

DRAFT

**CS1/06-0237 Updated 2<sup>nd</sup> draft Requirements for the  
Implementation of Role Based Access Control (RBAC)**

**Requirements for the Implementation of Role Based  
Access Control (RBAC)**

INCITS Project 1794  
1st draft January 31, 2006  
2<sup>nd</sup> draft August 14, 2006  
Updated 2<sup>nd</sup> draft September 2006

Editor: Ed Coyne

Co-editor: Richard Kissel

DRAFT

# DRAFT

## Contents

Introduction.....	2
1 Scope.....	2
2 Conformance.....	2
3 Normative References.....	2
4 Terms and Definitions.....	3
5 Symbols and Abbreviated Terms.....	3
6 Requirements .....	3
6.1 Combinations of Components.....	3
6.2 Interoperability.....	4
6.2.1 RBAC Interaction Functions for Interoperability .....	4
6.2.2 Data Requirements for Interoperability .....	6
6.2.3 Interface Requirements for Interoperability.....	7
6.3 User Interfaces .....	7
6.4 Audit .....	7
6.4.1 Audit Trail Content.....	7
6.4.2 Audit Recording.....	7
6.4.3 Audit Compliance .....	8
7 Normative Annexes .....	8
8 Informative Annexes.....	8
8.1 Discussion on Two Types of Roles: Basic and Functional.....	8
8.1.1 Purpose.....	8
8.1.2 Discussion.....	8
8.1.3 Basic Roles vs Organizational Roles .....	9
8.1.4 Roles in ASTM Healthcare Policy and Standard Guide.....	9
8.1.5 Conclusion .....	11
Appendix A: Rationale .....	11

# DRAFT

## 1 Introduction

This standard provides implementation requirements for Role Based Access Control (RBAC) systems, which use RBAC components defined in INCITS 359-2004. This standard is intended for (1) software engineers and product development managers who design products incorporating access control features; and (2) managers and procurement officials who seek to acquire computer security products with features that provide access control capabilities in accordance with commonly known and understood terminology and functional specifications. The implementation requirements in this standard are intended to ensure the interchange of RBAC data (e.g., roles, permissions, users) and promote functional interoperability among RBAC services and applications.

## 1 Scope

The INCITS 359-2004 section on System and Administrative Functional Specification specifies the features that are required of an RBAC system. These features fall into three categories, administrative operations, administrative reviews, and system level functionality.

This standard specifies the packaging of features through the selection of functional components and feature options within a component, beginning with a core set of RBAC features that must be included in all packages. Other components that may be selected in arriving at a relevant package of features pertain to role hierarchies, static constraints (Static Separation of Duty), and dynamic constraints (Dynamic Separation of Duty).

This standard also specifies formats and protocols for the interchange of RBAC data (e.g., roles, permissions, users) and for functional interoperability among RBAC services and applications.

This standard does not address role semantics or role engineering processes.

## 2 Conformance

To conform to this standard, an RBAC system shall comply with all of the requirements in clause 6.

### **Completeness**

### **Correctness**

.

## 3 Normative References

<include standard clause>

American National Standard *ANSI INCITS 359-2004* Role Based Access Control

41

## 42 **4 Terms and Definitions**

43 *Note: check to see if these terms are consistent with INCITS 359.*

44

45

46 For the purposes of this document, the following terms and definitions apply.

47 **Component:** A *Component* refers to one of the major blocks of RBAC features, core  
48 RBAC, hierarchical RBAC, SSD relations, and DSD relations.

49 **Feature:** A *Feature* is loosely defined as an item contained within an RBAC design to  
50 provide functionality.

51 **Object:** As used in this standard, an *object* can be any system resource subject to access  
52 control, such as a file, printer, terminal, database record, etc.

53 **Operation:** An *operation* is an executable image of a program, which upon invocation  
54 executes some function for the user.

55 **Policy Decision Point (PDP):** A locus where policy rules have been resolved, evaluated,  
56 and combined to yield a binary value for interpretation by a Policy Enforcement Point.

57 **Policy Enforcement Point (PEP):** A logical point where a policy decision is used to  
58 grant or deny access to a protected resource.

