of new groups in the RACF hierarchy

When a group is created according to the preceding profile, more rules might apply to the place of the new group in the RACF Group hierarchy.

zSecure Command Verifier provides three types of profiles to control this aspect. The Mandatory Value policy profiles enforce a specific OWNER and SUPGRP for the new group. The Default Value profiles provide a value in case the terminal user does not provide such a value, and the last set of profiles verifies that the values that the terminal user specified are acceptable. Subsequent sections describe how these profiles are used together and which values can be automatically supplied. The zSecure Command Verifier policy profiles use the abbreviation SUPGRP for the superior group, in contrast to the RACF commands that use the abbreviation SUPGROUP.

For Mandatory Value policy profiles, the third qualifier consists of an equals sign, followed by the keyword. So for the SUPGRP, the profile has the qualifier =SUPGRP. Table 24 on page 113 describes the Mandatory Value policy profiles.

Table 24. Mandatory Value policy profiles for RACF GROUP place-related command/keywords. The entries in this table reflect the Mandatory Values for keywords that describe the hierarchy of new groups.

Command	Keyword	Profile
ADDGROUP	*group*	C4R.GROUP.=SUPGRP.*group*
ADDGROUP	*group*	C4R.GROUP.=OWNER.*group*

Table 25 on page 113 describes the Default Value profiles that are used if the terminal user did not specify any keywords that control the place in the RACF Group hierarchy. For Default profiles, the third qualifier consists of a forward slash, followed by the keyword. So for the SUPGRP, the profile has / SUPGRP.

Table 25. Profiles used for Default values of RACF GROUP place-related command/keywords. The entries in this table reflect the default values for keywords that describe the hierarchy of new groups.

Command	Keyword	Profile
ADDGROUP	*group*	C4R.GROUP./SUPGRP.*group*
ADDGROUP	*group*	C4R.GROUP./OWNER.*group*

Finally, Table 26 on page 113 describes the profiles that are used to verify acceptability of the terminal user specified values. This table summarizes which profile is used for which keyword or function.

Table 26. Profiles used for verification of RACF GROUP. The entries in this table reflect the keywords that are specified by the terminal user to describe the name and place of new or changed groups.

Command	Keyword	Profile
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP.=RACUID(n)
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP.=RACGPID(n)
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP.=GROUP(n)
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP.*supgrp.group*

Table 26. Profiles used for verification of RACF GROUP. The entries in this table reflect the keywords that are specified by the terminal user to describe the name and place of new or changed groups. *(continued)*		
Command	**Keyword**	**Profile**
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP./SCOPE.*supgrp.group*
ADDGROUP ALTGROUP	SUPGRP	C4R.GROUP.SUPGRP./OWNER.*supgrp.group*
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER.=RACUID(n)
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER.=RACGPID(n)
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER.=GROUP(n)
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER.*owner.group*
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER./SCOPE.*owner.group*
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER./GROUP.*owner.group*
ADDGROUP ALTGROUP	OWNER	C4R.GROUP.OWNER./SUPGRP.*owner.group*

Policy profiles for the Superior Group (SUPGRP)

Aside from the name of a new group, two other important aspects are the place in the RACF hierarchy (= OWNER) and the superior group.

In standard RACF, the terminal user must have JOIN authority in that group, the group must be within the scope of a group-SPECIAL attribute, or the terminal user must own the group. In addition, if the owner is a RACF GROUP, the group must be the same as the SUPGRP. zSecure Command Verifier implements some additional controls on the superior group. The following sections describe how to use the profiles that are listed in the tables in “Placement of new groups in the RACF hierarchy” on page 113.

Mandatory and default value policy profiles for SUPGRP

The first set of profiles controls the superior group (SUPGRP) of the group for the **ADDGROUP** command. Because this first set specifies mandatory or default values, it is not used for the **ALTGROUP** command.

- C4R.GROUP.=SUPGRP.*group*

  This profile is used to specify a mandatory value for the SUPGRP of every newly defined group. It is only used for the **ADDGROUP** command. The superior group that is used, is obtained from the **APPLDATA** field in the profile. It is used to override any terminal user specified value, or added to the command if the terminal user did not specify a value. The SUPGRP value that is obtained by this Mandatory Value profile is not subject to more SUPGRP-related policy profiles.

  The value *group* represents the affected group. This setting allows the specification of exceptions to the general rule. Only the most specific profile is used by zSecure Command Verifier.

  The qualifier =SUPGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  The control is not active for the terminal user. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command. If this process does not yield a valid group, the current connect group of the terminal user is substituted.

  UPDATE
  Same as READ.

CONTROL
The control is not active for the terminal user. No mandatory value is enforced. If the terminal user specified a value for the GROUP, it is used. If no value was specified, the current group of the terminal user is used by RACF.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value policy profiles, this step leads to the odd situation that access NONE has the same net result as access CONTROL.

The values that are accepted for the **APPLDATA** field are given as follows. The terminal user still needs sufficient authority in this group to define new groups. This authorization is not verified in zSecure Command Verifier. Insufficient authority can result in failure of the command by RACF.

BLANK
This value is used to indicate that RACF default processing must be used. RACF uses the current group of the terminal user.

userid
This entry is an invalid entry. Because it is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current group of the terminal user.

group
This *group* is inserted. If the terminal user has insufficient access to this group, the command is rejected by RACF.

=OWNER
Reflects the OWNER as specified (or defaulted) by the OWNER keyword on the command. This value can also be an OWNER value as inserted by zSecure Command Verifier. If the OWNER resolves to the special value =SUPGRP indicating the superior group, the command is rejected.

=MYOWNER
Reflects the OWNERof the terminal user. This value must be a GROUP. All other situations are considered an error. Because these situations are not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current GROUP of the terminal user.

=GROUP(n)
Reflects the first *n* characters of the new GROUP itself. This value must be a GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to allow definition of the GROUP from =RACGPID(*n*) in C4R.GROUP.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to allow definition. In all other situations, the **APPLDATA** value =RACGPID is considered an error, and the current GROUP of the terminal user is used instead.

- C4R.GROUP./SUPGRP.*group*

This profile is used to specify a default value for the SUPGRP in case the terminal user did not specify a SUPGRP on the **ADDGROUP** command. The SUPGRP that is used as default is obtained from the **APPLDATA** field in the profile. The SUPGRP value that is obtained by this Mandatory Value profile is not subject to more SUPGRP-related policy profiles. If the preceding SUPGRP profile is used to provide a value, the /SUPGRP profile is not used.

The qualifier /SUPGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied.

NONE
No action. No default value is supplied. RACF does not provide a value for the SUPGRP. The command is rejected. An installation can use this access level to force the terminal user to explicitly specify a value for the SUPGRP.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied. The current group of the terminal user is used by RACF.

The values that are accepted for the **APPLDATA** field are given as follows. The terminal user needs sufficient authority in this GROUP to define new GROUPs. Insufficient authority can result in failure of the command.

BLANK
This value is used to indicate that RACF default processing must be used. The current GROUP of the terminal user is used.

userid
This entry is an invalid entry. Because this entry is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current GROUP of the terminal user.

group
The *group* is inserted.

=OWNER
Reflects the OWNER as specified (or defaulted) by the OWNER keyword on the command. This value can also be an OWNER value as inserted by zSecure Command Verifier. If the OWNER resolves to the special value =SUPGRP (indicating the superior group), the command is rejected.

=MYOWNER
Reflects the OWNER of the terminal user. This value must be a GROUP. All other situations are considered an error. Because this entry is not caused by incorrect entry by the terminal user, the command is allowed to continue by using the current GROUP of the terminal user.

=GROUP(n)
Reflects the first *n* characters of the new GROUP itself. This value must be a GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

=RACGPID
Reflects the GROUP that was used to allow definition of the GROUP from =RACGPID(*n*) in C4R.GROUP.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to allow definition. In all other situations, the APPLDATA value =RACGPID is considered an error, and the current GROUP of the terminal user is used instead.

Verification of the specified superior group

Use these profiles to control the selection of the superior group for new GROUPs, and a change of the superior group for existing groups.

The following profiles are used to verify the specification of the superior group by the terminal user.

- C4R.GROUP.SUPGRP.=RACUID(*n*)

  Specifies a special generic policy for the SUPGRP in **ADDGROUP** and **ALTGROUP** commands. The =RACUID stands for the terminal user ID. If the substring(=RACUID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the SUPGRP against the user ID.

  This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The terminal user ID is not used as naming convention or restriction for the SUPGRP.

NONE
The specified SUPGRP is not allowed. This decision can be overruled by authorization to profile *supgrp.group* described as follows.

READ
Same as NONE.

UPDATE
The specified SUPGRP is accepted.

CONTROL
Same as UPDATE.

- C4R.GROUP.SUPGRP.=RACGPID(*n*)

  Specifies a special generic policy for the SUPGRP in **ADDGROUP** and **ALTGROUP** commands. The =RACGPID stands for the list of groups to which the terminal user is connected. All the user groups are used, independent of the setting of "list of group access checking". This profile is used only if the preceding =RACUID(*n*) profile is not present or does not match. If the substring(=RACGPID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the userids.

  This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The current group of the terminal user is not used as naming convention or restriction for the SUPGRP.

  NONE
  The specified SUPGRP is not allowed. This decision can be overruled by authorization to profile *supgrp.group* described as follows.

  READ
  Same as NONE.

  UPDATE
  The specified SUPGRP is accepted.

  CONTROL
  Same as UPDATE.

- C4R.GROUP.SUPGRP.=GROUP(*n*)

  Specifies a special generic policy for the SUPGRP in **ADDGROUP** and **ALTGROUP** commands. The =GROUP stands for the group that is being defined or changed. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the SUPGRP against the target GROUP.

  This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile. This profile is used only if =RACUID(*n*) and =RACGPID(*n*) are not present or do not match.

  No profile found
  The first *n* characters of the GROUP are not used as restriction on the SUPGRP of the userid.

  NONE
  The specified SUPGRP is not allowed. This decision can be overruled by authorization to profile *supgrp.group* described as follows.

  READ
  Same as NONE.

  UPDATE
  The specified SUPGRP is accepted.

  CONTROL
  Same as UPDATE.

If any of the above three profiles allow the selected SUPGRP, the next profile is skipped. Processing continues with the /SCOPE and /OWNER policies. If the preceding profiles did not authorize the use of a certain SUPGRP, the next profile is used as alternative authorization method.

- C4R.GROUP.SUPGRP.*supgrp.group*

  This profile is used independently of the three rules previously defined. It can be used to specify exceptions to the generic name-based policies. It controls if *group* can be used as SUPGRP for new groups. For existing groups, it specifies which GROUP can become the new SUPGRP.

  In most situations, you specify *group* from a generic. Explicit profiles can be used to define exceptions for certain groups.

  This profile is not used if any of the previous profiles already allowed the command to continue.

  No profile found
  The control is not implemented. No name-based policy is enforced.

  NONE
  The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The groupname can be used.

  CONTROL
  Same as UPDATE.

Additional policy profiles for the Superior Group

The next profiles are used to define general restrictions on the SUPGRP.

The first profile restricts the superior group to be within the scope of a group-SPECIAL attribute. It effectively disables join authorization and direct ownership of a GROUP as a means to allow creation of new GROUPs. Because normal users usually do not have group-special, all changes to the SUPGRP are considered outside their scope.

The second profile compares the SUPGRP against the OWNER of the group. It can be used to enforce a match, but it also allows exceptions to this generic rule. RACF itself already enforces that if the OWNER is a GROUP, that it must be the same as the SUPGRP.

- C4R.GROUP.SUPGRP./SCOPE.*supgrp.group*

  This profile is used to specify that the superior group of new and existing groups must be within the scope of Group-SPECIAL. The main purpose of this profile is to prevent decentralized administrators from changing the SUPGRP to a group that they do not control.

  The variables *supgrp* and *group* represent the affected group and the specified (=new) SUPGRP of the group. This setting enables specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  For terminal users with group-special or system-special, use of their administrative authority over the new superior group is recorded through the audit-only policy profile.

  – C4R.USESCOPE.*group*

  Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the new superior group for the group. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

  The following access levels are supported for the /SCOPE policy profile:

No profile found
The control is not implemented.

NONE
Only GROUPs within the terminal user scope can be specified as SUPGRP on both the **ADDGROUP** and **ALTGROUP** commands. If any other GROUP is specified, the command is rejected.

READ
Same as NONE.

UPDATE
GROUPs outside the terminal user scope can be used on both the **ADDGROUP** and **ALTGROUP** command. If the terminal user does not have sufficient authority in the specified GROUP, the command is rejected by RACF.

CONTROL
This policy is not in effect for the terminal user.

- C4R.GROUP.SUPGRP./OWNER.*supgrp.group*

  This profile is used to specify that the superior group of new and existing groups must be the same as the OWNER of the group. Terminal users need access to this profile to specify anything but the OWNER as the value for the SUPGRP.

  If the OWNER is changed concurrently in the same **ALTGROUP** command, the new SUPGRP is verified against the new OWNER.

  For new groups, the use of the Mandatory Value policy profile C4R.GROUP.=SUPGRP.*supgrp.group* described previously is preferred. This Mandatory Value policy profile overlays any value that is specified by the terminal user. The current SUPGRP./OWNER profile requires the terminal user to specify the correct value. If the Mandatory Value profile is used, the current profile is skipped. The main purpose of the current profile is to allow certain GROUPs to be exempt from the SUPGRP=OWNER requirement.

  The variables *supgrp* and *group* represent the affected userid and the specified (=new) SUPGRP of the group. This setting permits the specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented.

  NONE
  The SUPGRP for the GROUP must be the same as the OWNER of the GROUP.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to specify a value for the SUPGRP that is different from the current (or new) OWNER of the group.

  CONTROL
  This policy is not in effect for the terminal user.

Policy profiles for the Group Owner

The other piece of information that describes a newly defined GROUP is the OWNER.

The following profiles apply both to the **ADDGROUP** and to the **ALTGROUP** command. In general, the processing for these profiles assumes that your installation's policy is to use GROUPs as owner. The profile /GROUP provides a control that can be used to indicate whether your installation wants to enforce such a policy.

The following description is split into several sets of profiles. The first set is used to specify a mandatory or default value for the OWNER. The second set of profiles is used to describe controls on a specified value

for the OWNER. The final set of three profiles describes general policies that can be used for the OWNER of GROUPs.

Mandatory and default value policy profiles for OWNER

The profiles described in this topic specify the Mandatory and Default Value policy profiles for the OWNER of the new GROUP. These profiles are used only for the **ADDGROUP** command.

- C4R.GROUP.=OWNER.*group*

  This profile is used to specify a mandatory (overriding) value for the OWNER of the newly defined Group profile. It is only used during **ADDGROUP** processing. The OWNER value that is obtained by this Mandatory Value profile is not subject to more OWNER-related policy profiles.

  The qualifier =OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  No action. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command. If the process yields an ID that is not valid, or a non-existing entry, the current group of the terminal user is used instead.

  UPDATE
  Same as READ.

  CONTROL
  The control is not active for the terminal user. No mandatory value is supplied. The value for the OWNER as specified by the terminal user is used in the command.

  Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value profiles, these profiles lead to the odd situation that access NONE has the same net result as access CONTROL.

  The values that are accepted for the **APPLDATA** field are given as follows. The owner can be a user or group.

  BLANK
  zSecure Command Verifier inserts the RACF default (the terminal user) as the explicit value for the OWNER.

  userid
  The user ID found is inserted as the OWNER.

  group
  The specified GROUP is used as OWNER of the new group.

  =SUPGRP
  Reflects the superior group (SUPGRP) as specified (or defaulted) on the command. If this value resolves to the special value =OWNER, indicating the OWNER of the new profile, the command fails.

  =MYOWNER
  The OWNER of the terminal user is inserted as the value for the owner.

  =GROUP(n)
  Reflects the first *n* characters of the new GROUP itself. This value must be a valid user ID or GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

  =RACGPID
  Reflects the GROUP that was used to allow definition of the GROUP from =RACGPID(*n*) in C4R.GROUP.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to allow definition. In

all other situations, the **APPLDATA** value =RACGPID is considered an error, and the current GROUP of the terminal user is used instead.

- C4R.GROUP./OWNER.*group*

  This profile is used to specify a default value for the OWNER of the newly defined Group profile. It is only used during **ADDGROUP** processing. The OWNER that is to be used as the default value is obtained from the **APPLDATA** field in the profile. The OWNER value that is obtained through this Default Value profile is not subject to more OWNER-related policy profiles. If the preceding =OWNER profile is used to provide a value, the /OWNER profile is not used.

  The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No default value is supplied. This setting results in RACF providing a default for the OWNER (=the terminal user itself).

  NONE
  No default value is supplied. RACF does not provide a value for the OWNER. The command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the OWNER.

  READ
  The **APPLDATA** field is extracted and used for the command. If the process yields an ID that is not valid, or a non-existing entry, the current group of the terminal user is used instead.

  UPDATE
  Same as READ.

  CONTROL
  The control is not active for the terminal user. No default value is supplied. Because the terminal user did not specify a value for the OWNER, RACF makes the terminal user the OWNER of the new profile.

  The values that are accepted for the **APPLDATA** field are given as follows. The specified OWNER can be a user ID or GROUP.

  BLANK
  zSecure Command Verifier inserts the RACF default terminal user as the explicit value for the OWNER.

  userid
  The user ID found is inserted as the OWNER.

  group
  The specified GROUP is used as OWNER of the new GROUP.

  =SUPGRP
  Reflects the superior group DFLTGRP as specified or defaulted on the command. If this value resolves to the special value =OWNER indicating the OWNER of the new profile, the command is rejected. See the preceding description at =SUPGRP for details.

  =MYOWNER
  The OWNER of the terminal user is inserted as the value for the owner.

  =GROUP(n)
  Reflects the first *n* characters of the new GROUP itself. This value must be a GROUP. All other situations are considered an error, and the current GROUP of the terminal user is used instead.

  =RACGPID
  Reflects the GROUP that was used to allow definition of the GROUP from =RACGPID(*n*) in C4R.GROUP.ID.=RACGPID(n). This value is only used if =RACGPID(*n*) was used to allow definition. In all other situations, the **APPLDATA** value =RACGPID is considered an error, and the current group of the terminal user is used instead.

Verification of the specified Group Owner

Use these policy profiles to verify group ownership when a new owner is specified in the **ADDGROUP** or **ALTGROUP** command.

RACF restricts the owner only if it is a GROUP. In that case, it must be identical to the SUPGRP. If the new owner is a user ID, RACF does not impose any restrictions. This set of profiles can be used to restrict the choice of new OWNERs. If the use of the specified OWNER is not accepted by any of the three general policy rules, the explicit profile is used.

- C4R.GROUP.OWNER.=RACUID(*n*)

  Specifies a special generic policy for the OWNER in **ADDGROUP** and **ALTGROUP** commands. The =RACUID stands for the terminal user ID. If the substring(=RACUID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the user ID against the OWNER.

  This profile is a discrete profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The userid of the terminal user is not used as naming convention or restriction for the OWNER.

  NONE
  The specified OWNER is not allowed. This decision can be overruled by authorization to profile *owner.group* described as follows.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted.

  CONTROL
  Same as UPDATE.

- C4R.GROUP.OWNER.=RACGPID(*n*)

  Specifies a special generic policy for the OWNER in **ADDUSER** and **ALTUSER** commands. The =RACGPID stands for the list of groups to which the terminal user is connected. All the user groups are used, independent of the setting of "list of group access checking". This profile is used only if the preceding =RACUID(*n*) profile is not present or does not match. If the substring(=RACGPID,1,*n*) matches, this profile is used in preference to other profiles described, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the OWNER.

  This profile is a discrete profile. Only the single digit between parentheses is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  No profile found
  The current group of the terminal user is not used as naming convention or restriction for the OWNER.

  NONE
  The specified OWNER is not allowed. This decision can be overruled by authorization to profile *owner.group* described as follows.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted.

  CONTROL
  Same as UPDATE.

- C4R.GROUP.OWNER.=GROUP(*n*)

  This profile specifies a special generic policy for the OWNER in **ADDGROUP** and **ALTGROUP** commands. The =GROUP stands for the group itself. If you defined multiple of these profiles, only the one with the

lowest value for *n* is used. This profile is only used if =RACUID(*n*) and =RACGPID(*n*) are not present or do not match.

This profile is a discrete policy profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

If the specified owner is accepted, more verifications against the general policies like /SCOPE and /GROUP are performed.

The special value =GROUP represent the affected user profile itself. This profile can be used to enforce a naming convention that states that the first *n* characters of a GROUP must match the first *n* characters of its OWNER.

No profile found
The target GROUP itself is not used as naming convention or restriction for the OWNER.

NONE
The specified OWNER is not allowed. This decision can be overruled by authorization to profile *owner.group*.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

If any of the above three profiles allow the specified OWNER, the next profile rule is skipped. Processing continues with the /SCOPE, /GROUP, and /SUPGRP policies that are described as follows. If the above three profiles did not authorize the use of a certain OWNER, the next profile is used as alternative authorization method.

- C4R.GROUP.OWNER.*owner.group*

The primary purpose of this control is to specify a policy if none of the general group-based policies described previously applies. The variable *owner* represents the new OWNER of the *group*. This setting allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The OWNER as verified by this policy profile is still subjected to more policies /SCOPE, /GROUP, /SUPGRP.

No profile found
This control is not implemented.

NONE
The command is rejected.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

Additional policy profiles for the Group Owner

Aside from the profiles that are intended to enforce a naming convention, it is also possible to implement a policy that is based on the existing RACF group hierarchy. The following profiles allow specification of general rules for the new OWNER. By using more specific or fully qualified profiles, it is possible to specify that some user IDs or GROUPs are exempt from such a restriction.

The following three profile rules are used as an extra set of OWNER policies. If the specified OWNER is accepted by any of the four rules above, it is verified again to comply with the three policies. If it fails any of these additional policies, the command is rejected.

- C4R.GROUP.OWNER./SCOPE.*owner.group*

  This profile is used to control whether the new OWNER as specified by the terminal user must be within the scope of a group-SPECIAL attribute. This case applies both for the **ADDGROUP** command and the **ALTGROUP** command. This profile can prevent the terminal user from giving away group profiles that are within the scope of a group-SPECIAL attribute.

  The variables *group* and *owner* represent the affected GROUP and the new OWNER of the GROUP. This setting allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  Specifying a user ID as the new owner is always considered to be outside the administrative scope of the terminal user.

  For terminal users with group-special or system-special, use of their administrative authority over the new owner is recorded through the audit-only policy profile.

  – C4R.USESCOPE.*group*

  Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the new owner specified for the group. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

  The following access levels are supported for the /SCOPE policy profile:

  No profile found
  The Group-SPECIAL scope of the terminal user is not used to control the new OWNER of Group profiles.

  NONE
  If the specified new OWNER is outside the scope of a group-SPECIAL attribute of the terminal user, the command is rejected.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted, irrespective of the scope of the terminal user.

  CONTROL
  Same as UPDATE.

- C4R.GROUP.OWNER./GROUP.*owner.group*

  The profile is used to control whether the specified owner must be a RACF group or not. This profile is verified independently of the other profiles. If either the =OWNER or the /OWNER profile is used, that this policy rule is bypassed.

  The variables *group* and *owner* represent the affected GROUP and the new OWNER of the GROUP. This setting allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /GROUP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  This control is not implemented. The specified owner can be a group and a user.

  NONE
  If the specified owner is an existing RACF group, the command is accepted. In all other situations, the command is rejected.

READ
Same as NONE.

UPDATE
The specified owner is accepted even if it does not represent an existing group. If the specified owner is not a valid entry, the command is rejected by RACF.

CONTROL
Same as UPDATE.

- C4R.GROUP.OWNER./SUPGRP.*owner.group*

  This profile is used to control whether the OWNER as specified by the terminal user must be the same as the SUPGRP of the GROUP. This profile applies both for the **ADDGROUP** command and the **ALTGROUP** command.

  The values *group* and *owner* represent the affected GROUP and the new OWNER of the GROUP. This setting allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

  The qualifier /SUPGRP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  This control is not implemented. The specified OWNER can be different from the current SUPGRP.

  NONE
  The specified new OWNER must be the same as the current or new SUPGRP.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted, irrespective of the value of the SUPGRP.

  CONTROL
  Same as UPDATE.

Implementation of a new group policy

Use these guidelines to plan the organization of policy profiles that are not part of the RACF group hierarchy.

The previous sections describe the profiles that are used in the decision process for the GROUP and the place in the RACF group hierarchy. These profiles allow great flexibility in the specification of the GROUPs that might or might not be defined. As an example you might want to consider the following organization:

- Central administrators can define all groups.
- Decentralized administrators must define groups only for their own department.
- Departments can be recognized by the RACF group structure (ownership).
- All group profiles must be owned by a RACF group, according to the departmental structure.
- A group naming convention is used where the first 3 characters of the group are the same as the first 3 characters of the department name.

For these organizations, the following profiles can be implemented.

C4R.group.id.* uacc(none) sysadmin(update)
This profile ensures that only system administrators are allowed to define new group profiles outside the regular naming conventions.

C4R.group.id.=racuid(3) uacc(update)
This profile allows all decentralized administrators to define new groups that have as first 3 characters the same characters as the decentralized administrator. Implementation of this policy from the =RACGPID(3) profile is not as effective. All the groups of the terminal user would be used as the naming convention. It is not guaranteed that the terminal user is not connected to a functional group of another department, which would have a different prefix.

C4R.group.delete. uacc(none) sysadmin(update)**
This profile ensures that only the central system administrators are allowed to delete existing groups.

