Enterprise Dynamic Access Control (EDAC) Compliance with the American National Standards Institute (ANSI) Role Based Access Control (RBAC)

Prepared for
Commander, U.S. Pacific Fleet
Pearl Harbor, HI 96860

Prepared by
Richard Fernandez
SSC San Diego
675 Lehua Ave, Building 992
Pearl City, HI 96782
(808) 474-9270, fax (808) 471-5837
richard.r.fernandez@navy.mil

Revisions

Publication Debut	May 1, 2005	Richard Fernandez

Bibliography

Ravi Sandhu, David Ferraiolo, Rick Kuhn. American National Standard for Information Technoloby – Role Based Access Control, ANSI INCITS 359-2004, February 3, 2004.

Richard Fernandez. Enterprise Dynamic Access Control (EDAC), May 1, 2005.

Acknowledgement

The author wishes to acknowledge assistance from Rick Kuhn for his contributions and collaboration to this paper. Mr. Kuhn is a computer scientist in the Computer Security Division of the National Institute of Standards and Technology (NIST). Mr. Kuhn co-authored the NIST Role Based Access Control, which is an ANSI standard.

The author also wishes to acknowledge the following COMPACFLT Secured Enterprise Access Tool (SEAT) architects: Wallace Fukumae, Tuan Huynh , Wilfredo Alvarez, Ryan Kanno, Dean Tanabe.

Table of Contents

This Page Intentionally Left Blank

Chapter 1 EDAC and ANSI INCITS 359-2004 RBAC Conformance

General

Access control is the process that evaluates resource access. Resources can represent software applications, web services and even facility access. An effective access control model should be capable of evaluating resource access based on user characteristics and environmentals. Access control lists (ACL) and Groups are predominantly used as mechanisms of access control. Unfortunately static listings such as ACLs and Groups are not scalable and present significant security shortfalls. The National institute of Standards and Technology (NIST) has recognized this problem and published a Role-Based Access Control (RBAC) standard. On February 3, 2004 the American National Standard for Information Technology Role-Based Access Control (RBAC) INCITS 359-2004 became effective.

The Enterprise Dynamic Access Control (EDAC) represents an access control model, which was designed and developed for COMPACFLT. Appendix A and B offers an overview and case study for this model. The reader is encouraged to read these appendices to develop a background understanding.

The purpose of this paper is to explain the conformance of the EDAC with the American National Standard for Information Technology Role-Based Access Control (RBAC) ANSI INCITS 359-2004. Throughout this paper excerpts from the ANSI INCITS 359-2004 are enclosed in quotation marks followed by a parenthetical reference. References to this standard can be found in the bibliography section at the beginning of this paper.

Contents

This paper will be broken down into the following chapters:

Chapter 2 ANSI INCITS 359-2004 RBAC Reference Model – defines and compares the EDAC and the ANSI INCITS 359-2004 model element sets and relations.

Chapter 3 ANSI INCITS 359-2004 RBAC System and Administrative Functional Specifications – examines how the EDAC complies with the ANSI INCITS 359-2004:

Administrative Commands
Supporting System functions
Review Functions
Advanced Review Functions

The EDAC overview and case study can be referenced at the NIST RBAC Standards Roadmap.

Conformance Criteria and Objectives

The ANSI INCITS 359-2004 RBAC reference model is defined by the following model components:

Core RBAC
Hierarchical RBAC
Static Separation of Duty Relations
Dynamic Separation of Duty Relations

According to the ANSI INCITS 359-2004, "The Core RBAC defines a minimum collection of RBAC elements, element sets, and relations in order to completely achieve a Role-Based Access Control system" (2). Therefore, in chapter 2 the EDAC is compared only with the "Core RBAC" model found in section 5.1 of the ANSI INCITS 359-2004).

In addition the ANSI INCITS 359-2004 mentions that, "To conform to this standard, an RBAC system shall comply with all of the core set of RBAC functional specifications in 6.1. Conformance of an RBAC system to any other functional specifications for a particular component and feature option, found in 6.2 through 6.4, is optional and dependent upon the functional requirements of a particular application" (1). Chapter 3 will only focus on the RBAC functional specifications in section 6.1 of the ANSI INCITS 359-2004.

Chapter 2

Chapter 2 ANSI INCITS 359-2004 RBAC Reference Model

Core RBAC

According to the ANSI INCITS 359-2004 the "Core RBAC" consists of five basic data sets called:

USERS
ROLES
OBS
OPS
PRMS

In the ANSI INCITS 359-2004 the "Core RBAC" is illustrated in figure 1.