59 **Permission:** A *Permission* is an approval to perform an operation on one or more RBAC  
60 protected objects.

61 **Role:** A *role* is a job function within the context of an organization with some associated  
62 semantics regarding the authority and responsibility conferred on the user assigned to the  
63 role.

64 **Sessions:** A mapping between a user and an activated subset of roles that are assigned to  
65 the user.

66 **User:** A *user* is defined as a human being. Although the concept of a user can be  
67 extended to include machines, networks, or intelligent autonomous agents, the definition  
68 is limited to a person in this document for simplicity reasons.

69

## 70 **5 Symbols and Abbreviated Terms**

71

72 *Add abbreviations.*

## 73 **6 Requirements**

74 The requirements are focused on support for a set of use cases. For each set of  
75 components (see section **Error! Reference source not found.**) included in a design, the  
76 corresponding use cases (see section 6.2) must be supported.

77

### 78 **6.1 Combinations of Components**

79 The RBAC standard's system and administrative functional specification contains  
80 descriptions of functions for four RBAC components. An implementation shall contain a  
81 set of components selected from the following:

# DRAFT

- 82  
83 1. Core RBAC,  
84 2. Hierarchical RBAC,  
85 3. Static Separation of Duty (SSD) Relations, and  
86 4. Dynamic Separation of Duties (DSD) Relations.  
87

88 Several options exist for conforming to this standard. All options include Core RBAC,  
89 which is defined in clause 6.1 of INCITS 359-2004. The options are defined as  
90 combinations of Core RBAC with one or more of the remaining three of the RBAC  
91 components, as illustrated in Table 1. Options for Inclusion of Components.  
92

93 **Table 1. Options for Inclusion of Components**

Component	Fundamental	Organizational	User Limiting - Universal	User Limiting - Operational
1. Core RBAC	•	•	•	•
2. Hierarchical RBAC		•		
3. Static Separation of Duty (SSD) Relations			•	
4. Dynamic Separation of Duties (DSD) Relations				•

94  
95 It is recognized that an RBAC implementation may contain additional combinations of  
96 the four components. For such an implementation, the terms in the heading of Table 1  
97 shall be used in combination. For example, an implementation containing both static and  
98 dynamic separation of duty constraints would be designated as “User Limiting –  
99 Universal plus User Limiting – Operational.” An implementation containing all four  
100 components would be designated as “Organizational plus User Limiting – Universal plus  
101 User Limiting – Operational.”

## 102 **6.2 Interoperability**

### 103 **6.2.1 RBAC Interaction Functions for Interoperability<sup>1</sup>**

#### 104 **6.2.1.1 Operational**

105 There are no standard operational interactions between RBAC implementations. For  
106 example, no run-time exchange of data is anticipated.

#### 107 **6.2.1.2 Management**

108 Management interaction functions are listed in Table x.

Interaction Function	Meaning
PostRoleName(rolename)	Inform of a new role name
GetRoleName(rolename)	Obtain new role name
PostUserAssignment(user,role)	Inform of user assignment to a role

<sup>1</sup> Using terminology in NIST SP 800-53.

# DRAFT

GetUserAssignment(user,role)	Obtain user assignment to a role
PostPermissionAssignment(role,permission)	Inform of permission assignment to a role
GetPermissionAssignment(role,permission)	Obtain permission assignment to a role
PostUserSet	Inform of current set of RBAC users
GetUserSet	Obtain current set of RBAC users
PostRoleSet	Inform of current set of roles
GetRoleSet	Obtain current set of roles
PostPermissionSet	Inform of current set of permissions
GetPermissionSet	Obtain current set of permissions
PostRoleUsers(rolename)	Inform of users currently assigned to a given role
GetRoleUsers(rolename)	Obtain users currently assigned to a given role
PostUserRoles(user)	Inform of roles currently assigned to a given user
GetUserRoles(user)	Obtain roles currently assigned to a given user
PostRolePermissions(role)	Inform of permissions currently assigned to a given role
GetRolePermissions(role)	Obtain permissions currently assigned to a given role
PostPermissionRoles(permission)	Inform of roles to which a given permission is assigned
GetPermissionRoles(permission)	Obtain roles to which a given permission is assigned
PostUserAssignmentConstraintStatic(user,role)	Inform of a given user's static assignment constraint
GetUserAssignmentConstraintStatic(user,role)	Obtain a given user's static assignment constraint
PostUserAssignmentConstraintDynamic(user,role)	Inform of a given user's dynamic assignment constraint
GetUserAssignmentConstraintDynamic(user,role)	Obtain a given user's dynamic assignment constraint
PostInheritanceRelationship(role,role)	Inform of an inheritance relationship between two given roles
GetInheritanceRelationship(role,role)	Obtain an inheritance relationship between two given roles