C4R.group.=supgrp. uacc(update) sysadmin(control) appldata('=myowner')**
This profile specifies that independent of what any decentralized administrator specifies, the newly defined group are always placed below the same group that owns the decentralized administrator itself. Central system administrators must specify a SUPGRP because this control does not apply to them. However, see the next profile.

C4R.group./supgrp. uacc(none) sysadmin(update) appldata('DEPTS')**
If the central system administrator does not specify a SUPGRP for new groups, the group is assigned to the group called DEPTS.

C4R.group.=owner. uacc(update) sysadmin(control) appldata('=myowner')**
This profile ensures that the OWNER of the new /GROUP profile is the same as the OWNER of the decentralized administrator. Again, this control does not apply to the central system administrators. The next profile is especially defined for their usage.

C4R.group./owner. uacc(none) sysadmin(update) appldata('=supgrp')**
The use of =SUPGRP as the APPLDATA value ensures that if no value is specified for the OWNER, the OWNER is completed by zSecure Command Verifier to be the same as the SUPGRP for the new GROUP.

Implementation of an existing group policy

Use these scenarios to implement policy profiles that set up more controls in the specification of existing groups.

You can set up a policy profile that determines how existing groups must be handled. The following list contains more rules for the Existing Group policy.

- Central administrators can modify all groups.
- Central administrators can specify any user or group as owner.
- Decentralized administrators must change only the owner within their own department.

For these organizations, the following profiles can be implemented.

C4R.group.supgrp./scope. uacc(none) sysadmin(control)**
This profile ensures that only system administrators are allowed to change the superior group to all values. The decentralized administrators can specify only groups that are within their scope of control.

C4R.group.owner./scope. uacc(none) sysadmin(control)**
This profile ensures that only system administrators have unrestricted authorization to change the OWNER of existing groups. Decentralized administrators can change the OWNER only within their scope. They cannot give away any of their groups. Normal users cannot change the OWNER of any groups that they own because they do not have Group-SPECIAL: everything is outside their scope.

Policy profiles for group attributes and authorizations

Use the policy profiles in this summary list to implement controls for group attributes and authorizations.

For all user to group connect attributes and authorization, see “CONNECT management” on page 130. The commands, keywords, and profiles are summarized in the following table. Detailed descriptions for each profile are provided in the sections that follow the table.

Table 27. Profiles used for RACF attributes. The entries in this table reflect the keywords that are specified on the **ADDGROUP** and **ALTGROUP** commands.

Command	Keyword	Profile
ADDGROUP		C4R.GROUP.=ATTR.*owner.group*
ADDGROUP	*UNIVERSAL*	C4R.GROUP.ATTR.UNIVERSAL.*owner.group*
ADDGROUP ALTGROUP	*(NO)TERMUACC*	C4R.GROUP.ATTR.TERMUACC.*owner.group*

Table 27. Profiles used for RACF attributes. The entries in this table reflect the keywords that are specified on the **ADDGROUP** and **ALTGROUP** commands. *(continued)*		
Command	**Keyword**	**Profile**
ADDGROUP ALTGROUP	*(NO)DATA*	C4R.GROUP.INSTDATA.*owner.group*
ADDGROUP		C4R.GROUP./INSTDATA.*owner.group*
ADDGROUP ALTGROUP	*(NO)MODEL*	C4R.GROUP.MODEL.*owner.group*

Mandatory attributes for new groups

By using the Mandatory policy profile for group attributes, an installation can specify that new groups must always have certain attributes, irrespective of the keywords that are used on the **ADDGROUP** command.

The most obvious use for this function is setting the NOTERMUACC value. The Mandatory Attribute policy profile and the applicable access level are described as follows.

- C4R.GROUP.=ATTR.*owner.group*

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The mandatory attributes do not apply for the terminal user.

  READ
  The APPLDATA of the Mandatory Value policy profile is used as the list of attributes for the new group.

  UPDATE
  Same as READ.

  CONTROL
  The control is not active for the terminal user. No mandatory value is supplied. The attributes as specified by the terminal user are used in the command.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value profiles, this case leads to the odd situation that access NONE has the same net result as access CONTROL.

The **APPLDATA** field of the mandatory policy profile specifies a list of group attributes. The following list contains group attributes that are recognized.

- TERMUACC and NOTERMUACC
- UNIVERSAL

It is not possible to use abbreviations for the attributes. If multiple attributes must be assigned, the individual attributes must be separated by a single comma without any intervening blanks; for example:

```
TERMUACC,UNIVERSAL
```

Group attributes and access level descriptions

The following information describes the access levels that are used to control which keywords and which values can be used.

In general, the access level that is required is READ to specify the value that RACF by default applies to new GROUPs, while UPDATE is required to set or modify the attribute. Also, for the **ADDGROUP** commands,

zSecure Command Verifier does not check the default value that is used by RACF. However, the non-default value is checked similar to the checking done for the **ALTGROUP** command.

- C4R.GROUP.ATTR.UNIVERSAL.*owner.group*

  This profile controls the definition of RACF Universal GROUPs. Universal GROUPs are user GROUPs that do not have complete membership information that is stored in their Group profiles. The benefit of using Universal GROUPs, is that RACF does not impose a limit on the number of regular user IDs connected to the GROUP.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to create Universal GROUPs.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to create Universal GROUPs.

  CONTROL
  The control is not implemented for the terminal user.

  In all the preceding situations, the terminal user needs sufficient RACF authorization to create the GROUP in the first place. The GROUP must comply with the implemented zSecure Command Verifier policies.

- C4R.GROUP.ATTR.TERMUACC.*owner.group*

  This profile controls the setting of the (NO)TERMUACC attribute of new and existing groups. TERMUACC specifies that during terminal authorization checking, RACF allows any user in the group access to a terminal based on the universal access authority (UACC) for that terminal. TERMUACC is the default value.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  The terminal user is not authorized to specify either keyword on the **ALTGROUP** command. The TERMUACC setting is allowed (defaulted) on the **ADDGROUP** command.

  READ
  The terminal user is authorized to explicitly specify the TERMUACC attribute on the **ALTGROUP** command. This setting allows reset of the attribute to its default state.

  UPDATE
  The terminal user is authorized to specify both keywords on the **ALTGROUP** command. This setting allows regular maintenance of these attributes.

  CONTROL
  The control is not implemented for the terminal user. The terminal user is authorized to specify both keywords on the **ADDGROUP** and **ALTGROUP** command. This setting allows regular maintenance of the TERMUACC setting.

- C4R.GROUP.INSTDATA.*owner.group*

  This profile is used to control the authorization to change the installation data of a GROUP. Normally this case is already restricted to the owner of the profile, and people with group-**SPECIAL** authorization. This profile implements further restrictions.

  The INSTDATA policy profile can also include a reference to the format required for the installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate (set of) format specification policy profiles. Format specification policy profiles (or short format profiles) use names similar to the following name:

  ```
  C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
  ```

Multiple format profiles can be used to specify different parts of the installation data of the Group profile. For a complete description of the format profiles, see "Installation data field format specification" on page 193.

The access levels that can be used for this profile are given in the following list.

No profile found
This control is not implemented. All RACF authorized users can change the installation data of groups within their control.

NONE
Specifying installation data is not allowed. The command is rejected. This setting applies both to the **ADDGROUP** and the **ALTGROUP** command.

READ
Specifying installation data on the **ADDGROUP** command is allowed. Changing the value afterward through the **ALTUSER** command is not allowed.

UPDATE
Changing the installation data is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

The optional value that is specified by APPLDATA.

format
The name of the format that must be used for the installation data of the *group* The *format* name is used to locate the appropriate set of format profiles.

- C4R.GROUP./INSTDATA.*owner.group*

This profile is used to control the authorization to set the default installation data of a GROUP. The /INSTDATA policy profile can also include a reference to the format that is required for the default installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate set of format specification policy profiles. Format specification policy profiles or short format profiles use names that are similar to the following name:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

Multiple format profiles can be used to specify different parts of the installation data of the RESOURCE profile. For a complete description of the format profiles, see "Installation data field format specification" on page 193. The following lists the access levels that can be used for the /INSTDATA profile.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied.

READ
The INSTDATA rules are extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied.

The value that is specified by APPLDATA is described as follows:

Format-Name
The name of the format that must be used for the installation data of the *userid* The *Format-Name* is used to locate the appropriate set of format profiles.

- C4R.GROUP.MODEL.*owner.group*

The model data set name is used by RACF when a new data set profile that starts with this group is defined. The model data set specified is prefixed by the group. Group data set modeling is only used if it is activated by SETROPTS. zSecure Command Verifier allows control over the authority to select which data set profile must be used as model. The C4R.*class*.TYPE.*type.profile* profile that is described in "Other policy profiles and access level descriptions" on page 187 allows control over the definition of MODEL data sets themselves.

No profile found
This control is not implemented.

NONE
Selection of the group MODEL data set name is not allowed.

READ
The MODEL can be specified on the **ADDGROUP** command. It is not possible to change it later on the **ALTGROUP** command.

UPDATE
Setting, changing, and removing the MODEL specification is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

CONNECT management

For the connections of users to groups (group membership) a distinction must be made between defining new connections with attributes and changing existing connect attributes.

In zSecure Command Verifier, two sets of controls are implemented for the two situations. Naming conventions and similar policies are enforced for new connections. For both new and existing connections, policies can be enforced about the connect authorizations and attributes.

Another issue that needs attention is whether the installation must control the user to group connections from a user viewpoint, or from a group viewpoint. In zSecure Command Verifier, the policy profiles have qualifiers for both the group and the user. However, to simplify policy implementation, implement only one of these qualifiers through generic characters. Determining the controlling profile can be more complex if you use both types of generics together.

"New CONNECTs" on page 132 describes the details of the zSecure Command Verifier policy profiles for new CONNECTs. Subsequent sections describe the requirements for existing CONNECTs, and the CONNECT authorization and attributes.

Authority to connect yourself

Use these guidelines to implement self-authorization profiles to enforce the separation of responsibilities between administrators.

This function is required by many organizations who want to enforce a strict separation of responsibilities between the security administrator and the data or application administrator. System security policies often specify that security administrators must not have access to application resources. In standard RACF, users with System or Group-SPECIAL can change any profiles under their control. So, even if security administrators currently do not have access to application resources, it is easy for them to obtain access. In this case, the access can be gained by connecting them to a GROUP that has access. Some organizations analyze SMF data to report on those administrators that connect or remove themselves from certain groups. In zSecure Command Verifier, several policies are available to prevent security administrators from modifying the list of GROUPs to which they are connected.

Table 28. Profiles used to control self-authorization. The entries in this table reflect the keywords that describe the ACL entries or CONNECTs.

Command	Keyword	Profile
PERMIT	*userid*	C4R.*class*.ACL.=RACUID.*access.profile*

Table 28. Profiles used to control self-authorization. The entries in this table reflect the keywords that describe the ACL entries or CONNECTs. *(continued)*		
Command	**Keyword**	**Profile**
PERMIT	*group*	C4R.*class*.ACL.=RACGPID.*access.profile*
CONNECT	*userid*	C4R.CONNECT.ID.*group.*=RACUID
REMOVE	*userid*	C4R.REMOVE.ID.*group.*=RACUID

The preceding profiles are applicable only if the *userid* is the terminal user or if the *group* is any of the connect groups of the user. If this situation applies, zSecure Command Verifier uses the preceding profiles in preference to the CONNECT profiles described in "New CONNECTs" on page 132. A detailed description of the last two profiles and the supported access levels is given as follows. For more information about the profiles for PERMIT, see "Policy profile selection for self-authorization" on page 151.

- C4R.CONNECT.ID.*group.*=RACUID

  This profile is used to specify the authority of terminal users to CONNECT themselves to *group*. If a generic pattern is used for the *group*, the profile describes the authority to connect themselves to any *group*. This profile is primarily intended to prevent administrators from increasing their own authority through CONNECTs to functional groups with high access to application or system resources. This profile can be used most efficiently in combination with the policy profile for =RACGPID for modifying the access list, as described in "Policy profile selection for self-authorization" on page 151.

  No profile found
  The control is not implemented. All terminal users can CONNECT themselves to any GROUP within their control.

  NONE
  The terminal user is not allowed to CONNECT him/herself to *group*, even if this GROUP is within scope.

  READ
  Same as NONE

  UPDATE
  The terminal user is allowed to CONNECT him/herself to *group* if the GROUP is within scope.

  CONTROL
  Same as UPDATE

- C4R.REMOVE.ID.*group.*=RACUID

  This profile is used to specify the authority of terminal users to REMOVE themselves from *group*. If a generic pattern is used for the *group*, the profile describes the authority to remove themselves from any *group*. This profile is primarily intended for those situations where a specific GROUP is used to reduce the access that people with the (Group-)OPERATIONS attribute might have. This profile can be used most efficiently in combination with the policy profile for =RACGPIDfor modifying the access list, as described in "Policy profile selection for self-authorization" on page 151.

  No profile found
  The control is not implemented. All terminal users can REMOVE themselves from any GROUP within scope.

  NONE
  The terminal user is not allowed to REMOVE him/herself from *group*, even if this GROUP is within scope.

  READ
  Same as NONE

  UPDATE
  The terminal users are allowed to REMOVE themselves from *group*, only if the GROUP is within scope.

CONTROL
Same as UPDATE

New CONNECTs

Use these connection-related policy profiles to control which user-to-group connects can be created.

The entries in this table reflect the user and group of newly defined connections.

Table 29. Profiles used for RACF connection-related command/keywords

Command	Keyword	Profile
CONNECT	GROUP(*group*)	C4R.CONNECT.ID.=USERID(*n*)
CONNECT	*userid* GROUP(*group*)	C4R.CONNECT.ID.*group.userid*
CONNECT	*userid* GROUP(*group*)	C4R.CONNECT.ID./USRSCOPE.*group.userid*
CONNECT	*userid* GROUP(*group*)	C4R.CONNECT.ID./GRPSCOPE.*group.userid*
CONNECT	*userid* GROUP(*group*)	C4R.CONNECT.ID.=DSN.*group.userid*
REMOVE	*userid* GROUP(*group*)	C4R.REMOVE.ID.*group.userid*

Authority to create CONNECTs

The first part of the CONNECT policies concerns itself with the rules for creating new CONNECTs. RACF looks only at the authorization of the terminal user in the GROUP. The user ID that is to be connected is irrelevant.

In zSecure Command Verifier, more controls are implemented that allow an installation to control authorizations that are based on the user ID as well.

- C4R.CONNECT.ID.=USERID(*n*)

  This profile can be used to implement a general naming convention-based policy on USER to GROUP CONNECTs. The qualifier =USERID(*n*) stands for the first *n* characters of the user ID (or GROUP). The first *n* characters of the GROUP are matched against the first *n* characters of the user ID. If they match, this profile is used to determine whether the new CONNECT can be created. If you defined multiple of these profiles, only the one with the smallest numerical value for *n* is used.

  This profile must be a discrete profile. The number *n* must be specified as a single digit 1 - 8.

  Note: The profile =USERID(*n*) would be functionally equivalent to =GROUP(*n*). In zSecure Command Verifier, only the =USERID(*n*) profile is implemented.

  No profile found
  This control is not implemented. The first characters of the GROUP are not matched against the characters of the user ID.

  NONE
  The terminal user is not authorized to CONNECT USERs to GROUPs that start with the same characters.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to CONNECT USERs to like-named (first *n* characters) GROUPs.

  CONTROL
  The control is not implemented for the terminal user. No general naming convention is used for USER to GROUP CONNECTs.

- C4R.CONNECT.ID.*group.userid*

This profile can be used to implement other naming convention-based policies. It can also be used as a way of specifying exceptions to the general =USERID(*n*) policy.

No profile found
This control is not implemented. The *userid* can be connected to the *group*.

NONE
The terminal user is not authorized to CONNECT *userid* to *group*.

READ
Same as NONE.

UPDATE
The terminal user is authorized to CONNECT the *userid* to *group*.

CONTROL
Same as UPDATE.

Additional policy controls for new CONNECTs

After a new connection is approved by the preceding policy rules, the new connection can be subjected to more controls. Policies that are based on RACF Group-SPECIAL scope and based on naming conventions can be implemented as well.

The policies that can be implemented are described as follows.

- C4R.CONNECT.ID./USRSCOPE.*group.userid*

  This profile is used to control whether USERs outside the Group-SPECIAL scope of the terminal user can be connected to the *group*.

  The qualifier /USRSCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  For terminal users with group-special or system-special, connecting a user within their administrative scope is recorded through the audit-only policy profile.

  – C4R.USESCOPE.*group*

  Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the user that is connected to the group. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

  The following access levels are supported for the /SCOPE policy profile:

  No profile found
  This control is not implemented. The **Group-**SPECIAL scope of the terminal user is not considered for the *userid* that is to be CONNECTed to the *group*.

  NONE
  The terminal user is not authorized to CONNECT users outside its scope to the *group*.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to CONNECT users outside its scope to the *group*.

  CONTROL
  Same as UPDATE.

- C4R.CONNECT.ID./GRPSCOPE.*group.userid*

  This profile is used to control whether GROUPs outside the Group-SPECIAL scope of the terminal user can be connected to this *userid*. This profile partially overlaps with the normal RACF authorization requirements. The main difference is that the zSecure Command Verifier policy does not take CONNECT authorization and direct ownership of the GROUP into account. Only Group-SPECIAL is considered to determine the authorization.

The qualifier /GRPSCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

For terminal users with group-special or system-special, connecting a user to a group within their administrative scope is recorded through the audit-only policy profile

– C4R.USESCOPE.*group*

Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the group to which the user is connected. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

The following access levels are supported for the /SCOPE policy profile:

No profile found
This control is not implemented. The **Group-**SPECIAL scope of the terminal user is not considered for the *group* to which the USER is CONNECTed.

NONE
The terminal user is not authorized to CONNECT users to GROUPs outside its scope.

READ
Same as NONE.

UPDATE
The terminal user is authorized to CONNECT users to GROUPs outside its scope.

CONTROL
Same as UPDATE.

- C4R.CONNECT.ID.=DSN.*group.userid*

This profile is used to control whether USERs can be connected to GROUPs that are used as the High-Level Qualifier (HLQ) of data sets.

The qualifier =DSN in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. The fact that the GROUP occurs as HLQ of data sets is not considered.

NONE
The terminal user is not authorized to CONNECT users to GROUPs that are used as HLQ of data sets.

READ
Same as NONE.

UPDATE
The terminal user is authorized to CONNECT users to GROUPs that are used as HLQ of data sets.

CONTROL
Same as UPDATE.

Removal of existing CONNECTs

zSecure Command Verifier also provides a policy to control which users can be removed from a group.

Standard RACF authorization is based on CONNECT authorization and on direct or indirect (from Group-SPECIAL) ownership of the GROUP. In some organizations, removal of a user ID from a group can remove required access authorizations from critical jobs. To prevent inadvertent removal of a user, te following policy can be implemented.

- C4R.REMOVE.ID.*group.userid*

This profile can be used to prevent USERs from being removed from certain GROUPs. Normally, any user with CONNECT authorization or other control over the GROUP can remove all users from the GROUP. Use this policy profile to specify exceptions to the standard authorization to manage CONNECTs.

No profile found
This control is not implemented. The *userid* can be removed from the *group*.

NONE
The terminal user is not authorized to REMOVE *userid* from *group*.

READ
Same as NONE.

UPDATE
The terminal user is authorized to REMOVE the *userid* from *group*.

CONTROL
Same as UPDATE.

Policy profiles for CONNECT attributes and authorizations

The RACF CONNECT command can be used to define new user-to-group connections and to change existing user-to-group connections. It allows specifying a list of users.

However, the CONNECT command does not support a list of values for the other keywords on the command. The implication is that all connections are assigned the same OWNER and attributes as specified on the command. If a keyword value is not allowed for one or more of the connections, the entire command is rejected, and a message similar to `C4R551E GrpSpecial attribute not allowed, command terminated` is issued. If you use Mandatory or Default Value policy profiles, a message similar to `C4R690E Cannot assign OWNER value, please split command` might be issued.

The next two tables summarize the policy profiles that are used for new and existing CONNECTs. The first table provides all the Mandatory and Default Value profiles, which are used mainly for new user-to-group CONNECTs. The second table provides all the other profiles, which are used when you create CONNECTs or when you modify existing CONNECTs. The main purpose of these policies is to control which authorizations and attributes are used for user-to-group CONNECTs. The most important attribute is probably the group-SPECIAL attribute.

Mandatory and Default value policies are intended for new user-to-group connections. For existing connections, keyword values that are not specified by the terminal user are unaffected: this ensures that non-standard values assigned by an authorized administrator are not unintentionally reset by an other administrator. If the CONNECT command specifies a mixture of new and existing connections, all connections are treated as being new and the Mandatory and Default value policies are evaluated for all connections. This might result in resetting a non-standard value for the OWNER, UACC, or AUTH that has been set by an authorized administrator, back to the value as required by the policy.

Table 30. Profiles used for RACF connection-related command/keywords. The entries in this table reflect the keywords that are specified on the **ADDUSER**, **ALTUSER**, and **CONNECT** commands.

Command	Keyword	Profile
CONNECT	OWNER	C4R.CONNECT.=OWNER.*group.userid*
CONNECT	OWNER	C4R.CONNECT./OWNER.*group.userid*
CONNECT ADDUSER	AUTH(*auth*)	C4R.CONNECT.=AUTH.*group.userid*
CONNECT ADDUSER	AUTH(*auth*)	C4R.CONNECT./AUTH.*group.userid*
CONNECT ADDUSER	UACC(*uacc*)	C4R.CONNECT.=UACC.*group.userid*
CONNECT ADDUSER	UACC(*uacc*)	C4R.CONNECT./UACC.*group.userid*

Table 31. Profiles used for RACF attributes and authorizations. The entries in this table reflect the keywords that are specified on the **CONNECT** command.

Command	Keyword	Profile
CONNECT	OWNER(*owner*)	C4R.CONNECT.OWNER.*owner.group.userid*
CONNECT ADDUSER ALTUSER	AUTH(*auth*)	C4R.CONNECT.AUTH.*auth.group.userid*
CONNECT ADDUSER ALTUSER	UACC(*uacc*)	C4R.CONNECT.UACC.*uacc.group.userid*
CONNECT	SPECIAL	C4R.CONNECT.ATTR.SPECIAL.*group.userid*
CONNECT	OPERATIONS	C4R.CONNECT.ATTR.OPERATIONS.*group.userid*
CONNECT	AUDITOR	C4R.CONNECT.ATTR.AUDITOR.*group.userid*
CONNECT	ADSP	C4R.CONNECT.ATTR.ADSP.*group.userid*
CONNECT	GRPACC	C4R.CONNECT.ATTR.GRPACC.*group.userid*
CONNECT	REVOKE	C4R.CONNECT.ATTR.REVOKE.*group.userid*
CONNECT	RESUME	C4R.CONNECT.ATTR.RESUME.*group.userid*
CONNECT	REVOKE(*date*)	C4R.CONNECT.ATTR.REVOKEDT.*group.userid*
CONNECT	RESUME(*date*)	C4R.CONNECT.ATTR.RESUMEDT.*group.userid*

Mandatory and default value policy profiles for CONNECTs

Several profiles describe the Mandatory and Default Value policy profiles for the OWNER, AUTH, and UACC of a new CONNECT.

These profiles are only used for the **CONNECT** command when you create a user to group CONNECT. Use of the **CONNECT** command to change an existing connection is not subjected to these policy profiles.

- C4R.CONNECT.=OWNER.*group.userid*

  The **APPLDATA** field from the profile is used to specify a value for the connect OWNER of new user to group connections. This profile is only used for the **CONNECT** command, and only for new CONNECTs. If the new CONNECT is created as part of the creation of a new user ID, this profile is not used and RACF uses the owner of the user ID as the OWNER of the CONNECT. The access levels that are used are given as follows.

  The qualifier =OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  This control is not implemented. No action is performed.

  NONE
  No action. No overriding value for the CONNECT OWNER is provided. The user specified value, or the RACF default value is used.

  READ
  The APPLDATA value is inserted as the OWNER of the new CONNECT.

  UPDATE
  Same as READ

  CONTROL
  The control is not implemented for the terminal user. The user specified value, or the RACF default value is retained.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Default Value profiles, this case leads to the odd situation that access NONE has the same net result as access CONTROL.

The following values re the special values that are recognized for the **APPLDATA** field.

BLANK
This value explicitly indicates that the RACF default behavior is to be accepted. The terminal user is inserted as the owner of the user to group connection.

=GROUP
The group part of the CONNECT is to become the owner of the CONNECT profile.

=USERID
The user part of the CONNECT is to become the owner of the CONNECT profile.

value
The specified value is inserted. If the specified value is not an existing RACF userid or GROUP, the current group of the terminal user is used instead.

- C4R.CONNECT./OWNER.*group.userid*

The **APPLDATA** field from the profile is used to specify a value for the connect OWNER of new user to group connections. This profile is only used for the **CONNECT** command, and only for new CONNECTs. If the new CONNECT is created as part of the definition of a new user ID, RACF uses the specified owner of the user ID as the OWNER of the CONNECT.

The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. No action is performed.

NONE
No action. No default value for the CONNECT OWNER is provided. For new connections, this case results in the use of the RACF default, which is to use the terminal user as the OWNER of the new connection.

READ
If the terminal user did not specify a value for the OWNER, the value of the **APPLDATA** is inserted as the OWNER of the new CONNECT.

UPDATE
If the terminal user did not specify a value for the OWNER, the value from the **APPLDATA** field is used instead.