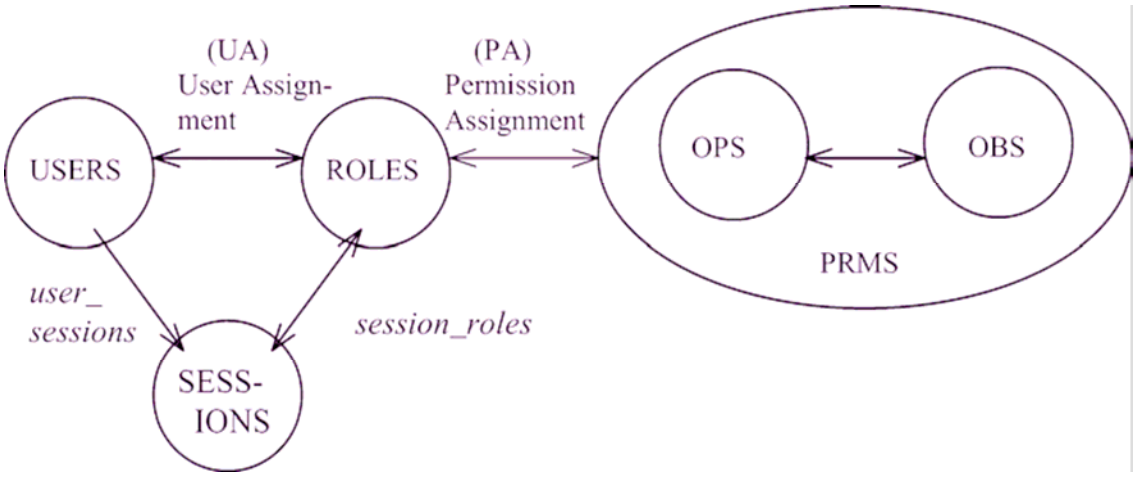


Figure 1

In addition to a discussion of the five basic ANSI INCITS 359-2004 elements this chapter will also compare the ANSI INCITS 359-2004: User Assignments (UA), Permission Assignments (PA) and SESSION with the EDAC.

Users

According to the ANSI INCITS 359-2004, "A *user* is defined as a human being. Although the concept of a user can be extended to include machines, networks, or intelligent autonomous agents, the definition is limited to a person in this document for simplicity reasons" (3).

The EDAC refers to users as objects, which qualifies anything (human or non-human) requiring access to a resource.

Roles

According to the ANSI INCITS 359-2004, "A *role* is a job function within the context of an organization with some associated semantics regarding the authority and responsibility conferred on the user assigned to the role" (3).

In the EDAC a role represents a subset of user characteristics as defined by the context of an organization. Figure 2 presents a list of user characteristics:

What branch: **organization**
What paygrade: **wage grade**
What vocation: **job function**
Where client works at: **location**
What security credentials: **clearance**

Figure 2

A compilation of user characteristics is placed into a user profile that is evaluated to determine resource access. Figure 3 illustrates an example of a user profile:

Categories	**COMPACFLT**
Organization:	CPF N65
Clearance:	Secret
Service:	DoD
Paygrade:	GS12
Job Description:	Program Manager

Figure 3

Note: not all user characteristics in a user profile are required or even sufficient to define a role. A user can fulfill the requirements of a role by more than one user profile. Figure 4 illustrates how a user could have multiple user profiles. Depending on the selected profile a user's characteristics could match an ANSI INCITS 359-2004 "role" and therefore, be granted access to a resource.

Categories	**COMPACFLT**	**USNR**
Organization:	CPF N65	Naval Intel
Clearance:	Secret	Top Secret
Service:	DoD	DoNR
Paygrade:	GS12	O2
Job description:	Program Manager	Intelligence

Figure 4

In some instances a user could be prevented from selecting a user profile. Maybe the selection of a user profile is automated by the condition of an environmental. For example, during weekdays (Monday – Friday) a COMPACFLT profile is submitted while on weekends (Saturdays and Sundays) a USNR profile is submitted for role evaluation. User profile submissions could also be based on work location. For example, a user at COMPACFLT could only submit a COMPACFLT profile while the same user at a Naval Reserve installation could only be permitted to submit a USNR profile.

The EDAC's "Object Profile Manager Service (OPMS)" performs the compilation of user characteristics into user profile(s) for selection by the user.