109

110 **6.2.1.3 Technical Controls**

111 Technical interactions between RBAC implementations are not automated. The  
 112 following informational categories are suggested.

113  
114 Languages used (names, versions)  
115 Tools used (names, version)  
116 Standards used (names, versions)  
117

## 118 **6.2.2 Data Requirements for Interoperability**

119 Interoperability is to be achieved through common data specifications and common  
120 interfaces.

### 121 **6.2.2.1 Data Model**

122 Data interchange between RBAC systems shall use the following set of entities. These  
123 are as defined in the OASIS XACML Profile for Role Based Access Control (RBAC).

124  
125 **HasPrivilegesOfRole policy** – an optional type of <Policy> that can be included in a Permission  
126 <PolicySet> to allow support queries asking if a subject “has the privileges of” a specific role.

127 **junior role** – in a role hierarchy, Role A is *junior* to Role B if Role B inherits all the permissions  
128 associated with Role A.

129 **PDP** - Policy Decision Point. An entity that evaluates an access request against one or more  
130 policies to produce an access decision.

131 **permission** – the ability or right to perform some action on some resource, possibly only under  
132 certain specified conditions.

133 **role** – a job function within the context of an organization that has associated semantics  
134 regarding the authority and responsibility conferred on the user assigned to the role [ANSI-  
135 RBAC].

136 **senior role** – in a role hierarchy, Role A is *senior* to Role B if Role A inherits all the permissions  
137 associated with Role B.

138 **policy** – a set of rules indicating which subjects are permitted to access which resources using  
139 which actions under which conditions.

140  
141 1. **Role <PolicySet> or RPS** : a <PolicySet> that associates holders of a given role attribute and  
142 value with a Permission <PolicySet> that contains the actual permissions associated with the  
143 given role. The <Target> element of a Role <PolicySet> limits the applicability of the <PolicySet>  
144 to subjects holding the associated role attribute and value. Each Role <PolicySet> references a  
145 single corresponding Permission <PolicySet> but does not contain or reference any other  
146 <Policy> or <PolicySet> elements.

147 2. **Permission <PolicySet> or PPS**: a <PolicySet> that contains the actual permissions  
148 associated with a given role. It contains <Policy> elements and <Rules> that describe the  
149 resources and actions that subjects are permitted to access, along with any further conditions on  
150 that access, such as time of day. A given Permission <PolicySet> may also contain references to  
151 Permission <PolicySet>s associated with other roles that are *junior* to the given role, thereby  
152 allowing the given Permission <PolicySet> to inherit all permissions associated with the role of  
153 the referenced Permission <PolicySet>. The <Target> element of a Permission <PolicySet>, if  
154 present, must not limit the subjects to which the <PolicySet> is applicable.

155 3. **Role Assignment <Policy> or <PolicySet>**: a <Policy> or <PolicySet> that defines which  
156 roles can be enabled or assigned to which subjects. It may also specify restrictions on  
157 combinations of roles or total number of roles assigned to or enabled for a given subject. This  
158 type of policy is used by a Role Enablement Authority.

159 4. **HasPrivilegesOfRole <Policy>**: a <Policy> in a Permission <PolicySet> that supports  
160 requests asking whether a subject has the privileges associated with a given role. If this type of  
161 request is to be supported, then a HasPrivilegesOfRole <Policy> must be included in each

# DRAFT

162 Permission <PolicySet>. Support for this type of <Policy>, and thus for requests asking whether a  
163 subject has the privileges associated with a given role, is optional.