CONTROL
The control is not implemented for the terminal user. If the terminal user does not specify a value, RACF use the terminal user as OWNER. No explicit owner is inserted by zSecure Command Verifier.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Default Value profiles, this case leads to the odd situation that access NONE has the same net result as access CONTROL.

The following values are special values that are recognized for the **APPLDATA** field.

BLANK
This value explicitly indicates that the RACF default behavior is to be accepted. The terminal user is inserted as the owner of the user to group connection.

=GROUP
The group part of the CONNECT is to become the owner of the CONNECT profile.

=USERID
The user part of the CONNECT is to become the owner of the CONNECT profile.

value
The specified value is inserted. If the specified value is not an existing RACF userid or group, the current group of the terminal user is used instead.

- C4R.CONNECT.=AUTH.*group.userid*

This profile is used to specify a mandatory value for the AUTH of new CONNECTs. It is only used for the **ADDUSER** and **CONNECT** command, and only for new CONNECTs. The AUTH value that is used, is obtained from the **APPLDATA** field in the profile. It is used to override any terminal user specified value, or added to the command if the terminal user did not specify a value. The AUTH value that is obtained from this Mandatory Value profile is not subject to more AUTH-related policy profiles.

The value *userid* represents the affected user. This value allows the specification of exceptions to the general rule. Only the most specific profile is used by zSecure Command Verifier. Generic profiles can be used to specify the AUTH of users within the group.

The qualifier =AUTH in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No mandatory value is enforced.

NONE
No action. No mandatory value is enforced.

READ
The **APPLDATA** field is extracted and used for the command. If this process does not yield a valid AUTH level, USE is used instead.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No mandatory value is supplied. If the terminal user specified a CONNECT AUTH, it is used. If no value was specified, RACF uses the value USE.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value policy profiles, this case leads to the odd situation that access NONE has the same net result as access CONTROL.

The values that are accepted for the **APPLDATA** field. The terminal user still needs sufficient authority in the GROUP to assign the specified AUTH level. This authorization is not verified in zSecure Command Verifier. Insufficient authority can result in failure of the command by RACF

auth
Any of the possible connect authorization levels USE, CREATE, CONNECT, JOIN. The value is inserted as the CONNECT authorization for this USER CONNECT.

other
This value is considered an error. The RACF default CONNECT authorization (USE) is used instead.

- C4R.CONNECT./AUTH.*group.userid*

This profile is used to specify a default value for the connect AUTH in case the terminal user did not specify a connect authorization level on the **ADDUSER** or **CONNECT** command. If the Mandatory Value policy profile is used to provide a value, the /AUTH profiles are not used. Also, when you define a new user profile, RACF inserts the value USE for the authority in the DFLTGRP. As a result, zSecure Command Verifier does not detect the absence of any value in the command as specified by the terminal user. Instead, zSecure Command Verifier processes the command as if the terminal user entered the value USE.

The AUTH value that is used as default, is obtained from the **APPLDATA** field in the profile. The CONNECT value that is obtained from this Mandatory Value profile is not subject to more CONNECT-related policy profiles.

The qualifierAUTH in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied. However, RACF also cannot provide a value for the CONNECT authorization. The command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the CONNECT AUTH.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied. RACF uses its default authorization (USE).

The values that are accepted for the **APPLDATA** field. The terminal user still needs sufficient authority to assign the CONNECTauthorization. Insufficient authority can result in failure of the command.

auth
Any of the possible connect authorization levels USE, CREATE, CONNECT, JOIN. The value is inserted as the CONNECT authorization for this USER.

other
This is considered an error. The RACF default CONNECT authorization (USE) is used instead.

- C4R.CONNECT.=UACC.*group.userid*

This profile is used to specify a mandatory value for the UACC of new CONNECTs. The Connect-UACC specifies, for data sets and some other resource classes, the default UACC for new resource profiles. The default value is used by RACF if the terminal user did not specify a value for the UACC of the new resource profile. Because the Connect-UACC setting can lead to confusing behavior of RACF, the preferred setting is NONE.

zSecure Command Verifier uses the =UACC policy profile to control the Connect-UACC setting. The policy profile is only used for the **ADDUSER** and **CONNECT** command, and only for new CONNECTs. The UACC value that is enforced, is obtained from the **APPLDATA** field in the policy profile. It is used to override any terminal user specified value, or added to the command if the terminal user did not specify a value. The UACC value that is obtained from this Mandatory Value profile is not subject to more UACC-related policy profiles.

The values *userid* and *group* represent the affected user and group. This setting allows the specification of exceptions to the general rule. Only the most specific profile is used by zSecure Command Verifier. Generic profiles can be used to specify the UACC of users within the group.

The qualifier =UACC in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No mandatory value is enforced.

NONE
No action. No mandatory value is enforced.

READ
The **APPLDATA** field is extracted and used for the command. If this process does not yield a valid UACC level, NONE is used instead.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No mandatory value is supplied. If the terminal user specified a CONNECT UACC, it is used. If no value was specified, RACF uses the value NONE.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value policy profiles, this case leads to the odd situation that access NONE has the same net result as access CONTROL.

The following values are accepted for the **APPLDATA** field.

uacc
Any of the possible UACC levels NONE, READ, UPDATE, CONTROL, ALTER. The value is inserted as the UACC for this CONNECT.

other
This value is considered an error. The RACF default UACC (NONE) is used instead.

- C4R.CONNECT./UACC.*group.userid*

This profile is used to specify a default value for the UACC in case the terminal user did not specify a UACC value on the **ADDUSER** or **CONNECT** command. If the preceding Mandatory Value policy profile is used to provide a value, the /UACC profile is not used. Also, when you define a new user profile, RACF inserts the value NONE as the UACC for the DFLTGRP. As a result, zSecure Command Verifier does not detect the absence of any value in the command as specified by the terminal user. Instead, zSecure Command Verifier processes the command as if the terminal user entered the value NONE.

The UACC value that is used by default is obtained from the **APPLDATA** field in the profile. The UACC value that is obtained from this Mandatory Value profile is not subject to more UACC-related policy profiles.

The qualifier /UACC in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied. However, RACF also cannot provide a value for the UACC level. The command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the UACC.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied. RACF uses its default authorization (NONE).

The following values are accepted for the **APPLDATA** field.

uacc
Any of the possible connect authorization levels NONE, READ, UPDATE, CONTROL, ALTER. The value is inserted as the UACC value for this CONNECT.

other
This value is considered an error. The RACF default UACC level (NONE) is used instead.

Verification of the CONNECT values specified by the terminal user

Use these profiles to verify the CONNECT authorization and UACC values as specified by the terminal user.

- C4R.CONNECT.OWNER.*owner.group.userid*

  This profile is used to verify the OWNER value that is specified by the terminal user. The policy can be implemented only for the **CONNECT** command. To define a general policy, use profiles with generic patterns for both the *owner* and the *userid* qualifiers. Use more specific (or discrete) profiles to define exceptions for certain user IDs. The value *owner* can be any RACF defined user ID or GROUP.

  This profile is not used if a Mandatory or Default Value policy profile was used to assign a **CONNECT** owner.

  No profile found
  The control is not implemented. Any **CONNECT** owner that is allowed by RACF can be assigned to this connection.

  NONE
  The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted for this GROUP and user ID.

  CONTROL
  Same as UPDATE.

- C4R.CONNECT.AUTH.*auth.group.userid*

  This profile is used to verify the AUTH value that is specified by the terminal user. The policy can be implemented for the **ADDUSER**, **ALTUSER**, and **CONNECT** commands. For most situations, you can use generic profiles for both the *owner* and the *userid*. Explicit profiles can be used to define exceptions for certain user IDs. The value *auth* can be any RACF accepted **CONNECT** Authorization that is, USE, CREATE, CONNECT, and JOIN.

  This profile is not used if a Mandatory or Default Value policy profile was used to assign a CONNECT authority. Also, when you define a new user profile or creating a CONNECT, the value USE is accepted without verification of these policy profiles.

  No profile found
  The control is not implemented. Any CONNECT authority that is allowed by RACF can be assigned to this connection.

  NONE
  The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The specified *auth* is accepted for this GROUP and user ID.

  CONTROL
  Same as UPDATE.

- C4R.CONNECT.UACC.*uacc.group.userid*

  This profile is used to verify the UACC value that is specified by the terminal user. The policy can be implemented for the **ADDUSER**, **ALTUSER**, and **CONNECT** commands. For most situations, you can use generic profiles for both the *owner* and the *userid*. Explicit profiles can be used to define exceptions for certain user IDs.

  This profile is not used if a Mandatory or Default Value policy profile was used to assign a UACC. Also, when you define a new user profile or creating a CONNECT, the value NONE is accepted without verification of these policy profiles.

No profile found
The control is not implemented. Any UACC value can be assigned to this connection.

NONE
The command is rejected.

READ
Same as NONE.

UPDATE
The specified *uacc* is accepted for this GROUP and user ID.

CONTROL
Same as UPDATE.

CONNECT attributes and access level descriptions

The following paragraphs describe the access levels that are used to control which attributes can be assigned to user and group CONNECTS.

In general, the access level that is required is UPDATE to give the attribute or READ to take away the attribute.

- C4R.CONNECT.ATTR.SPECIAL.*group.userid*
- C4R.CONNECT.ATTR.OPERATIONS.*group.userid*
- C4R.CONNECT.ATTR.AUDITOR.*group.userid*
- C4R.CONNECT.ATTR.ADSP.*group.userid*
- C4R.CONNECT.ATTR.GRPACC.*group.userid*

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to specify either keyword on the **CONNECT** command.

READ
The terminal user is authorized to explicitly specify the no-attribute keyword on the **CONNECT** command. This setting allows removal of these attributes.

UPDATE
The terminal user is authorized to specify both keywords on the **CONNECT** command. This setting allows regular maintenance of these attributes.

CONTROL
Same as UPDATE.

In all the preceding situations, the terminal user needs sufficient RACF authorization to specify the keyword. For instance, for most keywords, the terminal user must have the group-SPECIAL attribute in the group.

- C4R.CONNECT.ATTR.REVOKE.*group.userid*

This policy profile applies only to the REVOKE attribute without a future REVOKE date. Management of revoke dates is controlled by the REVOKEDT policy profile that is described as follows.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to revoke the user **CONNECT**. This setting applies to the REVOKE keyword without any specification of a future REVOKE date.

READ
The terminal user is authorized to REVOKE a user **CONNECT**. This setting applies to the REVOKE keyword without specification of a future REVOKE date.

UPDATE
Same as READ.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to revoke a `userid`.

- `C4R.CONNECT.ATTR.RESUME.`*group.userid*

This policy profile applies only to the RESUME attribute without a futureRESUME date. Management of resume dates is controlled by the RESUMEDT policy profile that is described as follows.

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to resume the user CONNECT. This setting applies to the RESUME keyword without specification of a future RESUME date.

READ
Same as NONE.

UPDATE
The terminal user is authorized to RESUME a user CONNECT. This setting applies only to an immediate RESUME without a future RESUME date.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to resume the user CONNECT.

- `C4R.CONNECT.ATTR.REVOKEDT.`*group.userid*

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to manage the revoke dates for CONNECTS between the *user* and the *group*. This setting applies to both REVOKE(*date*) and the NOREVOKE option.

READ
Same as NONE

UPDATE
The terminal user is allowed to manage the revoke dates by REVOKE(*date*) or NOREVOKE.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to manage the revoke dates for the CONNECT from the *userid* to *group*.

- `C4R.CONNECT.ATTR.RESUMEDT.`*group.userid*

No profile found
This control is not implemented. No action is performed.

NONE
The terminal user is not authorized to manage the resume dates for CONNECTS between the *user* and the *group*. This setting applies to both RESUME(*date*) and the NORESUME option.

READ
Same as NONE

UPDATE
The terminal user is allowed to manage the resume dates by RESUME(*date*) or NORESUME.

CONTROL
The control is not implemented for the terminal user. The terminal user is authorized to manage the resume dates for the CONNECT from the *userid* to *group*.

Policy profiles for managing DATASET and general resource profiles

Use the topics in this section to implement policy profiles to manage DATASET and general resource profiles.

General resource profiles are handled by RACF through two distinct sets of commands. In zSecure Command Verifier, the term *resource profiles* is used to refer to both types. Also, for all the zSecure Command Verifier policy profiles, no real distinction is made between the different types of resource profiles. However, some policies do not make any sense for a particular resource class. For instance, the timezone setting does not apply to data sets. The description of the policies ignores such specifics and concentrates on the general rule.

The following sections contain the various zSecure Command Verifier policy profiles that can be used for resource profiles. Policies regarding the authorization to create new resource profiles are described in "User authorization to create resource profiles" on page 157. These policies also describe the authorization to add or delete members to or from Grouping Resource Class Profiles. An `ADDMEM` for a `GCICSTRN` is treated as identical to an `RDEFINE` for the corresponding `TCICSTRN`. One special policy is implemented for data sets only. It describes the authorization to maintain data set profiles that have as HLQ the terminal user. See a full description in "Policy profiles for managing your own data set profiles" on page 150.

In sections where Policy profiles for `DATASET` profiles are used, the target profile is often represented by HLQ and rest-of-profile:

- The HLQ represents the High-Level Qualifier of the actual `DATASET` profile that is used in the RACF commands. It is the first qualifier that is specified in a quoted data set name in the RACF command. The HLQ used in the policy profiles does not reflect possible changes from the naming convention table. This HLQ must be an existing `userid` or group.
- The value for rest-of-profile reflects all the qualifiers after the HLQ. For most TSO users, this part is the part of the `DATASET` profile name that can be used as the non-quoted data set name.

Splitting the profile into two parts highlights the fact that most installations use generics to represent the rest-of-profile part. In some policy profiles, the HLQ might be reflected by the special qualifier `=RACUID`. The term HLQ can also be present in the **APPLDATA** specifications of certain policy profiles. In those situations, it also represents the first qualifier of the actual `DATASET` profile as defined in the RACF database.

The next important issue for resource profiles is the access. Access through UACC and the Access Lists is described in "Controlling the UACC and access list" on page 168.

Other items, like the owner of resource profiles, further identification of the resource profile (Volume, Unit, Profile type, RACF indicator) attributes, and auditing are shown in the following sections.

Generic and special characters in policy profiles

Use these guidelines to define the resource profile part in policy profiles.

Most of the policy profiles contain as part of their name the resource class and the resource profile to which the policy applies. It is possible to use generic patterns for the resource profile part. However, sometimes it is necessary to define a policy that applies to a generic resource profile such as the profile `**` in the `FACILITY` class. Defining a specific policy profile for this resource profile is rather difficult. To be able to protect generic profiles efficiently, zSecure Command Verifier modifies the resource profile part in the policy profiles. All generic characters are replaced by plus (+) signs. This way, an installation can define a specific policy profile to implement a policy for a specific generic resource profile. Using this translation, the authorization to define above `FACILITY` profile is described by:

```
C4R.FACILITY.ID.++
```

This profile describes the authorization to create the profile ** (and also profiles %% and %*) in the FACILITY class. The policy profile can also be specified as:

```
C4R.FACILITY.ID.**
```

However, this second profile would control the authorization to create any profile in the facility class. A generic profile that ends in %% C4R.FACILITY.ID.%% controls all two character profiles in the FACILITY class.

The translation process also affects some special characters. The single quotation mark is also translated into a plus sign, and the forward slash is translated into a period. The translation of special characters is done to allow effective handling of members in the GLOBAL resource class.

RACF variables in general resource profiles

Use this information if you need to support RACF variables in the general resource profiles that are controlled through zSecure Command Verifier policy profiles.

To implement some policies, zSecure Command Verifier must locate existing resource profiles in the RACF database. For example, for the automatic modeling on the current best fitting profile (see "Profile modeling based on the best-fitting generic" on page 147), zSecure Command Verifier uses several low level RACF interfaces to determine this best-fitting profile that is to be used. For example, suppose that profile PAYROLL.EMPLOYEE.** exists and that the administrator issues the following ADDSD command:

```
ADDSD 'PAYROLL.EMPLOYEE.Q4.**'
```

Then zSecure Command Verifier can transform that command into the following string:

```
ADDSD 'PAYROLL.EMPLOYEE.Q4.**' FROM('PAYROLL.EMPLOYEE.**')
```

This process also works for general resource profiles. However, for general resource profiles, there is an extra complication if RACF variables are used. For example, assume that only profile &RACLNDE.*.** exists in the JESSPOOL class. The administrator then issues the following command, expecting the new profile to be modeled on the &RACLNDE.*.** profile:

```
RDEF JESSPOOL &RACLNDE.IBMUSER.**
```

zSecure Command Verifier does locate the best fitting profile containing the RACF variable and supplies that profile in the FROM keyword. However, zSecure Command Verifier support for resolution of profiles containing a RACF variable has some limitations:

- The RACF variable must be defined with at least one member (value of the variable).
- The RACFVARS resource class must be RACLISTed.
- At least one of the first 64 members defined must result in using the existing resource profile containing the RACF variable.

For example, automatic modeling for the previous RDEF command is not available if the only member that is defined for &RACLNDE is IPO1, and the following two profiles are defined for the JESSPOOL class:

- IPO1.*.**
- &RACLNDE.*.**

In the previous description, automatic FROM profile selection is used as example. Similar considerations and limitations exist for zSecure Command Verifier undercut policies. For example, when adding the JESSPOOL profile &RACLNDE.IBMUSER.**, the policy profile C4R.JESSPOOL.=UNDERCUT.+RACLNDE.+.+ + might be used to check undercut authorization. See "User authorization to create more specific profiles" on page 153 for a detailed description of this function.

Profiles with lowercase names

zSecure Command Verifier policies apply to all target classes and profiles for which a matching policy profile is defined.

Often, the section of the policy profile that contains the resource class and the resource profile is represented by generic patterns. Exceptions can be implemented by specifying part of the resource profile by discrete characters. However, you cannot define an exception exclusively for a specific mixed case profile in a class that allows such mixed case profiles. Instead, the resulting uppercase policy profile applies to uppercase, lowercase, and mixed case resource profiles.

The zSecure Command Verifier policy profiles are defined in a class that does not support lowercase characters. If you try to define a policy profile for a lowercase resource profile, the RACF command processors immediately translate the policy profile to uppercase. zSecure Command Verifier follows this behavior, translating the resource profile to uppercase before it locates the matching policy profile. The following example describes this implementation:

```
Resource profile     EJBROLE         Test.Role-       EJBROLE         TEST.ROLE
                     EJBROLE         test.role

Policy profile       C4R.EJBROLE.ID.TEST.ROLE

Command              rdefine xfacilit c4r.ejbrole.id.test.role
```

In this example, the three different EJBROLE profiles are all controlled by the same zSecure Command Verifier policy profile. The case of the command that is used to create the policy profile is irrelevant for the resulting policy profile.

General policy profiles to add functionality

zSecure Command Verifier currently provides two general policies that you can use to add functionality.

The first policy automatically inserts the GENERIC keyword in an ambiguous **LISTDSD** command if no matching discrete profile exists. That way, the terminal user does not need to know whether a discrete profile or a generic profile exists when you use the **LISTDSD** command. You can use the second policy when you create new data set or general resource profiles. This policy automatically inserts the FROM keyword to model the new profile on the currently best-fitting profile. It ensures that the OWNER, auditing options, the UACC, and the ACL are copied from an existing profile. The following table lists the commands, keywords, and profile names available to add functionality.

Table 32. Profiles used for added functionality

Command	Keyword	Profile
LISTDSD	*hlq.rest-of-profile*	C4R.LISTDSD.TYPE.AUTO.*hlq.rest-of-profile*
ADDSD RDEFINE	*hlq.rest-of-profile*	C4R.*class*.=FROM.*hlq.rest-of-profile*
ADDSD RDEFINE	*hlq.rest-of-profile*	C4R.*class*./FROM.*hlq.rest-of-profile*
ADDSD RDEFINE	*hlq.rest-of-profile*	C4R.*class*.FROM.*hlq.rest-of-profile*

Automatic search for the best-fitting generic profile

Use the zSecure Command Verifier automatic search to find the best fitting generic profile for data set profiles.

For historical reasons, data set profiles are treated separately from other resources in RACF. For example, if you request a display of a data set profile, RACF assumes that the profile is a discrete profile (unless it has generic characters). If the discrete profile does not exist, RACF outputs the following message:

```
ICH35003I NO RACF DESCRIPTION FOUND FOR dataset_name
```

Quite often, the next command that is issued is the same **LISTDSD** command, but now including the GEN keyword. It can be used to display the best fitting generic profile.

Similarly, for general resource profiles, the **RLIST** command displays a discrete profile if it exists. But, in contrast to the **LISTDSD** command, **RLIST** automatically displays the best fitting generic profile if a discrete profile cannot be found. As a RACF usability feature, zSecure Command Verifier also provides this automatic search for the best fitting generic profile for data set profiles.

- C4R.LISTDSD.TYPE.AUTO.*hlq.rest-of-profile*

  If this profile exists, zSecure Command Verifier tests for the existence of a requested profile. If the requested profile does not exist, zSecure Command Verifier inserts the GEN keyword in the **LISTDSD** command, resulting in a search for the best fitting profile. The access level controls whether the function is active for the terminal user. Some considerations are listed for certain combinations of commands and non-existing profiles. In most installations, the *profile* is represented by a generic pattern like .** The following access levels are supported.

  No Profile Found
  This function is not implemented.

  NONE
  The function is not activated for the terminal user.

  READ
  If a discrete profile cannot be located, the best fitting generic profile is shown instead.

  UPDATE
  Same as READ

  CONTROL
  Same as READ

Notes:

1. If the specified profile contains generic characters, this particular zSecure Command Verifier processing is bypassed. It is assumed that the user wants the specified profile to be displayed, instead of the next best generic profile. If the profile does not exist, RACF provides the appropriate error message.
2. If the terminal user requested a discrete profile for a specific volume, and no discrete profile exists, the best fitting generic is shown.
3. If the terminal user requested a discrete profile for a specific volume, and a discrete profile is defined for a different volume, RACF provides an error message, informing the terminal user that a discrete profile for the volume cannot be located.
4. In situations where the automatic search for the best fitting generic is undesirable, you can disable the automatic search function on a per command basis by including the NOGENERIC keyword on the **LISTDSD** command.
5. The presence of a discrete profile does not imply that it is used to protect a particular data set. Protection is also dependent on the correct volser, and the setting of the RACF indicated bit in the VTOC (or ICF catalog).

Profile modeling based on the best-fitting generic

Use the MODEL policy profiles with corresponding user and group profiles to help manage access to resources such as data sets.

Frequently, a RACF administrator is asked to define a new profile to control access to a specific resource. In most situations, access to the resource is already controlled by some profile. For example, in a PROTECTALL environment, all data set profiles must be controlled by a profile. When RACF defines a new data set profile or general resource profile, it creates a profile with a UACC(NONE), and an empty access list by default. Exceptions to this general rule can occur when the NOADDCREATOR option is not set, when the terminal user has a UACC setting other than NONE, or when the terminal user has the GRPACC attribute. For data sets, an installation can have well-defined MODEL profiles with corresponding User and Group profiles such that the MODEL profile is used. However, use of MODEL profiles requires maintenance of the model profile to adequately describe the current access. It also does not apply to general resource profiles.

To allow greater flexibility in the creation of new profiles, RACF provides the FROM keyword on **ADDSD**, **RDEFINE**, and **PERMIT** commands. However, effective use of this function still requires some effort by the RACF administrator. In many situations, a sequence of several commands is still required to create the profile. When the administrator defines a new user profile or a more specific profile, the administrator does not want to lock out current users because locking out users can seriously impact the production environment. In the example that follows, the first command is used to find the profile that currently controls access to the data set. In the example, the data set was controlled by the profile PAYROLL.EMPLOYEE.**. In the second command, that profile is used as a model for the new profile. Finally, in the third command, the user ID or GROUP that needed access to the Q4 files is granted access.

```
LISTDSD DA('PAYROLL.EMPLOYEE.Q4.Y2003') GEN
ADDSD 'PAYROLL.EMPLOYEE.Q4.**' FROM('PAYROLL.EMPLOYEE.**')
PERMIT 'PAYROLL.EMPLOYEE.Q4.**' ID(PAYTMP) AC(UPDATE)
```

zSecure Command Verifier provides a function to automatically insert the FROM keyword that is based on the current resource profile. For example, when this function is enabled for the entire DATASET class, the administrator can issue the following two commands without having to check the current access list or UACC for the resource.

```
ADDSD 'PAYROLL.EMPLOYEE.Q4.**'
PERMIT 'PAYROLL.EMPLOYEE.Q4.**' ID(PAYTMP) AC(UPDATE)
```

You can activate the automatic model function per resource class or per resource profile. Most installation sites use only the *class* or the HLQ of the profile, and probably use generics for the remaining resource profile section of the policy profile.

- C4R.*class*.=FROM.*hlq.rest-of-profile*

  If this policy profile exists, and the terminal user has appropriate access, zSecure Command Verifier retrieves the APPLDATA of the profile to locate the resource profile to be used in the **RDEFINE** or **ADDSD** command. The *hlq.rest-of-profile* describes the new profile to be defined. Because this profile is a Mandatory Value policy profile, it overwrites any value that the terminal user specified for the FROM keyword. The value found from the **APPLDATA** field is used instead.

  The qualifier =FROM in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The following access levels are supported for the policy profile.

  No Profile Found
  This function is not implemented.

  NONE
  The function is not activated for the terminal user. No Mandatory FROM model profile is inserted in the RACF command as entered by the terminal user.

  READ
  The APPLDATA of the policy profile is retrieved and used to determine the profile to be used as the model.