Objects

According to the ANSI INCITS 359-2004, "…an *object* is an entity that contains or receives information. For a system that implements RBAC, the objects can represent information containers (e.g., files, directories, in an operating system, and/or columns, rows, tables, and views within a database management system) or objects can represent exhaustible system resources, such as printers, disk space, and CPU cycles. The set of objects covered by RBAC includes all of the objects listed in the permissions that are assigned to roles" (3 - 4).

A list of "objects" within a resource is shown in figure 5.

Objects
Log files
Config files
Test files
Operation files

Figure 5

Therefore, OBS consists of 4 elements:

OBS ∈ { Log files,
Config files,
Test files,
Operation files }

Operations

According to the ANSI INCITS 359-2004, "An *operation* is an executable image of a program, which upon invocation executes some function for the user" (2).

A list of "operations" is shown in figure 6 as an example:

Operations
Read: r
Write: w
Execute: x
Audit: a

Figure 6

Therefore, OPS consists of 4 elements:

OPS ∈ { r, w, x, a }

Permissions

According to the ANSI INCITS 359-2004, *"Permission* is an approval to perform an operation on one or more RBAC protected objects" (2).

Continuing with our example, figure 7 shows how certain "operations" can be performed on respective "objects" within a resource.

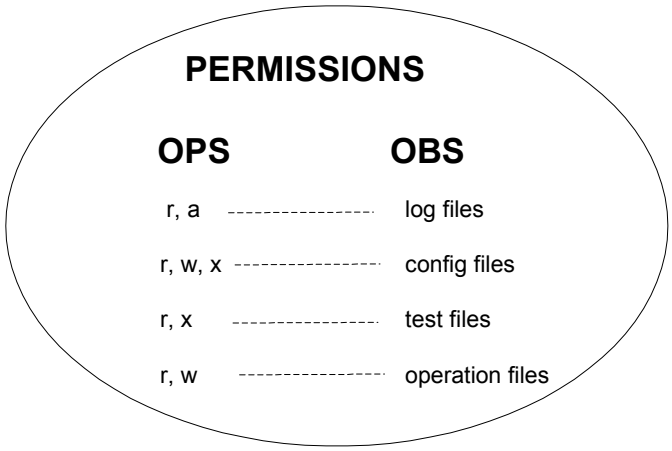


Figure 7

Permission Assignments (PA)

Permission assignments assign certain roles to specific permission(s) within a resource. Some roles can be assigned to more than one "operation" within a resource. Figure 8 illustrates offers an example.

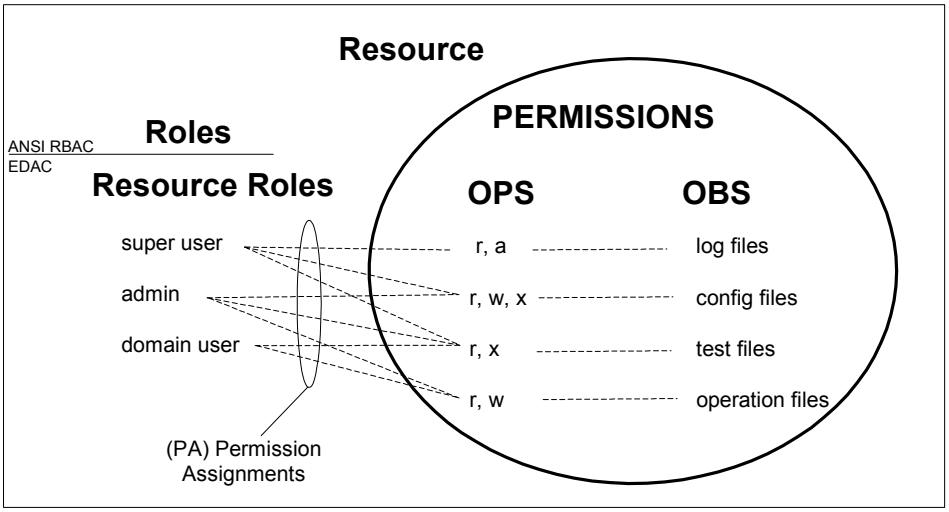


Figure 8

In this example the super user "role" is assigned to different operation sets:

read and audit "operations" on log file "objects"
read, write and execute "operations" on config file "objects"
read, execute "operations" on test file "objects"