  UPDATE
  Same as READ

  CONTROL
  The control is not active for the terminal user. No FROM model profile is inserted in the RACF command as entered by the terminal user.

  If the terminal user has READ or UPDATE access to the Mandatory Value policy profile, zSecure Command Verifier retrieves and uses the APPLDATA of the policy profile. The APPLDATA values currently supported are shown in a subsequent section.

- C4R.*class*./FROM.*hlq.rest-of-profile*

  If this policy profile exists, and the terminal user has appropriate access, zSecure Command Verifier retrieves the APPLDATA of the profile to locate the resource profile to be used in the **RDEFINE** or **ADDSD** command. The *hlq.rest-of-profile* describes the new profile that is to be defined.

The qualifier /FROM in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The following access levels are supported.

No Profile Found
This function is not implemented.

NONE
The function is not activated for the terminal user. No Default FROM model profile is inserted in the command.

READ
The APPLDATA of the policy profile is retrieved, and used to determine the profile to be used as model.

UPDATE
Same as READ

CONTROL
The control is not active for the terminal user. No FROM model profile is inserted in the RACF command as entered by the terminal user.

If the terminal user has READ or UPDATE access to the Default Value policy profile, zSecure Command Verifier retrieves and uses the APPLDATA of the policy profile. The APPLDATA values currently supported are shown in a subsequent section.

- C4R.*class*.FROM.*hlq.rest-of-profile*

  This policy profile controls if the terminal user is authorized to use the FROM keyword when it adds new data sets or general resource profiles. This profile is not used if one of the preceding Mandatory or Default Value policy profiles is used. The *hlq.rest-of-profile* describes the new profile that is to be defined. This policy profile does not contain the name of the model profile that is used in the command. The following access levels are supported.

  No Profile Found
  This function is not implemented.

  NONE
  The FROM keyword is not allowed.

  READ
  Same as NONE

  UPDATE
  The specified FROM keyword is allowed.

  CONTROL
  The control is not active for the terminal user.

If the terminal user has READ or UPDATE access to the Mandatory or Default Value policy profile, zSecure Command Verifier retrieves the APPLDATA of the policy profile. The **APPLDATA** field of the C4R.*class*.FROM.*hlq.rest-of-profile* is not used.

The **APPLDATA** field can have the following value types:

BLANK
This type is used to indicate that no explicit FROM profile must be inserted. For the Mandatory Value policy profile, it means that a possibly specified FROM value in the command as entered by the terminal user is removed. Subsequent RACF default processing can result in using a USER or GROUP-specific MODEL profile (if defined and modeling is enabled).

=BESTFIT
This value specifies that zSecure Command Verifier is to locate the current best fitting profile and use the profile that is found as the value for the FROM profile. The profile is in the same resource class as the new resource profile. If no profile can be found, processing is as if APPLDATA has the value BLANK.

profile
Any other value is considered to be the resource profile that must be used as model. If this resource profile does not exist, the entire command eventually fails resulting in the RACF message ICH09036I.

RACF profile management

zSecure Command Verifier provides several policies for RACF profile management.

These policies are listed in Table 33 on page 150. They can be used for controlling the following authorities:

- Authority to manage your own data sets
- Authority to authorize yourself (by USERID, or GROUP)
- Authority to create more specific profile (undercut)
- Authority to manage system resources (identified by LEVEL)
- Authority to grant UPDATE access to resources (identified by LEVEL)

For detailed profile descriptions, see the sections that follow the table.

Table 33. General Profiles used for profile management

Command	Keyword	Profile
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.ID.=RACUID.*rest-of-profile*
PERMIT	*userid*	C4R.*class.*ACL.=RACUID.*access.profile*
PERMIT	*group*	C4R.*class.*ACL.=RACGPID.*access.profile*
CONNECT	*userid*	C4R.CONNECT.ID.*group.*=RACUID
REMOVE	*userid*	C4R.REMOVE.ID.*group.*=RACUID
ADDSD RDEFINE	*profile*	C4R.*class*.=UNDERCUT.*current-profile*
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.=NOCHANGE.*dsname*
RDEF RDEL RALT PERMIT	*profile*	C4R.*class.*=NOCHANGE.*profile*
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.=NOUPDATE.*dsname*
RDEF RDEL RALT PERMIT	*profile*	C4R.*class.*=NOUPDATE.*profile*

Policy profiles for managing your own data set profiles

The function to control authority to manage your own data set profiles is also known as the *No-Store* function. The name is derived from a control available in ACF2 systems.

In standard RACF, every user can add, delete, and modify data set profiles for which the HLQ is the same as the user ID. RACF does not provide an easy method to change this behavior. The main method is to create a naming convention table, such that the HLQ is no longer the same as the user ID. It has the obvious disadvantages that are associated with any usage of the naming convention table. An alternative would be to write several installation exits. zSecure Command Verifier externalizes this functionality for the RACF commands. The following table describes the command, keyword, and profile to manage your own data set profiles. The section that follows the table provides a detailed description of the profile.

Table 34. Profiles used for verification of RACF Resources. The entries in this table reflect the keywords that describe the name of new resources.		
Command	**Keyword**	**Profile**
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.ID.=RACUID.*rest-of-profile*

A possible unexpected result from the use of the No-Store function is that decentralized system administrators can be authorized to create and maintain all data set profiles for all users in their department, except their own.

- C4R.DATASET.ID.=RACUID.*rest-of-profile*

  If the HLQ of the data set profile matches the user ID of the terminal user, this policy is verified before any other policy verification as described in "Policy profiles for creating RACF resource profiles" on page 160. If the user has insufficient access, the command is rejected and processing stops. If this policy does not prevent management of the target profile, other applicable policies (for example, for the ACL) are evaluated during subsequent processing. The value for *rest-of-profile* reflects all the qualifiers after the HLQ. For most TSO users, the *rest-of-profile* is the part of the data set name that can be used as the non-quoted data set name. For most situations, you probably want to use generic characters, like **, to represent the *rest-of-profile*.

  The qualifier =RACUID in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  The following list contains the available access levels.

  No Profile Found
  This control is not implemented. Command processing continues.

  NONE
  The terminal user is not authorized to manage their own DATASET profiles. The command is rejected.

  READ
  Same as NONE

  UPDATE
  The terminal user is not prevented from managing its own dataset profiles. Command processing continues with verifying other policies.

  CONTROL
  Same as UPDATE

Policy profile selection for self-authorization

The self-authorization function is required by many organizations who want to enforce a strict separation of responsibilities between the security administrator and the data (or application) administrator.

System security policies often specify that security administrators must not have access to application resources. In standard RACF, users with System or Group-SPECIAL can change any profiles under their control. Even if a security administrator currently does not have access to application resources, it is easy for an administrator to obtain access. Some organizations analyze SMF data to report on those administrators that give themselves access to application data or resources. In zSecure Command Verifier, several policies are available to prevent security administrators from modifying the ACL of resource profiles such that they can gain access to resources.

The following table describes the command, keyword, and profile to control self-authorization. Detailed descriptions of the profiles are provided in the sections that follow the table.

Table 35. Profiles used to control self-authorization. The entries in this table reflect the keywords that describe the ACL entries or CONNECTs.

Command	Keyword	Profile
PERMIT	*userid*	C4R.*class*.ACL.=RACUID.*access.profile*
PERMIT	*group*	C4R.*class*.ACL.=RACGPID.*access.profile*
CONNECT	*userid*	C4R.CONNECT.ID.*group*.=RACUID
REMOVE	*userid*	C4R.REMOVE.ID.*group*.=RACUID

These profiles are only applicable if the *userid* is the terminal user or if the *group* is any of the connect groups of the user. If this situation applies, zSecure Command Verifier uses the preceding profiles. If the policy does not prevent modifying the access list, processing continues with checking other ACL policies as described in "Controlling the UACC and access list" on page 168. A detailed description of the first two profiles and the supported access levels is given in this section. For more information about the policy profiles for CONNECT and REMOVE, see "Authority to connect yourself" on page 130.

- C4R.*class*.ACL.=RACUID.*access.profile*

  This profile is used to specify the authority of the terminal user to issue a PERMIT command that changes the access level of him/herself. It also applies to the **DELETE** option of the **PERMIT** command. If you implement this profile, make sure to set the SETROPTS NOADDCREATOR option. Otherwise, a RACF administrator can automatically be added to the access list of resource profiles, without any possibility for the administrator to remove this questionable access level.

  The qualifier =RACUID in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. The terminal users are not prevented from adding, changing, or removing themselves on an access list.

  NONE
  The terminal users are not allowed to add, change, or remove themselves on an access list.

  READ
  Same as NONE

  UPDATE
  The terminal users are not prevented from adding, changing, or removing themselves on an access list.

  CONTROL
  Same as UPDATE

- C4R.*class*.ACL.=RACGPID.*access.profile*

  This profile is used to specify the authority of the terminal user to issue a **PERMIT** command that changes the access level of any of the GROUPs to which the terminal user is connected. It also applies to the **DELETE** option of the **PERMIT** command. Ensure that if you implement this option, the GRPACC attribute is not specified for the terminal user or for any of the GROUP CONNECTs. Otherwise, the current GROUP of the RACF administrator can automatically be added to the access list of data set profiles, without any possibility for the administrator to remove this questionable access level.

  The qualifier =RACGPID in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. The terminal users are not prevented from adding, changing, or removing any of their connect GROUPs on an access list.

  NONE
  The terminal users are not allowed to add, change, or remove any of their connect GROUPs on an access list.

READ
Same as NONE

UPDATE
The terminal users are not prevented from adding, changing, or removing any of their connect GROUPs on an access list.

CONTROL
Same as UPDATE

User authorization to create more specific profiles

Use these guidelines to create *undercut-profiles* that help control the authorization to create resource profiles.

As described in "User authorization to create resource profiles" on page 157, RACF uses multiple methods to control the creation (definition) of profiles. For data set profiles, only the HLQ is used to determine the authorization. Existing generic profiles are not used in this authorization process. This case generally results in the possibility for group administrators, or users with CREATE authority in a GROUP, to define more specific generic profiles, or even discrete profiles that undermine existing access controls. The more specific profile is used by RACF, and the previously best fitting profile (including its UACC and ACL) is no longer used for some resources.

For general resources, RACF uses CLAUTH in combination with `GENERICOWNER` to control which users can define new profiles.

zSecure Command Verifier provides a generic facility that can be used to prevent creation of more specific profiles. Since the process of creating more specific profiles is sometimes referred to as *undercutting* the existing profile, the profiles are referred to as *undercut-profiles* in the remainder of this document.

Note: zSecure Command Verifier currently does not provide a similar function that prevents the use of ADDMEM to undercut existing members in existing grouping profiles.

The following table shows the controls that are provided by zSecure Command Verifier to prevent creation of more specific profiles. Depending on the status (RACLISTed or not) of the resource class, the use of `RDEFINE` might be restricted.

Table 36. Profiles used for verification of RACF Resources. The entries in this table reflect the keywords that describe the name of new resources.

Command	Keyword	Profile
ADDSD RDEFINE	*profile*	C4R.*class*.=UNDERCUT.*current-profile*

The authority to create a profile is controlled by a policy profile that contains the current best fitting profile. When the current best fitting profile contains generic characters, plus signs (+) are used to represent these generic characters in the zSecure Command Verifier policy profiles. Translation of generic characters in policy profiles is described in "Generic and special characters in policy profiles" on page 144. For instance, when the following data set profiles exist,

```
ABC.**
ABC.TEST*.**
```

the definition of data set profile:

```
ABC.TEST1.PROF*
```

is controlled by the definition:

```
C4R.DATASET.=UNDERCUT.ABC.TEST+.++
```

which can be covered by the following zSecure Command Verifier policy profile:

```
C4R.DATASET.=UNDERCUT.**
```

The following section describes the policy profiles for undercutting RACF resource profiles and the corresponding access levels. These profiles are used in addition to the standard RACF profile create authority to other zSecure Command Verifier policy profiles and to policies defined as described in "User authorization to create resource profiles" on page 157.

- C4R.*class*.=UNDERCUT.*current-profile*

  This profile describes the authorization to create resource profiles that would undercut the *current-profile*. The *current profile* is the profile that is used by RACF to protect the resources that would be covered by the new profile. Phrased differently, the *current-profile* is the existing profile that is being undercut, and not the new profile that is undercutting.

  The qualifier =UNDERCUT in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  The following access levels are available.

  No Profile Found
  This control is not implemented.

  NONE
  The user is not authorized to define the new profile.

  READ
  Same as NONE

  UPDATE
  The terminal user can create the profile, provided the terminal user has otherwise sufficient RACF authorization.

  CONTROL
  Same as UPDATE

User authorization to manage locked resource profiles

Use these guidelines to create profiles that help control changes to specific profiles, such as profiles that protect APF-authorized data sets.

This function is also known as the *No-Change* function. The name indicates that this control can be used to prevent changes to certain profiles. The most common use of this feature is to prevent users from updating profiles that protect system resources such as APF-authorized data sets. Since it is difficult to recognize all possible resources automatically for every RACF command, zSecure Command Verifier implements an indirect approach to the problem. A special =NOCHANGE policy profile is used to define a characteristic of the target profile. If the target profile has this characteristic, then more access to the policy profile is required to modify the target profile as illustrated in the following scenario.

Assuming that you want to implement this additional control for the data set SYS1.LINKLIB, define the following policy profile:

```
C4R.DATASET.=NOCHANGE.SYS1.** APPLDATA('LEVEL=99')
```

This policy profile indicates that all SYS1 data sets that have a LEVEL specification of 99 need this additional control. To activate this control for SYS1.LINKLIB, specify the value 99 for the LEVEL of the data set profile. Assuming that the data set is covered by the profile SYS1.LINK*, use the following command:

```
ALTDSD 'SYS1.LINK*' LEVEL(99)
```

By using SYS1.** in the =NOCHANGE profile, one policy profile is sufficient to indicate that all SYS1 data sets with a particular level are controlled. At the same time, you can implicitly specify that all non-SYS1 data sets are not controlled. If you wanted all data sets to be controlled by this No-Change function, you can also use the following policy profile instead:

```
C4R.DATASET.=NOCHANGE.** APPLDATA('LEVEL=99')
```

For most commands, the characteristic that is used to determine whether a resource profile is controlled is obtained from the profile that is specified in the RACF command. However, you cannot obtain that characteristic from the **ADDSD** and **RDEFINE** commands. For these two commands, the characteristic is obtained from the currently best fitting profile.

In effect, it enforces undercut control as described in "User authorization to create more specific profiles" on page 153. Adding a better fitting profile that would undermine the No-Change policy is not allowed. Similarly, removing a profile that currently enforces the No-Change policy is also not allowed. Although it can b e viewed as a mixing of control policies, it does allow the use of one profile to effectively indicate that a particular block of resources is off-limits. In the example above, the use of the next-best profile prevents the creation of a profile like SYS1.LINKLIB with a LEVEL(00), as shown in the following example:

```
ADDSD 'SYS1.LINKLIB' GENERIC LEVEL(00)
```

Specifying the LEVEL(00) effectively takes the resource profile out of the =NOCHANGE policy. This step must not be allowed. Use of the best fitting profile for the =NOCHANGE policy enforces this rule. Similarly, the command

```
DELDSD 'SYS1.LINK*' GENERIC
```

is not allowed because it would also remove the current No-Change policy from all data sets covered by that profile. Most likely the next-best fitting profile (SYS1.**) does not have the LEVEL(99) specification, and thus the =NOCHANGE policy would be disabled for these data sets. This is not allowed.

The access to the policy profile determines whether the profile modification is allowed or not.

The following table is split into two rows, one for data sets dsname and the other for general resources (profiles). The remainder of this topic does not discuss data sets separately, but treats them as a special case where *class* has the value DATASET.

Table 37. Profiles used for verification of RACF Resources. The entries in this table reflect the keywords that describe the name of new resources.

Command	Keyword	Profile
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.=NOCHANGE.*dsname*
RDEF RDEL RALT PERMIT	*profile*	C4R.*class.*=NOCHANGE.*profile*

The following access levels and values for the **APPLDATA** are currently available.

- C4R.*class.*=NOCHANGE.*profile*

  The **APPLDATA** of the policy profile is used to indicate which characteristic of the target profile to use to identify the profiles that cannot be modified without more authorization. The value of *profile* reflects the data set name or the general resource profile. For most situations, the *profile* is represented by using generic characters, like ".**".

  The qualifier =NOCHANGE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  Only one type of characteristic is implemented. The possible value for **APPLDATA** is given as follows:

  LEVEL= *nn*
  The LEVEL of the profile is used to indicate whether more controls on modification of the profile are required. If the target has *nn* specified for the LEVEL, at least UPDATE access to the policy profile is required to allow modification of the target profile.

  The following access levels are available.

  No Profile Found
  This control is not implemented. Modification of the target profile is not prevented.

NONE
If the target profile fits the requirement that is specified by the **APPLDATA**, the terminal user is not authorized to modify the target profile.

READ
Same as NONE

UPDATE
The terminal user can modify the target profile, provided it is within the regular RACF authorization of the terminal user.

CONTROL
Same as UPDATE

Policy profile selection to control UPDATE access

Use these guidelines to create policy profiles that control the granting of UPDATE access to specific profiles.

This function is also known as the *No-Update* function. The name indicates that this control can be used to prevent granting UPDATE access to certain profiles. The resources that are covered by this policy can be identified by a combination of the LEVEL of the resource profile and the name of the resource. The main difference to the regular ACCESS control mechanisms, is that the resources can be selected by the LEVEL of the profile. Using this selection method, a single policy profile can be used to apply this rule to as a separate set of resources.

The process is probably best illustrated by using an example. Assuming that you want to implement this additional control for the following data sets:

```
ACCPAY.JCLLIB
ACCPAY.PARMLIB
```

You can define the following policy profiles:

```
C4R.DATASET.=NOUPDATE.ACCPAY.**         APPLDATA('LEVEL=98')
```

This policy profile indicates that all ACCPAY data sets that have a LEVEL specification of '98' need this additional control. To effectuate this control for the two ACCPAY data sets, you must specify the value '98' for the LEVEL of the data set profiles. Assuming that the data sets are covered by fully qualified generics, you can use the following two commands. All other data sets with ACCPAY can be defined by discrete or generic profiles. Also, the two data set profiles cannot be easily covered by one generic profile.

```
ALTDSD 'ACCPAY.JCLLIB'  GEN LEVEL(98)
ALTDSD 'ACCPAY.PARMLIB' GEN LEVEL(98)
```

By using ACCPAY.** in the =NOUPDATE profile, you can use one policy profile to apply to multiple resources. If more resources also must be protected against UPDATE access, you add a matching generic (or discrete) profile that specifies the correct LEVEL value. There is no need to modify the existing policy profiles. The policy profiles must be extended only if other High-Level Qualifiers (HLQ) are involved. To reduce complexity and avoid possible confusion, ensure that all =NOUPDATE policy profiles specify the same value for the applicable LEVEL by the **APPLDATA**.

For most commands, the LEVEL is obtained from the profile that is specified in the RACF command. However, it is not possible for the **ADDSD** and **RDEFINE** commands. For these two commands, the LEVEL is obtained from the currently best fitting profile. In effect, it enforces undercut control as described in “Policy profile selection to control UPDATE access” on page 156. Although this task can be viewed as a mixing of control policies, it does allow the use of one profile to effectively indicate that a particular block of resources is off-limits. In the previous example, it prevents the creation of a discrete profile for such as ACCPAY.JCLLIB with a LEVEL(00) by

```
ADDSD 'ACCPAY.JCLLIB' LEVEL(00)
```

Specifying the LEVEL(00) effectively take the resource profile out of the =NOUPDATE policy. This case is explicitly prevented by zSecure Command Verifier

To achieve complete, consistent control of all selected data sets against UPDATE access, you also must control management of the LEVEL value of all involved data sets. See the description of the C4R.*class.*LEVEL.*level.profile* in "Other policy profiles and access level descriptions" on page 187 for details on the applicable policy profiles.

The access to the policy profile determines whether granting UPDATE access is allowed.

Table 38. Profiles used for NOUPDATE control. The entries in this table reflect the keywords that describe the affected profiles.

Command	Keyword	Profile
ADDSD DELDSD ALTDSD PERMIT	*profile*	C4R.DATASET.=NOUPDATE.*dsname*
RDEF RDEL RALT PERMIT	*profile*	C4R.*class.*=NOUPDATE.*profile*

This table is split into two rows for data sets and general resources. Data sets are not provided separately, but treated as a special case where *class* has the value DATASET. The following access levels and values for the **APPLDATA** are currently available.

- C4R.*class.*=NOUPDATE.*profile*

  The **APPLDATA** of the policy profile is used to specify the LEVEL of the target profile that must be used as identification of the resource profiles that must be protected against UPDATE access. The value of *profile* reflects the data set name or the general resource profile. For most situations, use generic characters, like ".**", to represent the profile.

  The qualifier =NOUPDATE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  At the moment, only one type of characteristic is implemented. The possible value for **APPLDATA** is given as follows:

  LEVEL= *nn*
  The LEVEL of the target resource profile is used to indicate whether more controls on modification of the target resource profile are required. If the target has *nn* specified for the LEVEL, at least UPDATE access to the policy profile is required to allow granting UPDATE access to the target profile.

  The following access levels are available.

  No Profile Found
  This control is not implemented. Granting UPDATE access to the target profile is not prevented.

  NONE
  If the target profile matches the LEVEL as specified by the **APPLDATA** , the terminal user is not authorized to grant UPDATE access to the target profile.

  READ
  Same as NONE

  UPDATE
  The terminal user can grant UPDATE access to the target profile, provided it is within the regular RACF authorization of the terminal user.

  CONTROL
  Same as UPDATE

User authorization to create resource profiles

RACF recognizes several methods for authorizing users to create new resource profiles.

For data sets, RACF uses the HLQ as the main criterion. Creation of a data set profile is allowed if any of the following conditions are true:

- The HLQ is the same as the user ID.

- The HLQ is a GROUP in which the terminal user has CREATE authority.
- The HLQ is a user ID or GROUP that is within the scope of Group-SPECIAL.
- The HLQ is a GROUP that is within the scope of Group-OPERATIONS.

For general resources, RACF uses CLAUTH in the resource class as the main criterion. For regular general resource classes, the authority to create new resource profiles can be further restricted by the SETROPTS setting for GENERICOWNER. In its simplest form, it can be described as a method to prevent undercutting existing generic profiles that are not yours. However, it does not work for grouping resource classes and adding new generic members.

Note: In some releases of RACF, users cannot define discrete general resource profiles even if the GENERICOWNER is not active because of an existing top generic profile. If the GENERICOWNER is not active, you can bypass this restriction by performing the following steps:

1. Add a temporary, more specific generic profile.
2. Add the discrete profile.
3. Delete the temporary intermediate generic profile.

zSecure Command Verifier provides a generic facility to prevent creation of more specific profiles, both for general resources and data sets. This facility is described in "User authorization to create more specific profiles" on page 153.

Enforcement of resource naming conventions

Use these guidelines to create policies to set additional controls within HLQ restrictions for the naming of data sets and general resources.

The need for naming conventions for user IDs and groups also applies to data sets and general resources. However, for data sets, the need is greatly reduced because RACF already implements severe restrictions on the names of data set profiles. RACF requires that the HLQ of any data set profile corresponds to an existing userid or GROUP that falls in the scope of the terminal user.

If you want to control creation of data set profiles with an HLQ of ABCX, only users ABCX1, ABCX2, and XYZA1 must be able to create such data set profiles. Using zSecure Command Verifier, you can implement this restriction by using the following policy profile definition:

```
C4R.DATASET.ID.ABCX.**  UACC(NONE)  UPDATE(ABCX1,ABCX2,XYZA1)
```

However, RACF already enforces that the HLQ is an existing RACF userid or GROUP. So, of all possible HLQs, most are already controlled because they do not fulfill this basic requirement. RACF requires that the terminal user has some form of authorization to create data set profiles. If the HLQ is a GROUP (as in this example), the user must be connected to the group with at least CREATE authority, or must have Group-SPECIAL authority over the group. It means that for all GROUPs and user IDs defined in your installation that can potentially occur as a data set HLQ, only a few are authorized for a specific terminal user. For example, although your RACF database has a GROUP SYS1, only a few users are authorized to create data set profile with SYS1 as HLQ. So, the preceding profile is only useful if the three users have either Group-SPECIAL authorization, or are connected with CREATE authority. Without either, the users are not authorized to create data set profiles, independent of the existence of the zSecure Command Verifier policy profile.

In current RACF implementations, the use of CREATE authorization is discouraged. It is because of its double function as both a method to control creation of data set profiles, which control security and the creation or allocation of new data set on disk and tape. In most modern RACF implementations, the authority to create application data set profiles is managed by Group-SPECIAL, while creation of data sets on disk and tape is controlled by ALTER access on the appropriate data set profile, which is set up by the security administrator.

If your installation must further control which data set profiles within a HLQ can be created by a person with RACF Group-SPECIAL authorization, you can use zSecure Command Verifier policy profiles. In the

previous example, you probably do not use the policy profile on its own but define it with several profiles as follows:

```
C4R.DATASET.ID.ABCX.**              UACC(NONE)      UPDATE(XYZA1)
     Only user XYZA1 can create within ABCX.
C4R.DATASET.ID.ABCX.TEST*.**        UACC(NONE)      UPDATE(ABCX1,ABCX2,XYZA1)
     All three users can create "test" dataset profiles.
```

Note: In this example, all three users still need basic RACF authorization to create the data set profile. zSecure Command Verifier policy profiles enforce the naming conventions, but generally do not increase authorization of the terminal user.