The EDAC refers to the ANSI INCITS 359-2004 "roles" as "resource roles" or "Access Control Roles" (ACR). Essentially ANSI INCITS 359-2004 "roles"

mean the same as EDAC “resource roles” and “ACRs”. The purpose for the EDAC distinction is to define a role’s location. The EDAC defines the “ACR” as extensions of “resource roles” within the access control system. “ACRs” have a one-to-one relation with “resource roles”. Refer to figure 9:

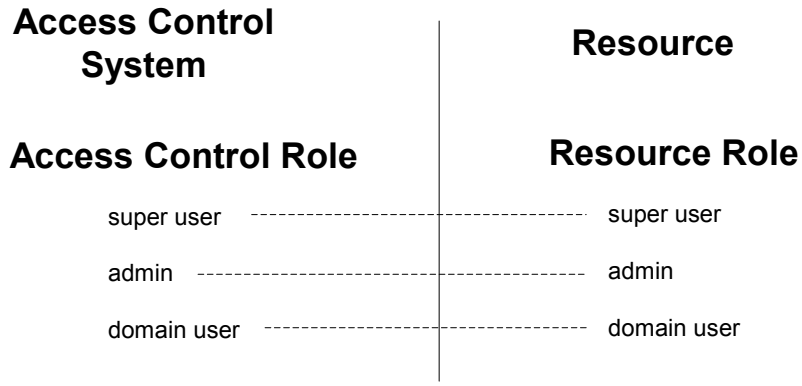


Figure 9

Sessions

According to the ANSI INCITS 359-2004, “In addition, the core RBAC model includes a set of sessions (SESSIONS) where each session is a mapping between a user and an activated subset of roles that are assigned to the user” (3).

In the EDAC, an ANSI INCITS 359-2004 “session” is created by a “resource profile”. The EDAC “reference conditions” in a “resource profile” are the minimum criteria to establish an ANSI INCITS 359-2004 “session”. A mapping between matched “reference conditions” within a “resource profile” and “reference inputs” (or user characteristics) within a “user profile” creates what the ANSI INCITS 359-2004 refers to as an “activated subset of roles“ (4). Refer to figure 10.

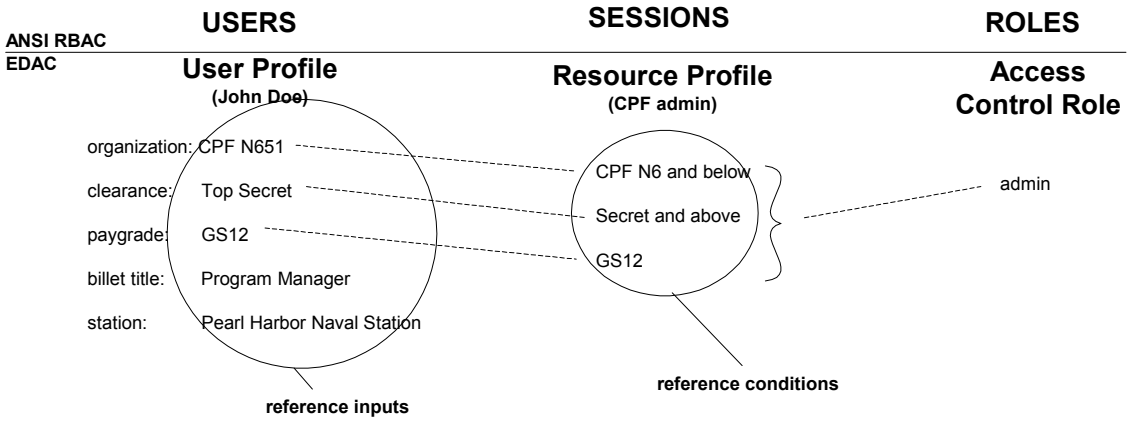


Figure 10

According to the ANSI INCITS 359-2004 "The function *session_roles* gives us the roles activated by the session and the function *session_users* gives us the user that is associated with a session" (4). In this example, the "user_session" represents the relationship between user: John Doe and session: CPF admin session. The "session_roles" represents the relationship between session: CPF admin and role: admin. Refer to figure 11.