Although discouraged, the previous set of example profiles can also be used to restrict the authority inherent in a CREATE level connect authorization. Even though the users ABCX1 and ABCX2 might be connected to the GROUP ABCX with CREATE authorization, they are not authorized to create data set profiles. An exception is implemented for the test data set profiles.

Policy profiles for enforcing resource naming conventions

In zSecure Command Verifier, the problem of authorizing the creation of profiles is solved by policy profiles.

These profiles are summarized in the following table. In most situations, you do not need these profiles, but they can help you restrict profile creation even further than already enforced by RACF.

Table 39. Profiles used for verification of RACF Resources. The entries in this table reflect the keywords that describe the name of new resources.

Command	Keyword	Profile
ADDSD DELDSD	*profile*	C4R.DATASET.ID.*hlq.rest-of-profile*
RDEFINE RDELETE	*profile*	C4R.*class*.ID.*profile*
RDEFINE RALTER	*ADDMEM*	C4R.*class*.ID.*member*
RDEFINE RALTER	*DELMEM*	C4R.*class*.ID.*member*

In these profiles, the variable *class* represents the class as specified on the **RDEFINE** command. When used for the ADDMEM and DELMEM keywords, the *class* represents the corresponding member class. The following examples can clarify the *class* used in the policy profiles.

Table 40. Profiles used for verification of RACF Resources. This table shows examples of the profile and class that is used for certain commands.

Command	profile	Class
RDEFINE DASDVOL xyzzyx	*xyzzyx*	DASDVOL
RDEFINE GDASDVOL pool1	*pool1*	GDASDVOL
RALTER GDASDVOL pool1 ADDMEM(xyzzyx)	*xyzzyx*	DASDVOL
RDEFINE GDASDVOL pool1 ADDMEM(xyzzyx)	*xyzzyx pool1*	DASDVOL GDASDVOL

In the policy profiles, the variable *profile* reflects the profile that is being defined, and the variable *member* reflects the member that is being manipulated. In the examples above, they are pool1 and xyzzyx, respectively. For data set profiles, the *profile* is sometimes split into two parts:

- The High-Level Qualifier (HLQ). This qualifier is the first qualifier of the data set profile. In RACF, the first qualifier must be an existing userid or GROUP.
- The remaining qualifiers (referred to as "rest-of-profile").

This split of the data set profile name is done to stress the special usage of the HLQ and highlight the similarity in form between the No-Store profiles that are described in "Policy profiles for managing your own data set profiles" on page 150 and the standard policy profiles that are described in the following section.

Policy profiles for creating RACF resource profiles

The topics in this section describe the policy profiles for creating RACF resource profiles and the corresponding access levels.

For the authority to create data set profiles for which the HLQ is the user ID of the terminal user, see "Policy profiles for managing your own data set profiles" on page 150.

- `C4R.DATASET.ID.`*hlq.rest-of-profile*

  This profile describes the authorization to create the data set profile that is specified by *hlq.rest-of-profile*. The policy profile can be a generic or discrete profile. When you define policy profiles, the profile part can contain plus signs to replace standard generic characters.

  zSecure Command Verifier does not pre-verify the normal RACF authorization to create the data set profile. If zSecure Command Verifier approves the creation of a certain data set profile, RACF still performs its own authorization verification. So, for data set profiles, the terminal user must also have authorization as described in "User authorization to create resource profiles" on page 157. The following access levels are available.

  No Profile Found
  This control is not implemented.

  NONE
  The user is not authorized to define the new data set profile.

  READ
  Same as NONE

  UPDATE
  The terminal user can create the data set profile, provided the terminal user has otherwise sufficient RACF authorization.

  CONTROL
  Same as UPDATE

- `C4R.`*class*`.ID.`*profile*
- `C4R.`*class*`.ID.`*member*

  These two policy profiles both refer to the same basic policy profile. Different names are used for the variables *profile* and *member* to describe the two different places where the values are obtained. The first profile describes the authorization to create the *profile* in *class*. This profile is used for the **RDEFINE** command. The second form of the same policy profile is used for the ADDMEM and DELMEM keywords on the **RDEFINE** and **RALTER** commands. See the discussion in the previous section for a general description and examples. The following access levels are available.

  No Profile Found
  Not Implemented. zSecure Command Verifier does not verify authorization to create *profile* in class *class*.

  NONE
  The user is not authorized to define the new *profile*.

  READ
  Same as NONE

  UPDATE
  The terminal user can create the *profile*. The terminal user still needs sufficient RACF authorization, like `clauth(`*class*`)`.

  CONTROL
  Same as UPDATE

Resource policy profiles for special applications

The profiles in the previous section and their translation can be used for some special applications. Examples of two special applications for the profiles are provided.

The first application is for profiles in the Global Access Checking Table. The second is for profiles in the PROGRAM class.

Global access checking table

The first special application relates to the definition of entries in the GAC table.

zSecure Command Verifier also performs more checking on the use of the ADDMEM keyword. It is possible to allow or disallow inclusion of certain entries in the GAC table. Situations are known where an authorized system administrator accidentally created an entry **/ALTER in the GAC table, resulting in ALTER access to all data sets in the system. This situation can be prevented by two policy profiles: One preventing definition of any GAC table entry, and one more specific profile that allows definition of GAC table entries that allow READ access. The following policy profiles can be used.

- C4R.GMBR.ID.**.* UACC(NONE)

  This profile explicitly uses an EGN feature that allows use of the ".**" in the middle of a profile to indicate that an unspecified number of qualifiers can be present. It also uses an explicit "*" as final qualifier to highlight the difference with the next profile. The UACC of the profile is NONE. It prevents anybody from defining an entry in the GAC table.
- C4R.GMBR.ID.**.R* UACC(UPDATE)

  This profile is more specific than the previous profile. The profiles are identical up to the R, which is not a generic character. The UACC of the profile in UPDATE. It allows anybody to add entries that have a last qualifier that starts with R.

If an authorized RACF administrator tries to add a GAC table entry like SYS1.LINKLIB/READ, the command is

```
RALT GLOBAL DATASET ADDMEM('SYS1.LINKLIB'/READ)
```

Because the GLOBAL resource class is matched in RACF with the pseudo member class GMBR that is needed for RACF internal reasons, zSecure Command Verifier verifies the following policy resource name:

```
C4R.GMBR.ID.+SYS1.LINKLIB+.READ
```

The translate mechanism that is described in “Generic and special characters in policy profiles” on page 144, translates all generic characters and some special characters into plus-signs. It also translates the slash (/) character into a period. This policy resource is covered by profile 2, and is thus allowed. If the administrator makes a mistake and by accident issue the command

```
RALT GLOBAL DATASET ADDMEM('SYS1.LINKLIB'/UPDATE)
```

the resulting policy resource name is

```
C4R.GMBR.ID.+SYS1.LINKLIB+.UPDATE
```

Because this policy resource is covered by profile 1, creation of the GAC table entry is denied.

The data set name in the ADDMEM keyword is always normalized. As part of the normalization process, quotation marks are placed around the data set name and a prefix is applied when needed. The translated value of the normalized data set name is used in the policy profiles. The previous example for SYS1.LINKLIB shows a translated normalized data set name as used in policy profiles.

The access level that in the ADDMEM keyword is not normalized. RACF accepts access levels even if they are abbreviated to a single letter (R=READ, U=UPDATE). The absence of normalization of the access level is the reason that the example profile is specified with the generic value R* as the last qualifier. The generic pattern matches all possible abbreviations of READ.

PROGRAM class

The second special application is one where the mandatory value profile is used for the UACC in combination with generic profile translation.

The purpose of this particular application is to prevent accidental definition of UACC=NONE on generic profiles in the PROGRAM class. There are situations where an administrator accidentally left out a UACC when defining generic program profiles for such as linklist data sets. It can result in rendering the entire system unusable. If all programs in SYS1.LINKLIB are protected against usage, all TSO commands, including all RACF commands, fail with an access violation. Using mandatory profiles, it is possible to prevent such a situation from occurring. The following profiles prevent default assignment of UACC=NONE for new generic profiles in the PROGRAM class:

```
C4R.PROGRAM.=UACC.+              UACC(READ)      APPLDATA('READ')
C4R.PROGRAM.=UACC.%+             UACC(READ)      APPLDATA('READ')
C4R.PROGRAM.=UACC.%%+            UACC(READ)      APPLDATA('READ')
  ...
C4R.PROGRAM.=UACC.%%%%%%%+       UACC(READ)      APPLDATA('READ')
```

The previous profiles control the UACC of all possible generic profiles in the PROGRAM class. If you want to extend these profiles to all including discrete profiles, it would be possible to replace these profiles with the following single profile.

```
C4R.PROGRAM.=UACC.*              UACC(READ)      APPLDATA('READ')
```

Either way, these profiles ensure that new PROGRAM profiles have a UACC=READ. It does not prevent an authorized administrator from later resetting the UACC to any other value. However, that requires a conscious decision and command by such an administrator. It is different from the default assignment that is used during the definition of the profile.

Selection of policy profiles for the resource profile owner

The OWNER describes who has control over the newly defined resource profile. Use these profiles and guidelines to control the specification of the owner.

These profiles apply to all four commands that allow specification of the OWNER; that is, **ADDSD**, **RDEFINE**, **ALTDSD**, and **RALTER**. In general, the processing for these profiles assumes that your installation's policy is to use the HLQ as OWNER. The last profile that is described in the following section (/HLQ) provides a control that can be used to indicate whether your installation wants to enforce such a policy or not. Again, the following description is split into several sets of profiles. The first is used to specify a mandatory or default value for the OWNER.

For Mandatory Value policy profiles, the third qualifier consists of an equals sign, followed by the keyword. So for the OWNER, the profile has the qualifier =OWNER. For Default profiles, the third qualifier consists of a forward slash, followed by the keyword. So, for the OWNER, the profile has /OWNER as third qualifier.

Table 41. Mandatory Value policy profiles for Owner of Resource Profiles. The entries in this table reflect the commands and keywords that describe the Mandatory or Default value for the OWNER of new resource profiles.

Command	Keyword	Profile
ADDSD	*profile*	C4R.DATASET.=OWNER.*profile*
ADDSD	*profile*	C4R.DATASET./OWNER.*profile*
RDEFINE	*profile class*	C4R.*class.*=OWNER.*profile*
RDEFINE	*profile class*	C4R.*class.*/OWNER.*profile*

The second set of profiles is used to describe controls on the value for the OWNER as specified by the terminal user. It also describes the general policies that can be used for the OWNER of resource profiles.

Table 42. Profiles used for Owner of Resource Profiles. The entries in this table reflect the commands and keywords that are specified by the terminal user that describe the owner of new or changed resource profiles.

Command	Keyword	Profile
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER.=RACUID(*n*)
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER.=RACGPID(*n*)
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER.=HLQ(*n*)
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER.*owner.profile*
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER./SCOPE.*owner.profile*
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER./GROUP.*owner.profile*
ADDSD ALTDSD	*profile owner*	C4R.DATASET.OWNER./HLQ.*owner.profile*
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER.=RACUID(*n*)
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER.=RACGPID(*n*)
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER.=HLQ(*n*)
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER.*owner.profile*
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER./SCOPE.*owner.profile*
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER./GROUP.*owner.profile*
RDEFINE RALTER	*profile class owner*	C4R.*class*.OWNER./HLQ.*owner.profile*

Mandatory and default value policy profiles for the owner

Use these policy profiles to specify the mandatory and default values for the OWNER of the new resource profile.

These profiles are only used for the **ADDSD** and **RDEFINE** commands.

- C4R.*class.*=OWNER.*profile*

  This profile specifies a mandatory overriding value for the OWNER of the newly defined resource profile. It is only used during **ADDSD** and **RDEFINE** processing. The OWNER value that is obtained by this Mandatory Value profile is not subject to more OWNER-related policy profiles.

  The qualifier =OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  No action. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command.

  UPDATE
  Same as READ

CONTROL
The control is not active for the terminal user. No mandatory value is supplied. The value for the OWNER as specified by the terminal user is used in the command.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value profiles, it leads to the odd situation that access NONE has the same net result as access OWNER.

The values that are accepted for the **APPLDATA** field are given as follows. The OWNER can be a user ID or GROUP.

BLANK
Any specified value of the new OWNER is suppressed, and replaced by the current GROUP of the terminal user.

=HLQ
Reflects the High-Level Qualifier HLQ of the resource profile. This setting typically makes sense only for data set profiles. If the HLQ is not an existing user ID or GROUP, the current GROUP of the terminal user is used instead.

=MYOWNER
Reflects the OWNER of the terminal user. If this OWNER is an existing user ID or GROUP, the value is used as the OWNER of the new resource profile. Otherwise, the current GROUP of the terminal user is used instead.

other
The specified user ID or GROUP is used as OWNER of the new resource profile. If this owner is not an existing user ID or GROUP, the current GROUP of the terminal user is used instead.

- C4R.*class.*/OWNER.*profile*

This policy profile specifies a default value for the OWNER of the newly defined resource profile. It is only used during **ADDSD** and **RDEFINE** processing. The OWNER that is to be used as default value is obtained from the **APPLDATA** field in the profile. The OWNER value that is obtained by this Default Value profile is not subject to more OWNER-related policy profiles. If the =OWNER profile is used to provide a value, the /OWNER profile is not used.

The qualifier /OWNER in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
The control is not implemented. No default value is supplied. This setting results in RACF providing a default for the OWNER (=the terminal user itself).

NONE
No default value is supplied. RACF does not provide a value for the OWNER. The command is rejected. Using this access level allows an installation to force the terminal user to explicitly specify a value for the OWNER.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ

CONTROL
The control is not active for the terminal user. No default value is supplied. Because the terminal user did not specify a value for the OWNER, RACF makes the terminal user the OWNER of the new profile.

The values that are accepted for the **APPLDATA** field are given as follows. The specified OWNER can be a user ID or GROUP.

BLANK
The current GROUP of the terminal user is inserted as the value for the OWNER.

=HLQ
Reflects the High-Level Qualifier (HLQ) of the resource profile. This setting typically makes sense only for data set profiles. If the HLQ is not an existing user ID or GROUP, the current group of the terminal user is used instead.

=MYOWNER
Reflects the OWNER of the terminal user. If this OWNER is an existing userid or GROUP, the value is used as the OWNER of the new resource profile. Otherwise, the current group of the terminal user is used instead.

other
The specified USERID or GROUP is used as OWNER of the new resource profile. If this owner is not an existing USERID or GROUP, the current group of the terminal user is used instead.

Resource policy profile owner verification

Use these policy profiles to verify a new OWNER in the **ADDSD**, **RDEFINE**, **ALTDSD**, or **RALTER** commands.

For data sets, RACF itself does sometimes impose a constraint that the owner must be connected to the HLQ-GROUP with at least CREATE authority. For general resources, or HLQ=USERID data sets, RACF does not impose any constraints on the OWNER. The policy profiles that are shown can be used to restrict the choice of new OWNERs. If the use of the specified OWNER is not accepted by any of the general policy rules =RACUID, =RACGPID, =HLQ, the explicit policy profile is used.

- C4R.*class*.OWNER.=RACUID(*n*)

  This profile specifies a special generic policy for the OWNER. The =RACUID stands for user ID of the terminal user. If the substring(=RACUID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. If you defined multiple of these profiles, only the one with the smallest numeric specification is used for matching the userids.

  This profile is a discrete profile. Only the single digit between parenthesis is variable, and must be specified from 1 to 8. It is not possible to use a true generic profile.

  If the specified OWNER is accepted, more verifications against the general policies like /SCOPE and /GROUP are performed.

  No profile found
  User ID of the terminal user is not used as naming convention or restriction for the OWNER.

  NONE
  The specified OWNER is not allowed. The command fails. This decision can be overruled by authorization to profile *owner.profile* described as follows.

  READ
  Same as NONE.

  UPDATE
  The specified OWNER is accepted.

  CONTROL
  Same as UPDATE.

- C4R.*class*.OWNER.=RACGPID(*n*)

  This profile specifies a special generic policy for the OWNER. The =RACGPID stands for the list of groups to which the terminal user is connected. The groups of all the user are used, independent of the setting of "list of group access checking". If the substring(=RACGPID,1,*n*) matches, this profile is used in preference to other profiles, independent of the value of *n*. It is only used if =RACUID(*n*) is not present or does not match. If you defined multiple of these profiles, only the one with the lowest value for *n* is used.

  This profile is a discrete policy profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

  If the specified OWNER is accepted, more verifications against the general policies like /SCOPE and /GROUP are performed.

No profile found
The GROUPs of the terminal user are not used as naming convention or restriction for the OWNER.

NONE
The specified OWNER is not allowed. The command fails. This decision can be overruled by authorization to profile *owner.profile* described as follows.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE

- C4R.*class.*OWNER.=HLQ(*n*)

This profile specifies a special generic policy for the OWNER. The special value =HLQ represents the High-Level Qualifier of the resource profile itself. This policy profile typically makes sense only for data set profiles. It can be used to enforce a naming convention, which states that the first *n* characters of a data set profile must match the first *n* characters of its owner.

The =HLQ stands for the HLQ of the resource profile in the command. If the substring(=HLQ,1,*n*) matches the specified OWNER, this profile is used in preference to other generic profiles, independent of the value of *n*. It is only used if =RACUID(*n*) and =RACGPID(*n*) are not present or do not match. If you defined multiple of these profiles, only the one with the lowest value for *n* is used.

This profile is a discrete policy profile. Only the single digit between parenthesis is variable and must be specified from 1 to 8. It is not possible to use a true generic profile.

If the specified OWNER is accepted, more verifications against the general policies like /SCOPE and /GROUP are performed. zSecure Command Verifier does not test if the specified OWNER is a valid user ID or GROUP.

No profile found
The target resource profile is not used as naming convention or restriction for its OWNER.

NONE
The specified OWNER is not allowed. The command fails. This decision can be overruled by authorization to profile *owner.profile* described as follows.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

If any of the above three profiles allow the specified OWNER, the next profile rule is skipped. Processing continues with the /SCOPE, /GROUP, and /HLQ policies. If the previous profiles did not authorize the use of a certain OWNER, the next profile is used as alternative authorization method.

- C4R.*class.*OWNER.*owner.profile*

The primary purpose of this control is to specify a policy if none of the general policies apply. The variable *owner* represents the new OWNER of the resource *profile*. It allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The OWNER as verified by this policy profile is still subjected to the additional policy profiles (/SCOPE, /GROUP, /HLQ) as described here:

No profile found
This control is not implemented.

NONE
The command is rejected.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted.

CONTROL
Same as UPDATE.

More policy profiles for the resource profile owner

Use these guidelines to implement profiles to specify general rules for the new OWNER. By using more specific or fully qualified profiles, it is possible to specify that some resource profiles are exempt from such a restriction.

Aside from the profiles that are intended to enforce a naming convention, it is also possible to implement a policy that is based on the existing RACF group hierarchy.

The following profile rules are used as an extra set of policies. If the specified OWNER is accepted by any of the rules above, it is verified against the following three policies. If it fails any of these policies, the command is rejected.

C4R.*class*.OWNER./SCOPE.*owner.profile*
This profile is used to control if the new OWNER as specified by the terminal user must be within the scope of a group-SPECIAL attribute. This profile can prevent the terminal user from giving away resource profiles that are within the scope of a group-SPECIAL attribute.

The variables *profile* and *owner* represent the affected resource profile and the new OWNER of the resource profile. It allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /SCOPE in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

Specifying a user ID as the new owner is always considered to be outside the administrative scope of the terminal user.

For terminal users with group-special or system-special, use of their administrative authority over the new owner is recorded through the audit-only policy profile

- C4R.USESCOPE.*group*

Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the new owner specified for the data set or resource. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

The following access levels are supported for the /SCOPE policy profile:

No profile found
The group-SPECIAL scope of the terminal user is not used to control the new OWNER of user profiles.

NONE
If the specified new OWNER is outside the scope of a group-SPECIAL attribute of the terminal user, the command fails.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted, irrespective of the scope of the terminal user.

CONTROL
Same as UPDATE.

C4R.*class*.OWNER./GROUP.*owner.profile*

The profile is used to control if the specified OWNER must be a RACF GROUP or not. This profile is verified independently of the other profiles. If either the =OWNER or the /OWNER profile is used, that this policy rule is bypassed.

The variables *profile* and *owner* represent the affected resource profile and the new OWNER of the resource profile. It allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /GROUP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. The specified OWNER can be a GROUP and a user ID.

NONE
If the specified owner is an existing RACF group, the command is accepted. In all other situations, the command fails.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted even if it does not represent an existing group. If the specified OWNER is not a valid entry, the command is rejected by RACF.

CONTROL
Same as UPDATE.

C4R.*class*.OWNER./HLQ.*owner.profile*

This profile is used to control if the OWNER as specified by the terminal user must be the same as the HLQ of the resource profile. It typically makes sense only for data set profiles.

The *profile* and *owner* values represent the affected resource profile and the new OWNER of the profile. It allows specification of exceptions to the general rule. The most specific profile is used by zSecure Command Verifier.

The qualifier /HLQ in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

No profile found
This control is not implemented. The specified OWNER can be different from the HLQ.

NONE
The specified new OWNER must be the same as the current (or new) HLQ.

READ
Same as NONE.

UPDATE
The specified OWNER is accepted, irrespective of the value of the HLQ.

CONTROL
Same as UPDATE.

Controlling the UACC and access list

Aside from the authority to create profiles, the most important part of a resource profile is its access specification. In zSecure Command Verifier, all forms of access management are supported by policy profiles.

RACF also has a fast path option from the Global Access Checking (GAC) table. In zSecure Command Verifier, this table is not directly controlled as an access mechanism. However, since the GAC table is defined by RACF profiles in the GLOBAL resource class, it can also be controlled by zSecure Command Verifier policy profiles.

Note: The authorization to issue the **PERMIT** or **ALTDSD** command is also subject to the No-Store control described in “Policy profiles for managing your own data set profiles” on page 150.

The following tables summarize the various policy profiles that are used for the different access mechanisms. The first one provides an overview of the UACC control and the standard Access List. The table also summarizes the additional policies for access management. The additional policies are enforced after the command is approved according to the other ACL policies.

These additional policy profiles can be used to prevent data set groups from being placed on the access list of resources and to prevent granting access to individual users or groups outside a possible group-special scope. A group is considered to be a data set group if a data set profile is defined with the group as the HLQ.

Table 43. Profiles used for verification of RACF access. The entries in this table reflect the commands and keywords that are used to manage access.

Command	Keyword	Profile
ADDSD RDEFINE	*profile*	C4R.*class.*=UACC.*profile*
ADDSD RDEFINE	*profile*	C4R.*class.*/UACC.*profile*
ADDSD RDEFINE ALTDSD RALTER	*profile*	C4R.*class.*UACC.*uacc.profile*
PERMIT	*userid*	C4R.*class.*ACL.=RACUID.*access.profile*
PERMIT	*group*	C4R.*class.*ACL.=RACGPID.*access.profile*
PERMIT	*profile* ID(*id*)	C4R.*class.*ACL.=PUBLIC.*profile*
PERMIT	*profile* ID(*userid*) AC(*access*)	C4R.*class.*ACL.*userid.access.profile*
PERMIT	*profile* ID(*) AC(*access*)	C4R.*class.*ACL.=STAR.*access.profile*
PERMIT	*profile* FROM(model)	C4R.*class.*ACL.=FROM.*profile*
PERMIT	*profile* RESET(Standard)	C4R.*class.*ACL.=RESET.*profile*
PERMIT	*profile* ID(*group*)	C4R.*class.*ACL.=DSN.*group.profile*
PERMIT	*profile* ID(*userid*)	C4R.*class.*ACL./GROUP.*userid.profile*
PERMIT	*profile* ID(*userid*)	C4R.*class.*ACL./SCOPE.*userid.profile*

Note: In this table, the policy profiles for granting access to yourself are repeated for completeness only. These policy profiles are described in "Policy profile selection for self-authorization" on page 151.

The following table summarizes the policy profiles that are used for the conditional access list. This summary describes the authority to use certain when-condition classes and the authority to reset the conditional access list.

Table 44. Profiles used for verification of RACF access. The entries in this table reflect the commands and keywords that are used to manage access.

Command	Keyword	Profile
PERMIT	*profile* WHEN(*whenclass*)	C4R.*class.*CONDACL.*whenclass.profile*
PERMIT	*profile* RESET(when)	C4R.*class.*CONDACL.=RESET.*profile*

In this table, some of the general policy rules use the same qualifiers for special keywords as those qualifiers used for ACL entries. For example, the /SCOPE qualifier is the same as the qualifier for the regular *userid*. If you want to explicitly allow one group administrator to put a group GROUPX on any access list, you must define several profiles to explicitly handle these general policies.

For example, consider the rule:

```
ADMINX is only allowed to put GROUPX on any ACL.
Any other ACLid is "protected" and can only be permitted by SUPERADM.
```

For this policy rule, you would need the following profiles:

```
C4R.*.ACL.*.**            uacc(none) update(superadm)
C4R.*.ACL.groupx.**       uacc(none) update(superadm,adminx)
```

These two policy profiles ensure that all ACL entries are allowed to SUPERADM, and that GROUPX is allowed for ADMINX. You might also want to explicitly spell out the following more policy profiles:

```
C4R.*.ACL./SCOPE.**       uacc(update)
C4R.*.ACL./GROUP.**       uacc(update)
C4R.*.ACL.=STAR.**        uacc(update)
C4R.*.ACL.=DSN.**         uacc(update)
C4R.*.ACL.=RESET.**       uacc(update)
```

You cannot use generics for qualifiers like /SCOPE or /GROUP. These five profiles ensure that the general policies are not implemented. You can also grant access CONTROL. If you want to explicitly deny the authority to put `userid` "*" on the access list, use the following profile:

```
C4R.*.ACL.=STAR.**    uacc(none) update(superadm)
```

The example policy rule did not specify how ADMINX is allowed to put GROUPX on any ACL. Normally RACF allows only management of access list entries that are based on the resource profile itself, and not based on the ACL entry. To manage all ACLs, the terminal user requires System-SPECIAL, or Group-SPECIAL authorization in several GROUPS.