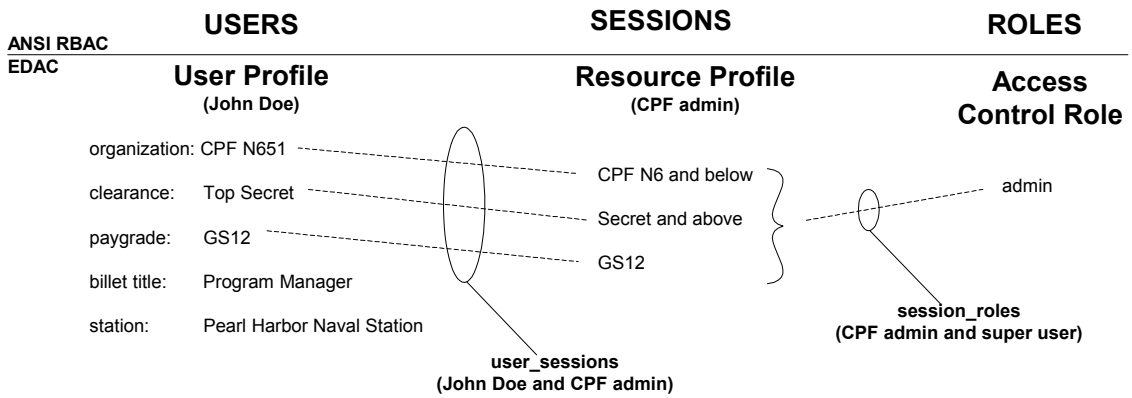
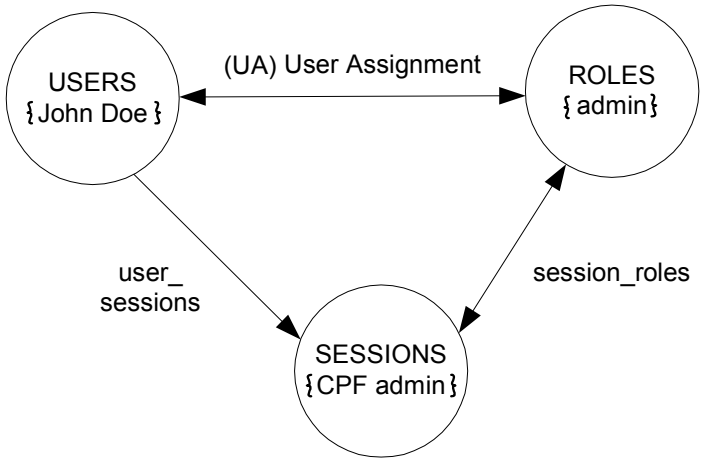


Figure 11

In the ANSI INCITS 359-2004 a "user assignment (UA)" is created between a "user" and a "role" upon the establishment of a "session". Referring to the previous example in figure 11, the CPF ADMIN session has established a John Doe/admin "user assignment (UA)". Refer to figure 12.



Where:

USERS ∈ { John Doe }
SESSIONS ∈ {CPF admin }
ROLES ∈ { admin }

user_session_1 = John Doe/CPF admin
session_roles_1 = CPF admin/admin
user assignment_1 = John Doe/admin

Figure 12

Multiple Sessions

This section discusses how multiple "sessions" establish multiple "user assignments". According to the ANSI INCITS 359-2004, "… a user can be assigned to one or more roles, and a role can be assigned to one or more users" (4). In the following example a "user" will be assigned two "roles" via the establishment of respective "sessions". Figure 13 illustrates how a "user" is assigned to "roles": admin and domain user.

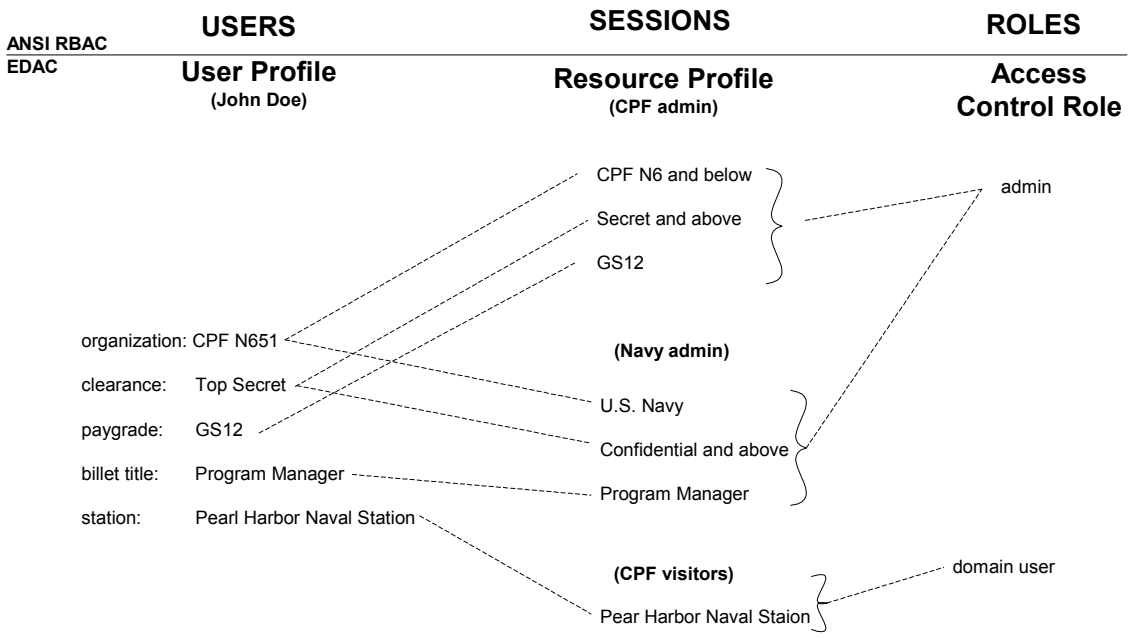
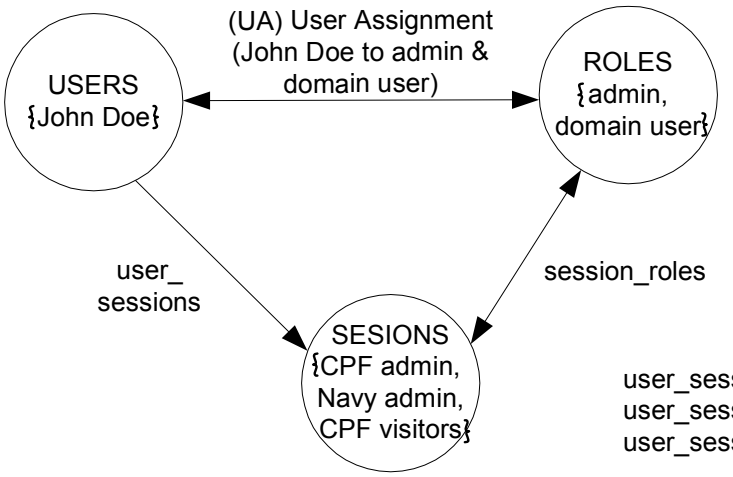


Figure 13

Figure 14 illustrates a summary of the "sessions" within the "Core RBAC" model.



Where:

USERS ∈ { John Doe }
SESSIONS ∈ {CPF admin, Navy admin, CPF visitors }
ROLES ∈ { admin, domain user }

user_sessions_1 = John Doe/CPF admin
user_sessions_2 = John Doe/Navy admin
user_sessions_3 = John Doe/CPF visitors

session_roles_1 = CPF admin/admin
session_roles_2 = Navy admin/admin
session_roles_3 = CPF visitors/domain user

user assignment_1 = John Doe/admin
user assignment_2 = John Doe/domain user

Figure 14

Summary

This chapter has demonstrated how the EDAC complies with the ANSI INCITS 359-2004 “Core RBAC” reference model and specifications. This chapter discussed how an EDAC “resource profile” can activate a subset of user characteristics within a user profile and thus establish an ANSI INCITS 359-2004 “user assignment(s)” between “user” and “role(s)”. This chapter also discussed how an EDAC “ACR” (or ANSI INCITS 359-2004 “role(s)”) is associated with a specific resource “operation(s)” via a “permission assignment”. This chapter also covered how the ANSI INCITS 359-2004 resource “operation(s)” can perform functions on respective “object(s)”.

Chapter 3

Chapter 3 ANSI INCITS 359-2004 RBAC System and Administrative Functional Specifications

General

Section 6.1 of the ANSI INCITS 359-2004 discusses administrative and maintenance of the "Core RBAC" elements. This section is divided in the following four parts:

Administrative Commands
Supporting System Functions
Review Functions
Advanced Review Functions

Compliance to these functions is a requirement for ANSI INCITS 359-2004 conformance.

Each administrative function will be followed by a quote from the ANSI INCITS 359-2004 with an associated parenthetical reference. An explanation will follow on how the EDAC complies with this particular feature.