Controlling access to the resource profile UACC

A set of three profiles controls the setting of the UACC of the newly defined resource profile.

The first profile can be used to specify a mandatory value for the UACC. One of the primary purposes of this profile is to prevent an administrator from accidentally defining profiles in the program class with a UACC=NONE. Defining such a profile can shut down the entire system, without an easy way of recovery. Mandatory UACC profiles can be used to prevent such a situation from occurring. The Mandatory Value profile is only used for the **ADDSD** and **RDEFINE** command.

The second policy profile provides a default UACC in case no UACC is specified and no mandatory profile is enforced.

The last policy profile is intended to verify the terminal user specified value. It is used both during the creation of new resource profiles and during the modification of the UACC value of existing resource profiles.

- C4R.*class.*=UACC.*profile*

  This profile specifies a mandatory value for the UACC. The **APPLDATA** field of this profile is extracted, and inserted as the value. The **APPLDATA** field must contain one of the standard RACF access level values NONE, EXECUTE, READ, UPDATE, CONTROL, or ALTER. Any other value is interpreted as NONE. The access of the terminal user to the policy profile determines whether the value found can be used.

  The qualifier =UACC in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No profile found
  The control is not implemented. No mandatory value is enforced.

  NONE
  The mandatory value is not used. The value that is specified by the terminal user is accepted, or RACF provides a default.

  READ
  The **APPLDATA** of the policy profile is extracted and inserted as the UACC value of the new profile.

UPDATE
Same as READ

CONTROL
The control is not active for the terminal user. No mandatory value is supplied. The value for the UACC as specified by the terminal user is used in the command.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value profiles, it leads to the odd situation that access NONE has the same net result as access CONTROL.

The UACC value that is obtained by this Mandatory Value profile is not subject to more UACC-related policy profiles.

- C4R.*class.*/UACC.*profile*

This profile specifies a default value for the UACC. This policy profile is only used if no value for the UACC was specified on the RACF command. The **APPLDATA** field of this policy profile is extracted, and inserted as the value.

The qualifier /UACC in the policy profile cannot be covered by generic characters. It must be present in the exact form shown. The **APPLDATA** field must contain one of the standard RACF access level values NONE, EXECUTE, READ, UPDATE, CONTROL, or ALTER. Any other value is interpreted as NONE. The access of the terminal user determines whether the **APPLDATA** value found must be inserted in the command or not.

No Profile Found
The policy is not implemented.

NONE
The Default value is not used. The possible default value that is provided by RACF is used.

READ
The **APPLDATA** of the policy profile is extracted and inserted as the UACC value of the new profile.

UPDATE
Same as NONE.

CONTROL
The default UACC policy rule is not applicable to this terminal user.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Default Value profiles, it leads to the odd situation that access NONE has the same net result as access CONTROL.

The UACC value that is obtained by this Mandatory Value profile is not subject to more UACC-related policy profiles.

- C4R.*class.*UACC.*uacc.profile*

This profile is used to verify the UACC value as specified by the terminal user. The variable *uacc* represents the UACC level as specified. Accepted values are all the RACF allowed values for the UACC, that is, NONE, EXECUTE, READ, UPDATE, CONTROL, or ALTER. This profile is not used for the **ADDSD** and **RDEFINE** command if the terminal user explicitly specified the RACF default value NONE. The value NONE is always accepted when you define new resources, independent of the specification of the corresponding UACC policy rule. For the **ALTDSD** and **RALTER** commands, all values for the UACC are verified by using the UACC policy rules.

No Profile Found
The control is not implemented. The terminal user specified value is accepted.

NONE
Specified UACC not allowed. The command is rejected.

READ
Same as NONE.

UPDATE
The UACC value that is specified by the terminal user is accepted.

CONTROL
Same as UPDATE.

Policy profiles for the resource profile ACL

zSecure Command Verifier uses several policy profiles to control the entries in the access list and in the conditional access list.

Two main types of profiles are being used. The first type specifies the combination of the ID (in RACF terms, the user ID, but in reality a user ID or GROUP) and the access level. The second type of profile controls usage of the WHEN (when-class) keyword. The current section describes the access level and the ID used in both the Standard ACL and the Conditional ACL. The following section describes the *class* used in the conditional access list.

In the profiles for the access list, a special qualifier is supported for usage of `ID(*)` on the ACL. In the policy profiles, it is represented by the special qualifier =STAR. If the default translation for profiles is used, it results in usage of a plus-sign in the policy profiles to represent the use of an asterisk in the Access List.

Policy profiles for granting access to yourself are included in the preceding table for reference only. These policy profiles are described in "Policy profile selection for self-authorization" on page 151.

All policy profiles in this section are only evaluated for non-public resources and for those public resources for which the policy profiles allow the terminal user to modify the access list. Resource profiles are considered public if their UACC is greater than NONE, or if `ID(*)` is granted access greater than NONE. For a discussion of public resource access control, see "General policy profiles for the ACL" on page 174. If an applicable *public* policy profile denies updating the access list, none of the policy profiles described in the current section are evaluated.

In the policy profiles, the qualifier *user* represents the ID specified in the command. This can be either a RACF user ID or a RACF GROUP. The qualifier *access* represents the access level. Possible values are ALTER, CONTROL, UPDATE, READ, EXECUTE, NONE, or the special value DELETE, which is used to represent the DELETE keyword on the PERMIT command.

- C4R.*class*.ACL.*user.access.profile*

  This profile describes the authority to grant user ID or GROUP *user* access to the *profile* in the resource class *class*, at the *access* level. The following list contains examples of policy profiles.

  ```
  C4R.DATASET.ACL.IBMUSER.UPDATE.SYS1.**
  C4R.FACILITY.ACL.IBMUSER.UPDATE.ICHBLP
  C4R.DATASET.ACL.*.*.**
  ```

  In general, the resource profile is expected to be covered by a generic pattern that consists of the HLQ followed by a double asterisk. As a backstop profile for all resource profiles, a profile similar to the third previous example is expected to be used, in which the required qualifiers are coded explicitly by the use of single asterisk generic characters.

  No Profile Found
  The policy is not implemented for this situation.

  NONE
  The specified access is not allowed. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The specified access is allowed for this *user* and resource profile.

  CONTROL
  Same as UPDATE.

- C4R.*class*.ACL.=STAR.*access.profile*

This profile represents the use of ID(*) on the access list. In general, ID(*) has the same net result as use of the UACC. Some organizations want to make an explicit distinction between all users of the system and all RACF defined users of the system. In well-protected systems, there is no difference between the two categories. For this reason, zSecure Command Verifier implements a special qualifier to quickly recognize and control the access level for ID(*).

In contrast to the general rule, the special value =STAR can be covered by a generic pattern. For example, the profile C4R.*class*.ACL.*.*profile* can be used to prevent all changes to the ACL.

Many installations are expected to define a profile similar to

```
C4R.DATASET.ACL.=STAR.*.**   UACC(NONE)
```

to prevent the use of ID(*) on any data set access list. The access levels that are supported for this policy profile are the same as those levels for the regular access list policy profiles.

No Profile Found
The policy is not implemented for this situation.

NONE
The specified access is not allowed. The command is rejected.

READ
Same as NONE.

UPDATE
The specified access is allowed for ID(*) and this resource profile.

CONTROL
Same as UPDATE.

- C4R.*class*.ACL.=FROM.*profile*

This profile controls the authorization to copy an existing ACL from one profile to another. The RACF **PERMIT FROM** function is a quick way for issuing **PERMIT** commands for several ACL entries. The ACL entries of the model profile are only added to the existing ACL for the target profile. Existing ACL entries in the target profile are not changed.

The main reason an installation might choose to implement this policy is that the copied ACL can contain entries that do not fit the policy rules. The ACL.=FROM profile is used to control authorization to use this copy function for access lists. The model profile name is not included in the policy profile.

In contrast to the general rule, the special value =FROM can be covered by a generic pattern. For example, the profile C4R.*class*.ACL.*.*profile* can be used to prevent all changes to the ACL.

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user is not authorized to copy an existing ACL into this *profile*

READ
Same as NONE.

UPDATE
Copying an existing ACL is allowed.

CONTROL
Same as UPDATE.

- C4R.*class*.ACL.=RESET.*profile*

This profile controls the authorization to reset the entire access list, and thus removing all entries from the access list. The RACF **PERMIT RESET** function is a quick way for issuing **PERMIT DELETE** commands for all entries in the access list. The ACL.=RESET profile is used to control authorization to reset the standard access list. A similar profile is described in the following section for the authorization to reset the **conditional** access list.

In contrast to the general rule, the special value =RESET can be covered by a generic pattern. For example, the profile C4R.*class*.ACL.*.*.*profile* can be used to prevent all changes to the ACL.

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user is not authorized to reset the ACL for *profile* in *class*.

READ
Same as NONE.

UPDATE
Reset of the standard ACL is allowed.

CONTROL
Same as UPDATE.

General policy profiles for the ACL

Many installations have general policy rules about the entries that can be placed on access lists. zSecure Command Verifier currently implements several of these general policies.

You can use these policies for the following purposes:

- Use the first policy to prevent any updates to the access list of so called *public* resources. Updates to the access list are most likely to be redundant, or needed only for exceptional situations.
- Use the second policy to prevent the granting of access to data set groups. A group is considered to be a data set group if a data set profile is defined with the group as HLQ.
- Use the third, fourth, and fifth policies to prevent user IDs from being put on the access list. Only groups are allowed.
- Use the last policy to prevent the granting of access to entries that are not within the scope of the Group-SPECIAL authorization of the decentralized administrator.

zSecure Command Verifier currently implements these general policies:

- C4R.*class*.ACL.=PUBLIC.*profile*

  This policy profile can be used to prevent modifications to the access list of public resources. Resource profiles are considered public, if their UACC is greater than NONE, or if ID(*) is granted access greater than NONE. If the policy applies, and the terminal user does not have sufficient access, the command is rejected, and none of the other access lists related policy profiles are evaluated. The policy applies only to changes to the access list made by the PERMIT command. Changes to the UACC itself (which can be the reason that the resource is considered a public resource) are not controlled by this policy profile. All changes to the ACL, including the access of ID(*), are subject to this policy. The access that is granted to ID(*) is also controlled by the =STAR policy.

  The qualifier =PUBLIC in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No Profile Found
  The policy is not implemented.

  NONE
  The terminal user is not authorized to manage the access list of the resource profile.

  READ
  The terminal user is authorized to delete entries from the access list of the resource profile. So, the command

  ```
  PERMIT profile ID(any-id) DELETE
  ```

  is allowed, provided none of the other policies or RACF authorizations prevent the command from executing.

UPDATE
The terminal user is authorized to manage the access list of this public resource profile.

CONTROL
Same as UPDATE.

- C4R.*class*.ACL.=DSN.*group.profile*

  This policy profile can be used to prevent granting access to data set groups. A group is considered to be a data set group if a data set profile is defined with the group as HLQ. The variable *group* is the entry that is being placed on the access list. In access lists, the term user ID is used to indicate any type of entry user ID or GROUP. In this profile, the term *group* is explicitly used to indicate that policy profile is only applicable to GROUPs. Since it is normal for user IDs to have data set profiles that are defined, the policy profile does not apply if the access list entry is a user ID. Most installations can use a double asterisk (**) to cover the *group* and *profile*. Exceptions to the general rule can be made by specifying more qualifiers. zSecure Command Verifier uses only the most specific profile.The qualifier =DSN in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No Profile Found
  The policy is not implemented for this situation.

  NONE
  The terminal user is not authorized to place data set groups on the access list.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to put GROUPs in the access list even if a data set profile is defined with a HLQ equal to this GROUP.

  CONTROL
  Same as UPDATE.

- C4R.*class*.ACL./GROUP.*userid.profile*

  This policy profile can be used to prevent user IDs from being put on access lists. It applies to the standard access list and the conditional access list. If the terminal user does not have sufficient access to this policy profile, only RACF GROUPs can be placed on access lists. The variable *userid* is the user that is being placed on the access list. In access lists, the term *userid* is used to indicate any type of entry user ID or GROUP. In this profile, the term user ID is used in the limited sense of the ID of a user. In most installations, the user ID and profile are to be covered by a double asterisk. Exceptions to the general rule can be made by specifying more qualifiers. zSecure Command Verifier uses only the most specific profile.

  This policy profile can also be used to override the two more /GROUP policy profiles (C4R.*class*.ACL./GROUP.=HLQTYPE.USER and C4R.*class*.ACL./GROUP.=HLQTYPE.GROUP).

  The qualifier /GROUP in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

  No Profile Found
  The policy is not implemented for this situation.

  NONE
  The terminal user is not authorized to place users on the access list.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to put individual user IDs and GROUPs on the ACL.

  CONTROL
  Same as UPDATE.

- C4R.*class*.ACL./GROUP.=HLQTYPE.USER

This policy profile can be used to prevent placing user IDs on the access list of user data sets. A data set is considered a user data set if the high-level qualifier (the first qualifier) is defined in RACF as a user ID. The policy profile applies to the standard access list and to the conditional access list. If the entry to be placed on the access list is a user ID, and the data set is a user data set, the terminal user must have sufficient access to the current policy profile. If the terminal user does not have sufficient access, the C4R.*class.*ACL./GROUP.*userid.profile* policy can be used to override the current policy. If the C4R.*class.*ACL./GROUP.*userid.profile* policy also does not allow placing a user ID on the access list, the command is rejected. In that case, only existing RACF GROUPs can be permitted access.

Although this policy can be defined by using a generic profile, the qualifiers /GROUP.=HLQTYPE must be present in the policy profile in the exact form shown.

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user is not authorized to put users on the access list. The C4R.*class.*ACL./GROUP.*userid.profile* policy might still allow users to be placed on the access list of user data sets.

READ
Same as NONE.

UPDATE
The terminal user is authorized to put individual user IDs and GROUPs on the access list of user data sets.

CONTROL
Same as UPDATE.

- C4R.*class.*ACL./GROUP.=HLQTYPE.GROUP

This policy profile can be used to prevent placing user IDs on the access list of group data sets. A data set is considered a group data set if the high-level qualifier (the first qualifier) is defined in RACF as a GROUP. The policy profile applies to the standard access list and to the conditional access list. If the entry to be placed on the access list is a user ID, and the data set is a group data set, the terminal user must have sufficient access to the current policy profile. If the terminal user does not have sufficient access, the C4R.*class.*ACL./GROUP.*userid.profile* policy can be used to override the current policy. If the C4R.*class.*ACL./GROUP.*userid.profile* policy also does not allow placing a user ID on the access list, the command is rejected. In that case, only existing RACF GROUPs can be permitted access.

Although this policy can be defined by using a generic profile, the qualifiers /GROUP.=HLQTYPE must be present in the policy profile in the exact form shown.

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user is not authorized to place users on the access list. The C4R.*class.*ACL./GROUP.*userid.profile* policy might still allow users to be placed on the access list of group data sets.

READ
Same as NONE.

UPDATE
The terminal user is authorized to put individual user IDs and GROUPs on the access list of group data sets.

CONTROL
Same as UPDATE.

- C4R.*class.*ACL./SCOPE. *userid.profile*

This general policy profile can be used to prevent the terminal user from placing people outside their Group-SPECIAL scope on access lists. It applies to the standard access list and the conditional access list. If the terminal user does not have sufficient access to this policy profile, only users and groups within the RACF Group-SPECIAL scope of the terminal user can be placed on access lists. The variable

userid is the entry that is being placed on the access list (either a user ID or GROUP). In this profile description, the RACF term *userid* is used in this special meaning. Exceptions to the general rule can be made by specifying more qualifiers for the *profile*. In most installations, it is expected that the *profile* is covered by a double asterisk (**). zSecure Command Verifier uses only the most specific profile.

The /SCOPE qualifier in the policy profile cannot be covered by generic characters. It must be present in the exact form shown.

For terminal users with group-special or system-special, granting access to a user or group within their administrative scope is recorded through the audit-only policy profile.

– C4R.USESCOPE.*group*

Successful access with UPDATE authority to this profile is recorded through SMF. The qualifier *group* represents the lowest group in the RACF group-tree that grants group-SPECIAL authority over the user or group that is specified in the PERMIT command. If the terminal user has system-SPECIAL, the fixed value **=SYSTEM** is used.

The /SCOPE policy profile uses only the RACF group-SPECIAL attribute for determining whether an entry can be added. Implementing this profile can result in normal users not being able to modify the access list of their own data set profiles because all entries are considered outside their scope. A similar effect can be obtained more directly by using the No-Store function that is described in "Policy profiles for managing your own data set profiles" on page 150.

The following access levels are supported for the /SCOPE policy profile:

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user can add or modify only Access List entries that are within the scope of Group-SPECIAL. If the terminal user does not have Group-SPECIAL scope, all access list entries are considered outside the scope.

READ
Same as NONE.

UPDATE
The terminal user is authorized to add or modify Access List entries that are outside the Group-SPECIAL scope.

CONTROL
Same as UPDATE.

Policy profiles for the conditional access list

Entries in the conditional access list consist of several parts: the ID, the access level, the class, and the resource in that class. To distinguish the class and resource used in the conditional access list from other occurrences, they are often called the *when-class* and the *when-resource*.

The ID and Access level related policy profiles described in "General policy profiles for the ACL" on page 174 also apply to the conditional access. There is no policy profile to control the name of the resource used on the conditional access list. The only remaining policy profile is for the when-class. Traditionally, the class used on the conditional access list is the PROGRAM class. Although other classes are also possible, it is still the most frequently used. The following policy profiles are used to describe the when-class.

- C4R.*class.*CONDACL.*whenclass.profile*

  This profile controls the use of the WHEN keyword to manage entries on the conditional access list. It is used in combination with the policy profiles for the standard access list. The CONDACL profile controls the use of the *whenclass*. The *whenclass* in the profile is normally the first parameter of the WHEN keyword in the **PERMIT** command. An exception is made for the form of the **PERMIT** commands that

uses the CRITERIA term. In that situation, the criteria-name is used instead. For example, when the following command is issued:

```
PERMIT DSND.USER01.HOMEWORK_GRADES.SELECT CLASS(MDSNTB) ID(STUDENT)
       WHEN(CRITERIA(SQLROLE('TEACHING ASSISTANT'))) ACCESS(READ)
```

The *whenclass* used in the policy profile is SQLROLE. This setting results in access verification to the following two resources:

```
C4R.MDSNTB.ACL.STUDENT.READ.DSND.USER01.HOMEWORK_GRADES.SELECT
C4R.MDSNTB.CONDACL.SQLROLE.DSND.USER01.HOMEWORK_GRADES.SELECT
```

Similarly, if using `WHEN(CRITERIA(SMS(DSENCRYPTION)))`, the following CONDACL policy resource is used:

```
C4R.class.CONDACL.SMS.profile
```

The following access levels are supported in the policy profile.

No Profile Found
The policy is not implemented for this situation.

NONE
The terminal user is not authorized to specify conditional access list entries.

READ
Same as NONE.

UPDATE
Creating conditional access list entries is allowed. The `ACL.`*user.access* profiles determine which entries on the conditional access list are allowed.

CONTROL
Same as UPDATE.

- `C4R.`*class.*`CONDACL.=RESET.`*profile*

  This profile controls the authorization to reset the entire conditional access list, and thus remove all entries from the conditional access list. The RACF **`PERMIT RESET(WHEN)`** function is a quick way for issuing **`PERMIT DELETE`** commands for all entries in the conditional access list. The `CONDACL.=RESET` profile is used to control authorization to reset the conditional access list.

  In contrast to the general rule, the special value=RESET can be covered by a generic pattern. For example, the profile `C4R.`*class.*`CONDACL.*.`*profile* can be used to prevent all changes to the conditional access list.

  No Profile Found
  The policy is not implemented for this situation.

  NONE
  The terminal user is not authorized to reset the Conditional ACL for *profile* in *class*.

  READ
  Same as NONE.

  UPDATE
  Reset of the Conditional ACL is allowed.

  CONTROL
  Same as UPDATE.

Further identification of the resource

Discrete data set profiles also contain information about the volume and type of device where the RACF indicator is kept.

The following two profiles control the use of these keywords.

- C4R.*class*.VOLUME.*dsname*

  This profile controls the use of the VOLUME keyword on the **ADDSD** command, and the **ADDVOL**, **DELVOL**, ALTVOL keywords on the **ALTDSD** command.

  No profile found
  This control is not implemented.

  NONE
  Specifying or Modifying Volume names on the **ADDSD** and **ALTDSD** commands is not allowed and causes the command to fail.

  READ
  Same as NONE.

  UPDATE
  Explicit selection and Management of the VOLUME for discrete data set profiles is allowed.

  CONTROL
  Same as UPDATE

- C4R.*class*.UNIT.*dsname*

  This profile controls the use of the UNIT keyword on the **ADDSD** command.

  No profile found
  This control is not implemented.

  NONE
  Specifying or Modifying the Unit-type on the **ADDSD** and **ALTDSD** commands is not allowed and causes the command to fail.

  READ
  Same as NONE.

  UPDATE
  Explicit selection and Management of the Unit type for discrete data set profiles is allowed.

  CONTROL
  Same as UPDATE

Policy profiles for DFP segment management

The DATASET DFP segment contains default information for some attributes when creating new data sets. Using the Command Verifier policies that are described in this section allows control of which users can manage these attributes for new data sets.

The RESOWNER field determines which RACF user or group is used to determine the default values for the DATACLAS, MGMTCLAS, and STORCLAS of the new data set. The DATAKEY field determines the label of the default encryption key that is used to encrypt the new data set. Without zSecure Command Verifier, a user with update access to the DFP segment (either through system special, or through access to the FIELD profiles) can manage these fields. Command Verifier enables you to control this access on a data set and user level. Table 45 on page 179 shows the policy profiles that are implemented.

Table 45. Keywords and policy profiles for managing the DFP segment information

Keyword	Value	Profile
RESOWNER	n/a	C4R.DATASET.DFP.RESOWNER.*profile*
DATAKEY	n/a	C4R.DATASET.DFP.DATAKEY.*profile*

The profiles in the preceding table describe the policies that can be used to verify the keywords that the terminal user enters. Currently, Command Verifier does not provide support for the value specified for the parameter on these keywords. The following list shows detail information about these policies and the supported access levels.

- C4R.DATASET.DFP.RESOWNER.*profile*

This policy profile describes the authorization to set the default RESOWNER for the data set profile. If no RESOWNER has been specified, the HLQ of the data set is used to determine the default RESOWNER. The DFP segment of the RESOWNER determines the default values for the DATACLAS, MGMTCLAS, and STORCLAS of the new data set. The RESOWNER also determines which ID must have access to these SMS resources. The way DFSMS uses these fields can be influenced through the USE_RESOWNER and ACSDEFAULTS options in parmlib member IGDSMS*xx*. Command Verifier supports the following access levels to the policy profile:

No profile found
This control is not implemented.

NONE
The terminal user is not authorized to specify or remove the RESOWNER. The command is rejected.

READ
Same as NONE.

UPDATE
The terminal user is authorized to specify or remove the RESOWNER.

CONTROL
Same as UPDATE.

- C4R.DATASET.DFP.DATAKEY.*profile*

  This policy profile describes the authorization to set the default DATAKEY for the data set profile. The specified value is the label of the encryption key in the CKDS that is used to encrypt the new data set that is covered by the data set profile. Command Verifier supports the following access levels to the policy profile:

  No profile found
  This control is not implemented.

  NONE
  The terminal user is not authorized to specify or remove the DATAKEY. The command is rejected.

  READ
  Same as NONE

  UPDATE
  The terminal user is authorized to specify or remove the DATAKEY.

  CONTROL
  Same as UPDATE.

Policy profiles for MFPOLICY segment management

The MFADEF resource class is used to record and define information that IBM Multi-Factor Authentication for z/OS uses. The MFADEF class has profiles that describe the available factors and policies. In the following sections, the explicit phrase "Command Verifier policy" is used whenever there might be a possible confusion between the MFA policy and the zSecure policy.

MFADEF profiles that describe factors have the string FACTOR as first qualifier. The remaining qualifiers are the name of a factor that the IBM Multi-Factor Authentication for z/OS product supports. When specifying factors for users or policies, the factor must be defined through an MFADEF profile. Information about the factor is also maintained in the MFA segment of the MFADEF profile. The contents of the MFA segment cannot be managed using RACF commands. Consequently, aside from the general policy to create or remove MFA segments, there are no related zSecure Command Verifier policies.

MFADEF profiles that describe policies have the string POLICY as first qualifier. The remaining qualifiers are the name of a policy that the IBM Multi-Factor Authentication for z/OS product supports. When assigning policies to users, the policy must be defined through an MFADEF profile. Information about the policy is defined in the MFPOLICY segment of the MFADEF profile. You can use RACF commands to maintain the MFPOLICY segment. zSecure Command Verifier provides policies to control adding, removing, and maintaining the contents of the MFPOLICY segment.