Administrative Commands for Core RBAC

AddUser

> This command creates a new RBAC user. The command is valid only if the new user is not already a member of the *USERS* data set. The *USER* data set is updated. The new user does not own any session at the time of its creation. (11)

Here "*USERS* data set" represents a "customer personnel database (CPD)" that interfaces with the EDAC. The "CPD" is managed and maintained by the customer and contains user records required to create a "user profile". "Users" can be created, deleted and edited in the "CPD".

DeleteUser

> This command deletes an existing user from the RBAC database. The command is valid if and only if the user to be deleted is a member of the *USERS* data set. The *USERS* and *UA* data sets and the *assigned_users* function are updated. It is an implementation decision how to proceed with the sessions owned by the user to be deleted. The RBAC system could wait for such a session to terminate normally, or it could force its termination. (11)

Here "*USERS* data set" represents a "customer personnel database (CPD)" that interfaces with the EDAC. The "CPD" is managed and maintained by the customer and contains user records required to create a "user profile". "Users" can be created, deleted and edited in the "CPD".

AddRole

> This command creates a new role. The command is valid if and only if the new role is not already a member of the *ROLES* data set. The ROLES data set and the functions *assigned_users* and *assigned_permissions* are updated. Initially, no user or permission is assigned to the new role. (12)

A resource manager can manage "roles" by adding or deleting "ACRs" in the EDAC's "Condition Manager Service (CMS)". Refer to the section on "Permission Assignments" in chapter 2 of this document.

DeleteRole

> This command deletes an existing role from the RBAC database. The command is valid if and only if the role to be deleted is a member of the *ROLES* data set. It is an implementation decision how to proceed with the sessions in which the role to be deleted is active. The RBAC system could wait for such a session to terminate normally, it could force the termination of that session, or it could delete the role from that session while allowing the session to continue. (12)

A resource manager can manage "roles" by adding or deleting "ACRs" in the EDAC's "Condition Manager Service (CMS)". Refer to the section on "Permission Assignments" in chapter 2 of this document.

AssignUser

> This command assigns a user to a role. The command is valid if and only if the user is a member of the *USERS* data set, the role is a member of the *ROLES* data set, and the user is not already assigned to the role. The

data set UA and the function *assigned_users* are updated to reflect the assignment. (12)

According to the ANSI INCITS 359-2004 "user assignments (UA)" to a "role" is established upon a successful "session". In the EDAC a "user" can be assigned to a "role" whenever a "user's profile" matches a "resource profile". A "user" can also be de-assigned to a "role" if the "user profile" and/or the "resource profile" is removed or edited.

DeassignUser

This command deletes the assignment of the user *user* to the role *role*. The command is valid if and only if the user is a member of the *USERS* data set, the role is a member of the *ROLES* data set, and the user is assigned to the role. It is an implementation decision how to proceed with the sessions in which the session user is *user* and one of his/her active roles is *role*. The RBAC system could wait for such a session to terminate normally, could force its termination, or could inactivate the role. (13)

According to the ANSI INCITS 359-2004 "user assignments (UA)" to a "role" is established upon a successful "session". In the EDAC a "user" can be assigned to a "role" whenever a "user's profile" matches a "resource profile". A "user" can also be de-assigned to a "role" if the "user profile" and/or the "resource profile" is removed or edited.

GrantPermission

This command grants a role the permission to perform an operation on an object to a role. The command may be implemented as granting permissions to a group corresponding to that role, i.e., setting the access control list of the object involved. (13)

The EDAC allows resource managers to grant and revoke access to specific "permissions" on a resource. Assigning or de-assigning "ACR's" to specific "resource roles" (within a resource) can affect if a "role" will have access to certain resource "permissions". Refer to the section on Permission Assignments in chapter 2 of this document.

RevokePermission

This command revokes the permission to perform an operation on an object from the set of permissions assigned to a role. The command may be implemented as revoking permissions from a group corresponding to that role, i.e., setting the access control list of the object involved. (13)

The EDAC allows resource managers to grant and revoke access to specific "permissions" on a resource. Assigning or de-assigning "ACR's" to specific "resource roles" (within a resource) can affect if a "role" will have access to certain resource "permissions". Refer to the section on Permission Assignments in chapter 2 of this document.

Supporting System Functions for Core RBAC

This section focuses on managing ANSI INCITS 359-2004 sessions. Instead of presenting a case under each Supporting System Functions a summary of EDAC session management is offered at the beginning of this section. The summary satisfies all of the Supporting System Functions.