Table 46. Profiles used for verification of MFPOLICY values		
Keyword	**Value**	**Profile**
FACTOR ADDFACTOR DELFACTOR NOFACTOR	factor	C4R.MFADEF.MFPOLICY.FACTOR.*factor-name.policy-profile* For NOFACTOR, the value for the *factor-name* in the Command Verifier policy profile is a plus sign (+).
TOKENTIMEOUT	*value*	C4R.MFADEF.MFPOLICY.ATTR.TOKENTIMEOUT.*policy-profile*
REUSE	YES/NO	C4R.MFADEF.MFPOLICY.ATTR.REUSE.*policy-profile*

The profiles in the preceding table describe the policies that can be used to verify the keywords and values as entered by the terminal user. Currently, Command Verifier provides policy support for the values that are specified for the FACTOR, but not for the values that are specified for TOKENTIMEOUT and REUSE.

- C4R.MFADEF.MFPOLICY.FACTOR.*factor-name.policy-profile*

  This policy profile describes the authorization to specify, add, or remove a factor name from the policy. When the terminal user uses the NOFACTOR keyword, RACF removes all factors from the policy. For that keyword, Command Verifier does not verify the names of the policies individually. Instead, the special value + (plus sign) is used for the *factor-name*. The *factor-name*, including the special value + can be covered using a generic pattern in the Command Verifier policy profile.

  Command Verifier supports the following access levels to the policy profile:

  No profile found
  This control is not implemented.

  NONE
  The terminal user is not authorized to specify or remove the FACTOR for the policy. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to specify or remove the FACTOR for the policy.

  CONTROL
  Same as UPDATE.

- C4R.MFADEF.MFPOLICY.ATTR.TOKENTIMEOUT.*policy-profile*

  This policy profile describes the authorization to specify the TOKENTIMEOUT value. The Command Verifier policy profile does not include the new value for the TOKENTIMEOUT. This policy profile is also used when resetting the TOKENTIMEOUT value back to its default value (using the NOTOKENTIMEOUT keyword).

  Command Verifier supports the following access levels to the policy profile:

  No profile found
  This control is not implemented.

  NONE
  The terminal user is not authorized to specify or remove the TOKENTIMEOUT value for the policy. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to specify or remove the TOKENTIMEOUT value for the policy.

CONTROL
Same as UPDATE.

- C4R.MFADEF.MFPOLICY.ATTR.REUSE.*policy-profile*

  This policy profile describes the authorization to specify the token REUSE option. The Command Verifier policy profile is used for allowing and disallowing the reuse of MFA tokens.

  Command Verifier supports the following access levels to the policy profile:

  No profile found
  This control is not implemented.

  NONE
  The terminal user is not authorized to manage the REUSE option for the policy. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The terminal user is authorized to manage the REUSE option for the policy.

  CONTROL
  Same as UPDATE.

Policy profiles for STDATA segment management

Because of the sensitive nature of certain fields in the STDATA segment, your organization might need to maintain control over the STDATA profiles, beyond the controls already provided by RACF.

As stated previously, RACF command authorization allows verification only on the field name itself, and not on its value. Using `Field Level Access Checking`, it is possible to restrict setting the PRIVILEGED flag to certain users, if the terminal user does not have the System-SPECIAL attributes. Profiles in the **FIELD** class are not checked for System-SPECIAL users.

Additionally, some installations want to restrict the assignment of certain values for the USER and GROUP in the STDATA segment.

More controls are provided by zSecure Command Verifier for the STDATA segment. These controls are in addition to the RACF requirements like System-SPECIAL or UPDATE access to the applicable profiles in the FIELD class. For instance, the zSecure Command Verifier policy profiles can be used to prevent the accidental assignment of the PRIVILEGED attribute by RACF administrators with System-SPECIAL.

Table 47. Profiles used for verification of STDATA values. The entries in this table reflect the Class, Segment, and Field and the corresponding policy profiles.

Class	Field	Profile
STARTED	PRIVILEGED	C4R.STARTED.STDATA.ATTR.PRIVILEGED.*started-profile*
STARTED	TRUSTED	C4R.STARTED.STDATA.ATTR.TRUSTED.*started-profile*
STARTED	TRACE	C4R.STARTED.STDATA.ATTR.TRACE.*started-profile*
STARTED		C4R.STARTED.STDATA.=USER.*started-profile*
STARTED		C4R.STARTED.STDATA./USER.*started-profile*
STARTED	*userid*	C4R.STARTED.STDATA.USER.*userid.started-profile*
STARTED	NOUSER	C4R.STARTED.STDATA.USER.=NONE.*started-profile*
STARTED		C4R.STARTED.STDATA.=GROUP.*started-profile*
STARTED		C4R.STARTED.STDATA./GROUP.*started-profile*
STARTED	*group*	C4R.STARTED.STDATA.GROUP.*group.started-profile*

Table 47. Profiles used for verification of STDATA values. The entries in this table reflect the Class, Segment, and Field and the corresponding policy profiles. *(continued)*		
Class	**Field**	**Profile**
STARTED	NOGROUP	C4R.STARTED.STDATA.GROUP.=NONE.*started-profile*

The profiles in the preceding table describe mandatory and default values for both the USER and the GROUP. They also describe the policies that verify whether the values for the keywords, as entered by the terminal user, are acceptable.

- C4R.STARTED.STDATA.ATTR.PRIVILEGED.*started-profile*
- C4R.STARTED.STDATA.ATTR.TRUSTED.*started-profile*
- C4R.STARTED.STDATA.ATTR.TRACE.*started-profile*

  These profiles specify the authorization to set one of the attributes in the STDATA segment. The Privileged attribute results in passing most authorization checking. No installation exits are called, and no SMF records are written. It must be strictly controlled. The Trusted attribute is similar to the Privileged attribute, but SMF records can be written. The Trace attribute specifies that a record must be written to the console when the STARTED profile is used to assign an ID to a started task.

  No profile found
  Control not implemented. Only RACF authorization is used to control assignment of STDATA attributes.

  NONE
  The terminal user is not authorized to assign the attribute to this STARTED profile. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  The attribute setting is accepted. RACF authorization requirements can still cause failure of the command.

  CONTROL
  Same as UPDATE.
- C4R.STARTED.STDATA.=USER.*started-profile*
- C4R.STARTED.STDATA.=GROUP.*started-profile*

  These two Mandatory Value policy profiles can be used to assign a mandatory value for these STDATA fields. The mandatory value must be specified in the **APPLDATA** field of the policy profile. zSecure Command Verifier does not recognize any special values for the **APPLDATA**. This setting allows use of the value "=MEMBER" for the USER. This value is not substituted by zSecure Command Verifier but is used by RACF when the STARTED profile is used.

  These Mandatory Value policy profiles are only used when you add an STDATA segment either through the **RDEFINE** or the **RALTER** command. When you change existing STDATA segments, the Mandatory Value policy profiles are not used. The USER or GROUP obtained from this Mandatory Value profile is not subject to more user-or group-related policy profiles. The USER or GROUP value that is obtained from this Mandatory Value profile is not subject to more user- or group-related policy profiles.

  The qualifiers =USER and =GROUP in the policy profile cannot be covered by generic characters. They must be present in the exact form shown.

  No profile found
  The policy is not implemented. As a result, no mandatory value is enforced.

  NONE
  No action. No mandatory value is enforced.

  READ
  The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ

CONTROL
The policy profile is not active for the terminal user. No mandatory value is supplied. The value for the USER or GROUP as specified by the terminal user is used in the command.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Mandatory Value policy profiles, the profiles lead to the odd situation that access NONE has the same net result as access CONTROL.

- Currently, the following values for the **APPLDATA** are recognized:

BLANK
This setting is used to indicate that no explicit ID must be inserted.

id
Any other value is considered to be the *userid* or *group* that must be inserted in the STDATA segment. No verification is done to ensure that this value is a valid user ID or GROUP.

- `C4R.STARTED.STDATA./USER.`*started-profile*
- `C4R.STARTED.STDATA./GROUP.`*started-profile*

These two Default value profiles can be used to assign a Default value for these STDATA fields. The Default value must be specified in the **APPLDATA** field of the policy profile. zSecure Command Verifier does not recognize any special values for the **APPLDATA**. This setting allows use of the value `"=MEMBER"` for the USER. This value is not substituted by zSecure Command Verifier, but is used by RACF when the STARTED profile is used.

These Default value profiles are only used when you add an STDATA segment without a value for the USER or the GROUP, either through the **RDEFINE** or the **RALTER** command. When you change existing STDATA segments, the Default value policy profiles are not used. The USER or GROUP value that is obtained from this Default Value profile is not subject to more user- or group-related policy profiles.

The qualifiers `/USER` and `/GROUP` in the policy profile cannot be covered by generic characters. They must be present in the exact form shown.

No profile found
The policy is not implemented. No default value is provided.

NONE
No action. No default value is provided.

READ
The **APPLDATA** field is extracted and used for the command.

UPDATE
Same as READ

CONTROL
The policy profile is not active for the terminal user. No default value is provided.

Note: The access levels for this profile are not hierarchical. In general, zSecure Command Verifier policies do not apply to users that have CONTROL access or higher. Alternatively, access NONE indicates that the facility as described by the policy is not available to the terminal user. For the Default Value profiles, these profiles lead to the odd situation that access NONE has the same net result as access CONTROL.

Currently, the following values for the **APPLDATA** are recognized:

BLANK
This setting is used to indicate that no explicit ID must be inserted.

id
Any other value is considered to be the *userid* or *group* that must be inserted in the STDATA segment. No verification is done to ensure that this value is a valid user ID or GROUP.

- C4R.STARTED.STDATA.USER.*userid.started-profile*
- C4R.STARTED.STDATA.USER.=NONE.*started-profile*

  This policy profile specifies valid values for the *userid* for the *started-profile*. The special value =NONE is used when the terminal user specified the NOUSER keyword for the STDATA segment. This special value can be covered by a generic pattern. This setting allows treating the removal of the user assignment from the same policy profile as setting the user to a value. The following access levels are used.

  No profile found
  The policy is not implemented. The user specified value is accepted.

  NONE
  The specified USER is not allowed. The command is rejected.

  READ
  Same as NONE

  UPDATE
  The specified value for the USER is accepted

  CONTROL
  Same as UPDATE
- C4R.STARTED.STDATA.GROUP.*group.started-profile*
- C4R.STARTED.STDATA.GROUP.=NONE.*started-profile*

  This policy profile specifies valid values for the *group* for the *started-profile*. The special value =NONE is used when the terminal user specified the NOGROUP keyword for the STDATA segment. This special value can be covered by a generic pattern. This setting allows treating the removal of the group assignment from the same policy profile as setting the group to a value. The following access levels are used.

  No profile found
  The policy is not implemented. The user specified value is accepted.

  NONE
  The specified GROUP is not allowed. The command is rejected.

  READ
  Same as NONE

  UPDATE
  The specified value for the GROUP is accepted

  CONTROL
  Same as UPDATE

Other resource-related policy profiles

The dataset and general resource profiles have several fields that are not easily categorized. An appropriate term for some of them might be *attributes*. This section describes the remaining fields and attributes.

In the following table, the Keyword column shows the keywords and parameters that are specified on the RACF commands shown in the Command column. The Profile column shows the Command Verifier policy profile that applies to the combination of command, keywords, and parameters.

Table 48. Profiles used for Resource profile settings

Command	Keyword	Profile
ADDSD	*noset* *setonly*	C4R.DATASET.RACFIND.*set-value.profile*

Table 48. Profiles used for Resource profile settings (continued)

Command	Keyword	Profile
ADDSD RDEFINE	*generic* *model* *tape* *discrete*	C4R.*class.*TYPE.*type-value.profile*
ADDSD ALTDSD RDEFINE RALTER	*level*	C4R.*class.*LEVEL.*level.profile*
RDEFINE RALTER	APPLDATA	C4R.*class.*APPLDATA.*profile*
ADDSD ALTDSD RDEFINE RALTER	AUDIT(SUCCESS(*level*))	C4R.*class.*AUDIT.SUCCESS.*level.profile*
ADDSD ALTDSD RDEFINE RALTER	AUDIT(FAILURES(*level*))	C4R.*class.*AUDIT.FAIL.*level.profile*
ADDSD ALTDSD RDEFINE RALTER	ADD/DEL CATEGORY	C4R.*class.*CATEGORY.*category.profile*
ADDSD ALTDSD RDEFINE RALTER	(NO)DATA	C4R.*class.*INSTDATA.*profile*
ADDSD RDEFINE		C4R.*class.*/INSTDATA.*profile*
ADDSD ALTDSD	NO(ERASE)	C4R.DATASET.ATTR.ERASE.*profile*
ALTDSD RALTER	GLOBALAUDIT(SUCCESS(*level*))	C4R.*class.*GLOBALAUDIT.SUCCESS.*level.profile*
ALTDSD RALTER	GLOBALAUDIT(FAILURES(*level*))	C4R.*class.*GLOBALAUDIT.FAIL.*level.profile*
ADDSD ALTDSD RDEFINE RALTER	(NO)NOTIFY	C4R.*class.*NOTIFY.*notify-id.profile*
ADDSD ALTDSD RDEFINE RALTER	(NO)SECLABEL	C4R.*class.*SECLABEL.*seclabel.profile*
ADDSD ALTDSD RDEFINE RALTER	(NO)SECLEVEL	C4R.*class.*SECLEVEL.*seclevel.profile*

Table 48. Profiles used for Resource profile settings (continued)		
Command	**Keyword**	**Profile**
RDEFINE RALTER	SINGLEDSN	C4R.*class.*ATTR.SINGLEDSN.*profile*
RDEFINE RALTER	TIMEZONE	C4R.*class.*ATTR.TIMEZONE.*profile*
RDEFINE RALTER	TVTOC	C4R.*class.*ATTR.TVTOC.*profile*
ADDSD ALTDSD	RETPD	C4R.DATASET.RETPD.*profile*
ADDSD ALTDSD RDEFINE RALTER	NO(WARNING)	C4R.*class.*ATTR.WARNING.*profile*
RDEFINE RALTER	WHEN	C4R.*class.*ATTR.WHEN.*profile*

Other policy profiles and access level descriptions

This section describes the supported access levels and specific processing for the policy profiles that are available for other fields and attributes of dataset and general resource profiles.

- C4R.DATASET.RACFIND.*value.profile*

  This policy profile can be used to control the setting of the RACF indicator bit for data sets. The variable *value* can be specified as either NOSET or SETONLY. RACF also supports the explicit value SET, but that setting can default and ignored in many inappropriate situations like when you define a generic profile. The NOSET keyword specifies that for discrete profiles, the RACF indicator is not to be modified. The SETONLY keyword indicates that the TAPE data set is covered by a discrete profile, and that only the TVTOC must be updated. The following access rules apply:

  No profile found
  This control is not implemented.

  NONE
  Specifying the NOSET or SETONLY keyword is not allowed and causes the command to fail.

  READ
  Same as NONE.

  UPDATE
  Explicitly manipulating the RACF indicator flag is allowed.

  CONTROL
  Same as UPDATE

- C4R.*class.*TYPE.*type.profile*

  This policy profile controls the type of profile that is to be created. It is only applicable to the **ADDSD** and **RDEFINE** commands. The following are the possible values for *type*.

  - GENERIC
  - MODEL
  - TAPE
  - DISCRETE

  The first three values can be specified as keyword on most RACF commands. If these keywords are used, zSecure Command Verifier uses them as profile type for the policy profile. If none of these

keywords are present, zSecure Command Verifier examines the resource profile to determine whether a discrete or generic profile is to be created and set the value for the profile type accordingly.

Discrete data set profiles are discouraged because of the operational characteristics and cumbersome security administration. For some special situations, however, discrete profiles are still preferred. Some installations choose to automatically convert all discrete profiles to generic profiles. If operational procedures in the organization are not changed, it can have devastating effects on RACF performance.

Discrete profiles for general resources have none of the undesirable side effects that are associated with discrete data set profiles. In general, the choice between discrete and generic profiles must be based on the need for one or more profiles to protect multiple resources.

zSecure Command Verifier provides a policy that allows an administrator to make conscious decisions about using discrete or generic profiles. Using policy profiles, it is possible to globally disallow the creation of discrete data set profiles, while still allowing room for exceptions. This step is done by inclusion of the resource profile as part of the policy profile.

For many situations, you can use a double asterisk (**) to cover the profile part of the policy profile, as shown in these example policy profiles:

```
C4R.DATASET.TYPE.DISCRETE.**       UACC(NONE)
C4R.DATASET.TYPE.DISCRETE.SYS1.**  UACC(UPDATE)
C4R.FACILITY.TYPE.*.**             UACC(UPDATE)
```

The third type of profile in these examples is only needed if other policies for that resource class are implemented. The profile can be needed to create a more specific profile that allows creation of both discrete and generic profiles.

The available access levels to the policy profile are shown as follows.

No profile found
This control is not implemented. All users can create discrete, generic, and other types of profiles.

NONE
Creating a particular type of profile is not allowed and causes the command to fail. The type of profile can be specified by the terminal user, or can be determined automatically by zSecure Command Verifier.

READ
Same as NONE.

UPDATE
Creating the type of profile is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*LEVEL.*level.profile*

This policy profile can be used to control assignment of levels. The policy is not used for LEVEL(00) when creating a new resource profile. Depending on the access level, the policy profile might be used for other values, or when changing the value of an existing resource profile. The LEVEL in a resource profile must not be confused with the SECLEVEL. The LEVEL value is not used by RACF for any purpose.

No profile found
This control is not implemented.

NONE
Assignment of the LEVEL *level* to *profile* is not allowed and causes the command to fail.

READ
Same as NONE.

UPDATE
Assignment of this LEVEL to the resource profile is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*APPLDATA.*profile*

  This profile is used to control the authorization to change the application data of a resource profile. Normally the owner of the profile and people with (Group-)SPECIAL authorization are restricted. This profile implements further restrictions. The access levels that can be used for this profile are given as follows:

  No profile found
  This control is not implemented. All RACF authorized users can change the **APPLDATA** of resource profiles within their control.

  NONE
  Specifying installation data is not allowed and causes the command to fail. It applies both to **RDEFINE** and **RALTER**.

  READ
  Specifying **APPLDATA** on the **RDEFINE** command is allowed. Changing the value afterward by the **RALTER** command is not allowed.

  UPDATE
  Changing the **APPLDATA** is allowed.

  CONTROL
  The control is not implemented for this terminal user. No restrictions are imposed.

- **C4R.***class.***AUDIT.SUCCESS.***level.profile*
- **C4R.***class.***AUDIT.FAIL.***level.profile*

  These policy profiles can be used to control setting of SUCCESSFUL or FAILED access auditing as specified through the AUDIT keyword. Viewing and setting the AUDIT value is usually not restricted. Auditing through SMF can be specified for all access at or above the specified level. There are separate policy profiles for setting successful auditing and for setting failure auditing. The following access levels for the policy profile are supported:

  No profile found
  This control is not implemented.

  NONE
  Assignment of successful or failure auditing at the indicated access level is not allowed. The command is rejected.

  READ
  Same as NONE.

  UPDATE
  Assignment of successful or failure auditing at the indicated access level is allowed.

  CONTROL
  The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*CATEGORY.*category.profile*

  This profile can be used to control assignment of security categories. Normally, RACF administrators can assign their own CATEGORY to resource profiles in their scope. Security categories can be used as extra method of preventing access to resources. Users must have at least all the security categories that are assigned to the resource. The current profile allows control over the assignment and removal of a CATEGORY to a resource profile.

  No profile found
  This control is not implemented. Administrators who are assigned *category* can assign and remove this CATEGORY to resource profiles within their scope.

  NONE
  Assignment and removal of the CATEGORY *category* to *profile* is not allowed and causes the command to fail. It applies to all commands that can be used to set or modify the categories.

  READ
  System-SPECIAL users can assign and remove the *category* to this *profile*.

UPDATE
Administrators who are assigned *category* can assign this level to resource profiles within their scope. They also have the authority to remove *category*.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*INSTDATA.*profile*

This profile is used to control the authorization to change the installation data of a resource profile. Normally the owner of the profile and people with (group-)SPECIAL authorization are restricted. This profile implements further restrictions.

The INSTDATA policy profile can also include a reference to the format required for the installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate (set of) format specification policy profiles. Format specification policy profiles (or short format profiles) are named like:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

Multiple format profiles can be used to specify different parts of the installation data of the resource profile. For a complete description of the format profiles, see "Installation data field format specification" on page 193.

The following access levels can be used for this profile:

No profile found
This control is not implemented. All RACF authorized users can change the installation data of resource profiles within their control.

NONE
Specifying installation data is not allowed and causes the command to fail. It applies to all commands that can be used to set or modify INSTDATA.

READ
Specifying installation data on the **ADDSD** and **RDEFINE** commands is allowed. Changing the value afterward by the **ALTDSD** or **RALTER** commands is not allowed.

UPDATE
Changing the installation data is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

The optional value that is specified by **APPLDATA** is described as follows:

format
The name of the format that must be used for the installation data for the *profile*. The *format* name is used to locate the appropriate set of format profiles.

- C4R.*class*/INSTDATA.*profile*

This profile is used to control the authorization to set the default installation data of a resource profile. The /INSTDATA policy profile can also include a reference to the format that is required for the default installation data. The name of the format can be specified by the APPLDATA of the best fitting policy profile. The name of the format is used to determine the appropriate set of format specification policy profiles. Format specification policy profiles or short format profiles use names that are similar to the following name:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

Multiple format profiles can be used to specify different parts of the installation data of the RESOURCE profile. For a complete description of the format profiles, see "Installation data field format specification" on page 193. The following lists the access levels that can be used for the /INSTDATA profile.

No profile found
The control is not implemented. No default value is supplied.

NONE
No default value is supplied.

READ
The INSTDATA rules are extracted and used for the command.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied.

The value that is specified by APPLDATA is described as follows:

Format-Name
The name of the format that must be used for the installation data of the *userid* The *Format-Name* is used to locate the appropriate set of format profiles.

- C4R.*class*.ATTR.ERASE.*profile*
- C4R.*class*.ATTR.SINGLEDSN.*profile*
- C4R.*class*.ATTR.TIMEZONE.*profile*
- C4R.*class*.ATTR.TVTOC.*profile*

These four attribute-related policy rules have the same access rules. The possible UACC and ACL values are described as follows:

No profile found
This control is not implemented.

NONE
The terminal user is not authorized to specify either keyword on the **ADDSD** and **RDEFINE** commands. The no-attribute keyword is allowed on these commands.

READ
Same as NONE.

UPDATE
The terminal user is authorized to set and remove the attributes through the **ALTDSD**, **RALTER**, **ADDSD**, and **RDEFINE** commands. It allows regular maintenance of these attributes.

CONTROL
The control is not implemented for the terminal user. It allows regular maintenance of these attributes.

- C4R.*class*.GLOBALAUDIT.SUCCESS.*level.profile*
- C4R.*class*.GLOBALAUDIT.FAIL.*level.profile*

These policy profiles can be used to control setting of SUCCESSFUL or FAILED access auditing as specified through the GLOBALAUDIT keyword. Viewing and setting the GLOBALAUDIT value is usually restricted to people with the RACF AUDITOR attribute. Auditing through SMF can be specified for all access at or above the specified level. There are separate policy profiles for setting successful auditing and for setting failure auditing. The following access levels for the policy profile are supported:

No profile found
This control is not implemented.

NONE
Assignment of successful or failure auditing at the indicated access level is not allowed. The command is rejected.

READ
Same as NONE

UPDATE
Assignment of successful or failure auditing at the indicated access level is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*NOTIFY.*notify-id.profile*

This profile can be used to control setting of the Notify-ID. Normally, RACF administrators can specify which TSO user receives access violation messages. This policy profile provides control over the authorization to specify and select the Notify-ID.

No profile found
This control is not implemented. Administrators can specify and select any TSO user for notify messages.

NONE
Setting the Notify-ID is not allowed and causes the command to fail.

READ
Same as NONE.

UPDATE
Setting and selecting the Notify-ID is allowed

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*SECLABEL.*seclabel.profile*

Currently, there is only limited support for SECLABELs. It is possible to control assignment of a SECLABEL to a dataset or general resource. However, it is not possible to control removal of a SECLABEL.

- C4R.*class.*SECLEVEL.*seclevel.profile*

Currently, there is only limited support for SECLEVELs. It is possible to control assignment of a SECLEVEL to a dataset or general resource. However, it is not possible to control removal of a SECLEVEL.

- C4R.*class.*RETPD.*profile*

This policy profile controls the setting of the RETPD of tape data sets.

No profile found
This control is not implemented. The RETPD can be set and modified.

NONE
The retention period cannot be set or changed. It does not prevent setting of the RACF retention period that is based on the presence and value of JCL parameters.

READ
Same as NONE.

UPDATE
Specification and removal of tape data set RETPD is allowed.

CONTROL
The control is not implemented for this terminal user. No restrictions are imposed.

- C4R.*class.*ATTR.WARNING.*profile*

This profile can be used to control the setting of the WARNING or NOWARNING attribute of resource profiles. The WARNING attribute results in effectively disabling the resource access rules as defined by the profile. All users of the system can do anything to the resources protected by the resource profile. The only difference with UACC(ALTER) is the generation of more warning messages that the access is not granted if it was a regular profile.

No profile found
This control is not implemented.

NONE
The terminal user is not authorized to specify WARNING or NOWARNING on the **ADDSD**, **ALTDSD**, **RDEFINE**, and **RALTER** command. The NOWARNING keyword is allowed defaulted on the **ADDSD** and **RDEFINE** command.

READ
The terminal user is authorized to explicitly specify the NOWARNING attribute keyword on the **ALTDSD** and **RALTER** command. It allows removal of these attributes.

UPDATE
The terminal user is authorized to specify both keywords on all four relevant commands. It allows regular maintenance of these attributes.

CONTROL
Same as UPDATE.