The EDAC "resource profile" is the catalyst that creates a ANSI INCITS 359-2004 "session". The EDAC Rules Engine Service (RES) establishes the ANSI INCITS 359-2004 "user_sessions", which associates a user to a "session" by determining if a subset of "reference inputs" within a "user profile" matches the "reference conditions" within a "resource profile". The match set of references between "user" and "resource profiles" represents the ANSI INCITS 359-2004 "active role set".

The resource manager establishes the ANSI INCITS 359-2004 "session_roles", which associates a "session" to a "role" by assigning a "resource profile" to an "ACR". Refer to the Sessions and Multiple Sessions section in chapter 2 of this document.

In the EDAC a "session" can be identified by the name of the "resource profile" that matches a "user profile" and each "session" is identified with a "user". The EDAC offers the capability to add and drop "sessions" by reference changes in a "user profile" or "resource profile".

An EDAC logging system can record access control activities. The following EDAC log events pertain to the "Core RBAC":

> Disabled session – displays which "resource profile" have been disabled. Active session – displays which session furnishes a user with a role to access a resource "permission". For an active "session" the EDAC logs associated user, resource profile and role.

CreateSession(user, session)

> This function creates a new session with a given user as owner and an active role set. The function is valid if and only if:
>
> - the user is a member of the *USERS* data set, and

- the active role set is a subset of the roles assigned to that user. In a RBAC
implementation, the session's active roles might actually be the groups that represent those roles. (14)

DeleteSession(user, session)

This function deletes a given session with a given owner user. The function is valid if and only if the session identifier is a member of the *SESSIONS* data set, the user is a member of the *USERS* data set, and the session is owned by the given user. A particular user profile such as COMPACFLT or SPAWAR profile could have numerous members. (14)

AddActiveRole

This function adds a role as an active role of a session whose owner is a given user. The
function is valid if and only if:
- the user is a member of the *USERS* data set, and
- the role is a member of the *ROLES* data set, and
- the session identifier is a member of the *SESSIONS* data set, and
- the role is assigned to the user, and
- the session is owned by that user.

In an implementation, the new active role might be a group that corresponds to that role. (14-15)

DropActiveRole

This function deletes a role from the active role set of a session owned by a given user. The function is valid if and only if the user is a member of the *USERS* data set, the session identifier is a member of the *SESSIONS* data set, the session is owned by the user, and the role is an active role of that session. (15)

CheckAccess

This function returns a Boolean value meaning whether the subject of a given session is allowed or not to perform a given operation on a given object. The function is valid if and only if the session identifier is a member of the *SESSIONS* data set, the object is a member of the *OBJS* data set, and the operation is a member of the *OPS* data set. The session's subject has the permission to perform the operation on that object if and only if that permission is assigned to (at least) one of the session's active roles. An

implementation might use the groups that correspond to the subject's active roles and their permissions as registered in the object's access control list. (15)

Review Functions for Core RBAC

The EDAC logging system can be audited in order to review "Core RBAC" functions. An active session log can determine assign "user" and assign "role" associated with a particular "session".

AssignedUsers

This function returns the set of users assigned to a given role. The function is valid if and only if the role is a member of the *ROLES* data set. (15)

AssignedRoles

This function returns the set of roles assigned to a given user. The function is valid if and only if the user is a member of the *USERS* data set. (16)

Advanced Review Functions for Core RBAC

The EDAC logging system can be audited in order to review the following advanced review functions of the Core RBAC.

RolePermissions

This function returns the set of permissions (*op*, *obj*) granted to a given role. The function is valid if and only if the role is a member of the *ROLES* data set. (16)

UserPermissions

This function returns the permissions a given user gets through his/her assigned roles. The function is valid if and only if the user is a member of the *USERS* data set. (16)

SessionRoles

This function returns the active roles associated with a session. The function is valid if and only if the session identifier is a member of the *SESSIONS* data set. (16)

SessionPermissions

This function returns the permissions of the session *session*, i.e., the permissions assigned to its active roles. The function is valid if and only if the session identifier is a member of the *SESSIONS* data set. (16)

RoleOperationsOnObject

This function returns the set of operations a given role is permitted to perform on a given object. The function is valid only if the role is a member of the *ROLES* data set, and the object is a member of the *OBJS* data set. (17)

UserOperationsOnObject

This function returns the set of operations a given user is permitted to perform on a given object, obtained either directly or through his/her assigned roles. The function is valid if and only if the user is a member of the *USERS* data set and the object is a member of the *OBJS* data set. (17)