- C4R.*class*.ATTR.WHEN.*profile*

  This single policy profile controls both the setting of the WHEN(DAYS) and the WHEN(TIME) specification for the TERMINALs. These two options control which days of the week, and which hours of the day, the TERMINAL can be used.

  No profile found
  This control is not implemented. WHEN(DAYS) and WHEN(TIME) can be specified.

  NONE
  Specification of LOGON restrictions is not allowed.

  READ
  Same as NONE.

  UPDATE
  Specification and removal of LOGON restrictions is allowed.

  CONTROL
  The control is not implemented for this terminal user. No restrictions are imposed.

Installation data field format specification

Use the guidelines in this topic to understand how to implement format rule policy profiles for the installation data field in RACF profiles. The format rule policy profiles are also used to set the default value for the installation data field in RACF profiles.

All RACF profiles provide a field that is reserved for installation usage. Many organizations use that installation data field for various purposes. RACF does not suggest any particular usage for this installation data field. Within the industry, there is also no real standard for usage of the installation data field. The only industry-wide consensus seems to exist for group profiles, where the installation data describes in text how the group is used like HLQ of particular application or access for certain jobs to certain applications.

RACF considers the installation data field as just a variable length text string. The only restriction for the RACF commands is the length of the field (maximum of 255 characters) and automatic uppercase translation for lowercase alphabetic characters. However, organizations might want to use certain positions in the installation data field for specific purposes. zSecure Command Verifier supports the enforcement of restrictions on the format of the installation data field (the INSTDATA policy profile). It can also set the default values to the installation data field (the /INSTDATA policy profile). For both cases, it is implemented by the definition of a named set of format rule policy profiles. These profiles are named like:

```
C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
```

The *format-name* is specified by the APPLDATA of the INSTDATA or /INSTDATA policy profiles:

INSTDATA policy profile
Controls the authorization to manage the installation data of the target profile in class *class*.

/INSTDATA policy profile
Specifies the default installation data of the target profile in class *class*.

For example, the INSTDATA profile for IBMUSER, owned by SYS1. might be:

```
C4R.USER.INSTDATA.SYS1.IBMUSER
       Appldata('SYS1FMT')
```

And the /INSTDATA profile for IBMUSER, owned by SYS1. might be:

```
C4R.USER./INSTDATA.SYS1.IBMUSER
       Appldata('SYS1FMT')
```

In both examples, the *format-name* is SYS1FMT. The format profiles for IBMUSER can then be:

```
C4R.USER.INSTDATA.=FMT.SYS1FMT.POS(start:end)
```

Multiple format profiles can be defined to specify the formats for different parts of the installation data field. The parts are indicated by the *start* and *end* positions. The format specifications for user IDs, GROUPs, and resource profiles must all be specified independently. In the absence of any profile for a certain resource class, it is also possible to use generic policy profiles for the INSTDATA format. The only generic profile that is supported is of the form:

```
C4R.*.INSTDATA.=FMT.format-name.POS(start:end)
```

The resource *class* can be replaced by an asterisk. It is not possible to specify partial generics for the resource class. If any format profile is defined for a particular resource class, all generic format profiles are ignored.

The zSecure Command Verifier policy profiles for the format are designed in a hierarchical way:

- The following sample is the structure of the top policy profile.

  ```
  C4R.class.INSTDATA.profile
  ```

  The top policy profile specifies the access that the user must have with the INSTDATA, and also specifies in the APPLDATA the format name (*format-name*) that is used to locate the FORMAT policies.
- The next level of policy profiles forms the set of FORMAT policies. You can specify up to 10 FORMAT profiles. These profiles are structured as:

  ```
  C4R.class.INSTDATA.=FMT.format-name.POS(start:end)
  ```

  These profiles specify the format rules and the access of the user to these format rules. The format rules are included as part of the APPLDATA of the FORMAT policy profiles.
- The format rules are specified as a list of comma-separated values without any additional blanks.

  ```
  'NB,ALPHA'
  'NB,LIST(EXPIRED,DELETE,STARTED)'
  ```

  “Format rules for value verification ” on page 197 describes the possible format rules for verification of the installation data. The following format rules are currently implemented format rules. The NC format rule does not apply to the default value policy.

  ```
   NB, NC, ALPHA, NUM, ALPHANUM, PICT, LIST, LISTX, =USERID, =GROUP
  ```

- The following format rules require more specification between parentheses:
  - PICT(*picture-string*)
  - LIST(*list-of-strings*)
  - LISTX(*list-of-strings*)

  These rules are documented in the following sections. Format rules that follow these three formats are ignored.

The profiles that are involved are summarized in the following two tables. The following sections describe the required access to these profiles and possible values for the *Format-Rules*.

Table 49. Profiles used for INSTDATA verification. The profiles in this table describe the authority to manage the INSTDATA, and are used to identify the INSTDATA format profile.

Class	Profile	Appldata
USER	C4R.USER.INSTDATA.*owner.userid*	*Format-Name*
GROUP	C4R.GROUP.INSTDATA.*owner.group*	*Format-Name*
DATASET	C4R.DATASET.INSTDATA.*hlq.rest-of-profile*	*Format-Name*
class	C4R.*class*.INSTDATA.*profile*	*Format-Name*

Table 50. Profiles used for default INSTDATA specification. The profiles in this table are used to identify the INSTDATA format profile.

Class	Profile	Appldata
USER	C4R.USER./INSTDATA.*owner.userid*	*Format-Name*
GROUP	C4R.GROUP./INSTDATA.*owner.group*	*Format-Name*
DATASET	C4R.DATASET./INSTDATA.*hlq.rest-of-profile*	*Format-Name*
class	C4R.*class*./INSTDATA.*profile*	*Format-Name*

The next table describes the policy profiles that are used for the format rules. These profiles are called format profiles. The APPLDATA of these profiles is used to specify the format rules.

Table 51. Profiles used for INSTDATA verification and to set default INSTDATA. The profiles in this table describe the INSTDATA format profile. The APPLDATA is used to describe the format rules.

Class	Profile	Appldata
class	C4R.*class*.INSTDATA.=FMT.*format-name.*POS(*start*:*end*)	*Format-Rule*
class	C4R.***.INSTDATA.=FMT.*format-name.*POS(*start*:*end*)	*Format-Rule*

INSTDATA policy profiles

Some zSecure Command Verifier policy profiles are used to control the authority to modify INSTDATA for a particular profile. These profiles can also be used to specify the Format-Name for the profile or type of profile.

The starting point is the profile that is used to control a change of the INSTDATA. This same profile is used to locate the applicable INSTDATA format descriptions.

- C4R.*class.*INSTDATA.*profile*

  Normally, authorization is already restricted to the owner of the profile and people with (group-)SPECIAL authorization. This profile implements further restrictions. The following access levels can be used for the INSTDATA profile.

  No profile found
  This control is not implemented. All RACF authorized users can change the installation data of users within their control.

  NONE
  Specifying installation data is not allowed and causes the command to fail. The **APPLDATA** field is not used, since the INSTDATA modification is not allowed.

READ
Specifying installation data on the add and define commands, like **ADDUSER** and **RDEFINE**, is allowed. Changing the value afterward through the alter commands, like **ALTUSER** and **RALTER**, is not allowed. The Format-Name that is specified by the **APPLDATA** is used to locate the applicable rules for the INSTDATA.

UPDATE
Changing the installation data is allowed. The Format-Name that is specified by the **APPLDATA** is used to locate the applicable rules for the INSTDATA.

CONTROL
The INSTDATA control is not implemented for this terminal user. However, the Format-Name that is specified by the **APPLDATA** is used to locate the applicable rules for the INSTDATA.

The optional value that is specified by **APPLDATA** is described as follows:

Format-Name
The name of the format that must be used for the installation data for the *profile*. The *Format-Name* is used to locate the appropriate set of format profiles that describe the Format-Rules.

/INSTDATA policy profiles

Some zSecure Command Verifier policy profiles are used to control the authority to set the default INSTDATA for a particular profile. These profiles can also be used to specify the Format-Name for the profile or type of profile.

The starting point is the profile that is used to set a default value of the INSTDATA. This same profile is used to locate the applicable INSTDATA format descriptions.

- C4R.*class.*/INSTDATA.*profile*

The following lists the access levels that can be used for the /INSTDATA profile:

No profile found
This control is not implemented. No default value is supplied.

NONE
No default value is supplied.

READ
The INSTDATA rules are extracted and the default value for INSTDATA is supplied.

UPDATE
Same as READ.

CONTROL
The control is not active for the terminal user. No default value is supplied.

The optional value that is specified by **APPLDATA** is described as follows:

Format-Name
The name of the format that must be used for the installation data for the *userid*. The *Format-Name* is used to locate the appropriate set of format profiles that describe the Format-Rules.

Format profiles

Use the guidelines in this topic to implement profiles that control the formatting of various parts of the INSTDATA field or to set a default value for the INSTDATA field.

You can define up to 10 different format profiles for each Format-Name to control 10 different parts of the INSTDATA of any profile. If the APPLDATA field of the preceding profile contains a nonblank value, zSecure Command Verifier tries to locate the format profiles. The sections of the INSTDATA are defined by the *start* and *end* variables in the format profiles.

The *start* to *end* range of the INSTDATA field as specified in one format profile can overlap with the field of other format profiles. In that case, the overlapping part must comply to both sets of format rules.

Format profiles can contain multiple format rules. These rules must be specified as a list and separated by commas. The following is an example of such a combined format rule.

```
NB,NC,ALPHA,LISTX(EXPIRED)
```

This sample indicates that this portion of the installation data field must be specified (NB) and cannot be changed (NC). If the user is authorized to bypass the NoChange requirement, the field must be all alphabetic characters, and cannot be specified as the value EXPIRED.

- C4R.*class*.INSTDATA.=FMT.*format-name.*POS(*start*:*end*)

  This profile is used to describe the format of a part of the INSTDATA. The part is specified by *start* and *end*. The *start* and *end* values must both be specified as three-digit numbers. The *end* value must be greater or equal than the *start* value, and must also be 255 or less. The *start* value must be 001 or greater. The access levels that are used for the format profile are given as follows.

  This profile is a discrete policy profile. It is possible to replace the *class* by a single generic asterisk. However, this generic profile is only used in complete absence of any matching format profile.

  The following access levels can be used for the =FMT policy profile that is used for the verification of the INSTDATA field.

  NONE
  The set of format rules applies to this terminal user. The new INSTDATA must comply with all format rules.

  READ
  The set of format rules applies to this terminal user. When present, the NB format rule does not apply. All other format rules must be followed.

  UPDATE
  The set of format rules applies to this terminal user. When present, the NB and NC format rules do not apply. All other format rules must be followed.

  CONTROL
  The set of format rules does not apply to this terminal user.

  The following access levels can be used for the =FMT policy profile that is used to set the default value of the INSTDATA field.

  NONE
  The set of format rules applies to this terminal user. The default value of INSTDATA field is supplied based on format rules.

  READ
  The set of format rules applies to this terminal user. When present, the NB format rule does not apply. All other format rules are used to provide the default value.

  UPDATE
  Same as READ.

  CONTROL
  The set of format rules does not apply to this terminal user.

  The **APPLDATA** describes the format rules that are applicable to the specific section of the INSTDATA.

  Set of format rules
  The format rules that describe allowable contents for this part of the INSTDATA. For a detailed description of the possible format rules, see the next section.

Format rules for value verification

Use these format rules to implement format profiles for the verification of the INSTDATA field.

Table 52 on page 198 summarizes the available format rules that are recognized by zSecure Command Verifier. If a specified format rule is not listed in this table, the entire set of format rules that are specified in the format profile is ignored. Processing continues with the next format profile. Multiple format rules

can be combined in the **APPLDATA** of one format profile. There is no test for logical consistency of the different format rules. For example, no specific error message is issued if the format rule specifies ALPHA, NUM, which would reject all possible characters.

Table 52. Format rules used for INSTDATA verification. The entries in this table contain the format rule and a description of the rule.

Format Rule	Description
NB	NonBlank. The specified part of the installation data field cannot consist of all blanks.
NC	NoChange. The current value of the specified part of the installation data cannot be modified.
ALPHA	Alphabetic. The specified part of the installation data field can contain only alphabetic characters or blanks.
NUM	Numerics. The specified part of the installation data field can contain only numeric characters or blanks.
ALPHANUM	Alphanumerics. The specified part of the installation data field can contain only alphabetic or numeric characters or blanks.
PICT(*picture-string*)	Picture format. The specified part of the installation data field must match the *picture-string* format. See "Picture string format (for value verification)" on page 198.
LIST(*list-of-strings*)	List of allowed values for the specified part of the installation data field. See "List of strings format (for value verification)" on page 199.
LISTX(*list-of-strings*)	List of values that are not allowed for the specified part of the installation data field. See "List of strings format (for value verification)" on page 199.
=USERID	Any valid RACF user ID.
=GROUP	Any valid RACF GROUP.

The following format rules require more specification between parentheses:

- PICT(*picture-string*)
- LIST(*list-of-strings*)
- LISTX(*list-of-strings*)

These rules are documented in the following sections.

Picture string format (for value verification)

Use these guidelines to specify PICTURE string characters in format rules for the INSTDATA field (for value verification).

For the PICT format rule, the *picture-string* specification is a single string of characters that describe possible values for each position in the INSTDATA. If the *picture-string* is shorter than the specified part of the installation data field, the remaining characters are not tested. If the *picture-string* is longer than the specified part of the installation data field, the extraneous characters are ignored.

The picture characters that can be specified are shown in Table 53 on page 199.

Note: Currently, you cannot use a right parenthesis as a literal character in the *picture-string*. If your installation data uses a right parenthesis, you can revert to using a period as a pattern character for that position.

Table 53. PICTURE string characters used for format rules. The entries in this table describe the supported PICTURE string characters.

Picture	Description
#	Numeric character (0-9)
@	Alphabetic character (A-Z)
*	Alphanumeric character (A-Z, 0-9)
$	Special character (@#$)
·	Anything. No verification that is done.
Other	Literal value. The installation data character must be identical to the picture-string character.

List of strings format (for value verification)

Use these guidelines to specify multiple strings in format rules for the INSTDATA field.

For LIST and LISTX, *list-of-strings* is a list of comma-separated strings. Each character, other than the comma and the parentheses, is significant. Each string is delimited by either a comma or the parenthesis at the beginning or end of the list. Each string can be at most 32 characters. If the string is shorter than the range specified by POS(*start:end*), the remaining characters in the INSTDATA must be blank. If the string is longer than the range specified by POS(*start:end*), the remainder is ignored. You do not need to explicitly specify an empty string as a possible LIST value because that is already controlled by the NB format rule.

The following example contains list-of-strings.

```
EXPIRED,DELETE,STARTED
```

Note in this example that the strings are of different length and contain no imbedded blanks.

You can include blanks in the strings, even at the beginning or end of a string. For example, the following string list contains blanks in the beginning, middle, and end of a string. The double quotation marks are not part of the syntax; they are included in the example only to indicate the inclusion of a trailing blank character:

```
  "EXPIRED USERID, TO BE DELETED, STARTED TASK "
```

This example assumes that POS(*start:end*) specifies a total of at least 14 characters in the format profile. Otherwise, it would not make sense to specify each string as exactly 14 characters.

One possible use of the LIST format rule is to implement special rules for a single position. Currently, zSecure Command Verifier does not provide a format rule to specify vowels or consonants. You can specify alphabetics, numerics, alphanumerics, and national characters, but you cannot specify that you need vowels. However, you can implement such a requirement by specifying a single POS and by including the format rule that lists all allowed characters. For example:

```
C4R.OPERCMDS.INSTDATA.=FMT.OPER.POS(001:001) APPLDATA('NB,LIST(A,E,I,O,U,Y)')
```

Using this same example, if you want to implement this format rule for two character positions, you must specify two different format policies:

```
C4R.OPERCMDS.INSTDATA.=FMT.OPER.POS(001:001) APPLDATA('NB,LIST(A,E,I,O,U,Y)')
C4R.OPERCMDS.INSTDATA.=FMT.OPER.POS(002:002) APPLDATA('NB,LIST(A,E,I,O,U,Y)')
```

You can also specify POS(001:002), but then you must list all 36 combinations of two vowels.

Because each format name can have at most 10 format policies, you can use this approach only for a limited number of character positions in the INSTDATA for the profile.

Format rules for the default value

Use these format rules to implement format profiles to set the default value of the INSTDATA field.

Table 54 on page 200 summarizes the available format rules that zSecure Command Verifier recognizes. If a specified format rule is not listed in this table, the entire set of format rules that are specified in the format profile is ignored. Processing continues with the next format profile. Multiple format rules can be combined in the APPLDATA of one format profile. There is no test for logical consistency of the different format rules.

Table 54. Format rules to set a default value for the INSTDATA field. The entries in this table contain the format rule and a description of the rule.

Format Rule	Description
NB	Each position specified in =FMT policy for installation data is filled with '.'
ALPHA	Each position specified in =FMT policy for installation data is filled with 'A'.
NUM	Each position specified in =FMT policy for installation data is filled with '0'.
ALPHANUM	Each position specified in =FMT policy for installation data is filled with 'Z'.
PICT(*picture-string*)	Picture format. The specified part of the installation data field must match the *picture-string* format. See "Picture string format (for default INSTDATA)" on page 200.
LIST(*list-of-strings*)	The specified part of the installation data field is filled with the first value from the list of strings. See "List of strings format (for default INSTDATA)" on page 201.
LISTX(*list-of-strings*)	The specified part of the installation data field is filled with the 'Default' keyword.
=USERID	The specified part of the installation data field is filled with a new userid (ADDUSER command) or with the ID of the terminal user.
=GROUP	The specified part of the installation data field is filled with the default group (ADDUSER command) or with the default group or the ID of the terminal user.

The following format rules require more specification between parentheses:

- PICT(*picture-string*)
- LIST(*list-of-strings*)

These rules are documented in the following sections.

Picture string format (for default INSTDATA)

Use these guidelines to specify PICTURE string characters in format rules for the default INSTDATA field.

For the PICT format rule, the *picture-string* specification is a single string of characters that describe default values for each position in the INSTDATA. If the *picture-string* is shorter than the specified part of the installation data field, the remaining characters are not set. If the *picture-string* is longer than the specified part of the installation data field, the extraneous characters are ignored.

Table 55. PICTURE string characters used for format rules for the default value. The entries in this table describe the supported PICTURE string characters.

Picture	Description
#	The installation data character is filled with '0'.
@	The installation data character is filled with 'A'.

Table 55. PICTURE string characters used for format rules for the default value. The entries in this table describe the supported PICTURE string characters. *(continued)*	
Picture	**Description**
*	The installation data character is filled with 'Z'.
$	The installation data character is filled with '$'.
.	The installation data character is filled with '.'.
Other Literal value	The installation data character replaced with a value that is identical to the picture-string character.

List of strings format (for default INSTDATA)

The first value of a list of strings is used as the default installation data field value.

For LIST, *list-of-strings* is a list of comma-separated strings. Each character, other than the comma and the parentheses, is significant. Each string is delimited by either a comma or the parenthesis at the beginning or end of the list. Each string can be at most 32 characters. If the string is shorter than the range specified by POS(*start:end*), the remaining characters in the INSTDATA are set to blank. If the string is longer than the range specified by POS(*start:end*), the remainder is ignored.

Policy profiles for USS segment management

Using RACF FIELD profiles, it is possible to control which administrators are authorized to maintain segment information. The way in which RACF implements this step can most accurately be described as a way to facilitate "product administrators".

Many organizations expressed a desire to allow Group Administrators to maintain their own users, across all products. RACF does not support such an implementation. “Profiles for controlling management of non-base segments” on page 52 describes how zSecure Command Verifier implements a facility to restrict "product administrators" to profiles within their RACF Group-SPECIAL scope, effectively creating the "across-all-products" Group Administrators.

However, there are certain fields, especially in the USS environment, that must remain restricted to central administrators. The most important one is the USS UID that is assigned to users by the OMVS segment. UID(0) is equivalent to SuperUser authority in the USS environment. You do not want your decentralized Administrators to be able to assign UID(0) to any of their users.

Similarly, a GID of a GROUP is used in the access verification process for USS files. It therefore requires similar controls.

Aside from the value of the UID or GID, you can also ensure that the specific UID or GID is assigned to a single user or group only. RACF allows sharing the same UID or GID through the use of the SHARED keyword. The RACF requirement for use of the SHARED keyword is either System-SPECIAL or READ access to the SHARED.IDS resource in the UNIXPRIV class. zSecure Command Verifier includes policies to control sharing a UID or GID, even for users with System-SPECIAL.

Table 56. Profiles used for verification of USS ID values. The entries in this table reflect the Class, Segment, and Field and the corresponding policy profiles.			
Class	**Segment**	**Field**	**Profile**
USER	*OMVS*	*UID*	C4R.USER.OMVS.UID.*uid.owner.userid*
USER	*OMVS*	*UID*	C4R.USER.OMVS.SHARED.*owner.userid*
USER	*OVM*	*UID*	C4R.USER.OVM.UID.*uid.owner.userid*
GROUP	*OMVS*	*GID*	C4R.GROUP.OMVS.GID.*gid.owner.group*
GROUP	*OMVS*	*GID*	C4R.GROUP.OMVS.SHARED.*owner.group*

Table 56. Profiles used for verification of USS ID values. The entries in this table reflect the Class, Segment, and Field and the corresponding policy profiles. *(continued)*

Class	Segment	Field	Profile
GROUP	*OVM*	*GID*	C4R.GROUP.OVM.GID.*gid.owner.group*

In the previous table, the values *uid* and *gid* in the policy profiles for the UID or GID must be specified as 10-digit numbers. Generics can be used as well. The following example shows the profile that protects the assignment of UID(0) to any user.

```
C4R.USER.OMVS.UID.0000000000.**
```

If you wanted to allow UID(0) to be assigned to people in systems support, you might use a profile similar to

```
C4R.USER.OMVS.UID.0000000000.SYSSUP.*
```

Specifying the 10-digit zeros by an asterisk, indicating that all UIDs are open to people in system support, does not work. RACF treats the numeric characters as more specific, and uses that profile instead of the asterisk profile. If you specified a second profile with a generic for the UID number, it would indicate that all *other* UIDs are open to people in systems support. The terminal user that does the actual assignment needs access to the policy profile. Also, RACF authorization, like System-SPECIAL or FIELD profiles, is required.

- C4R.USER.OMVS.UID.*uid.owner.userid*
- C4R.USER.OVM.UID.*uid.owner.userid*
- C4R.GROUP.OMVS.GID.*gid.owner.group*
- C4R.GROUP.OVM.GID.*gid.owner.group*

  These profiles specify the authorization to set a specific *uid* for *userid* owned by *owner*. The *UID* and *GID* must be specified as a 10 digit, zero padded, right-aligned number. It is also possible to use generics to allow or disallow management of certain ranges of *UID*s and *GID*s.

  No profile found
  Control not implemented. Only RACF authorization is used to control assignment of USS ID values.

  NONE
  The terminal user is not authorized to assign this UID/GID to the USERID or GROUP. This setting causes the command to fail.

  READ
  Same as NONE.

  UPDATE
  The value for the UID/GID is accepted. RACF authorization requirements can still cause failure of the command.

  CONTROL
  Same as UPDATE.

- C4R.USER.OMVS.SHARED.*owner.userid*

  This policy profile is used to control the use of the SHARED keyword when assigning a UID to a USERID. Assigning the same UID to multiple users is in general not advised because individual user control would be lost. To use the SHARED keyword, RACF requires either system special or, when defined, at least READ access to resource SHARED.IDS in the UNIXPRIV class. The following access rules for the policy profile apply:

  No profile found
  Control not implemented. All RACF authorized users can assign a SHARED UID to the target user ID.

NONE
The terminal user is not authorized to use the SHARED keyword when assigning a UID to the target user ID. This is independent of the specified UID that is actually being shared or not. The command is rejected.

READ
Same as NONE.

UPDATE
The terminal user is authorized to assign a SHARED UID to the target user ID. The terminal user still needs sufficient RACF authorization.

CONTROL
Same as UPDATE.

- C4R.GROUP.OMVS.SHARED.*owner.group*

This policy profile is used to control the use of the SHARED keyword when assigning a GID to a GOUP. Assigning the same GID to multiple groups is in general not advised because individual group control would be lost. To use the SHARED keyword, RACF requires either System-SPECIAL or, when defined, at least READ access to resource SHARED.IDS in the UNIXPRIV class. The following access rules for the policy profile apply:

No profile found
Control not implemented. All RACF authorized users can assign a SHARED GID to the target group.

NONE
The terminal user is not authorized to use the SHARED keyword when assigning a GID to the target group. This is independent of the specified GID that is actually being shared or not. The command is rejected.

READ
Same as NONE.

UPDATE
The terminal user is authorized to assign a SHARED GID to the target group. The terminal user still needs sufficient RACF authorization.

CONTROL
Same as UPDATE.

Notices

This information was developed for products and services offered in the U.S.A.

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

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

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

For license inquiries regarding double-byte (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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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 might 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:

IBM Corporation
2Z4A/101

11400 Burnet Road
Austin, TX 78758 U.S.A.

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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains 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. 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 IBM‘s application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation, in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on ibm.com/legal/copytrade.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware, the VMware logo, VMware Cloud Foundation, VMware Cloud Foundation Service, VMware vCenter Server, and VMware vSphere are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are registered trademarks of Hewlett Packard Enterprise, International Business Machines Corporation and Quantum Corporation in the United States and other countries.

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

Index

Special Characters

A

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Part Number:

(1P) P/N:

SC27-5648-07