164  
165 Permission <PolicySet> instances must be stored in the policy repository in such a way that they  
166 can never be used as the initial policy for an XACML PDP; Permission <PolicySet> instances  
167 must be reachable only through the corresponding Role <PolicySet>. This is because, in order to  
168 support hierarchical roles, a Permission <PolicySet> must be applicable to every subject. The  
169 Permission <PolicySet> depends on its corresponding Role <PolicySet> to ensure that only  
170 subjects holding the corresponding role attribute will gain access to the permissions in the given  
171 Permission <PolicySet>.

172 Use of separate Role <PolicySet> and Permission <PolicySet> instances allows support for  
173 Hierarchical RBAC, where a more *senior* role can acquire the permissions of a more *junior* role. A  
174 Permission <PolicySet> that does not reference other Permission <PolicySet> elements could  
175 actually be an XACML <Policy> rather than a <PolicySet>. Requiring it to be a <PolicySet>,  
176 however, allows its associated role to become part of a role hierarchy at a later time without  
177 requiring any change to other policies.

178

## 179 **6.2.3 Interface Requirements for Interoperability**

### 180 **6.2.3.1 Policy Decision Point**

181 The concept of Policy Decision Point (also known as Access Control Decision Function)  
182 is where policy rules have been resolved, evaluated, and combined to yield a binary value  
183 for interpretation by a Policy Enforcement Point. The OASIS XACML standard defines  
184 Policy Decision Point and its implementation using the XACML language.

185

### 186 **6.2.3.2 Policy Enforcement Point**

187 The concept of Policy Enforcement Point (also known as Access Control Enforcement  
188 Function) is where a policy decision is used to grant or deny access to a protected  
189 resource. A Policy Enforcement Point typically exists within an application.

190

## 191 **6.3 User Interfaces**

192 Standard for any user interface ISO 13407 - *Human Centred Design Process for*  
193 *Interactive Systems*

194

195

## 196 **6.4 Audit**

197

### 198 **6.4.1 Audit Trail Content**

199

### 200 **6.4.2 Audit Recording**

201

202 **6.4.3 Audit Compliance**

203  
204  
205

206 **7 Normative Annexes**

207  
208

209 **8 Informative Annexes**

210 **8.1 Discussion on Two Types of Roles: Basic and Functional**

211 Provided by US Department of Veterans Affairs, Veterans Health Administration

212 **8.1.1 Purpose**

213 "Basic"<sup>2</sup> roles being defined in ASTM and elsewhere provide a means to enforce  
214 "connect" authorizations for authenticated users independent of determining functional  
215 roles and authorizing detailed operations on protected information objects. Basic roles  
216 "firewall" information resources by effectively managing which applications and  
217 workflows are permitted to a user in the first place. Basic roles support service-based  
218 architectures where it is desirable to centrally manage user access to protected resources.  
219 It is advantageous in these environments to place the concepts and definitions of  
220 "functional" roles into a context that includes basic roles. The basic role can be  
221 considered to be a type of prerequisite role, i.e., supporting a user authorization that  
222 occurs before other roles can be activated.<sup>3</sup>

223 **8.1.2 Discussion**

224 In Figures 1 and 2 in the RBAC standard, user activation of roles follows once a session  
225 is established. The session activates a subset of the user's assigned roles (session roles).  
226 This subset of roles consists of functional (dynamic) roles. They are dynamic because  
227 they are activated in the context of the session and user session attributes. They contain  
228 the permissions that a user has available once the session is established and the roles are  
229 activated. Implementations of such roles are typically managed in applications,  
230 directories, and attribute certificates.

231  
232 In establishing the session, however, there is an implicit assumption that the users in  
233 Figures 1 and 2 are in fact authenticated users of the system who are authorized to invoke  
234 certain permissions, such as opening a session. Thus, these authenticated users have  
235 implicit or explicit permissions to initiate sessions. Functional role activation (session

---

<sup>2</sup> Basic roles are defined in Bernd Blobel's *Analysis, Design and Implementation of Secure and Interoperable Distributed Health Information Systems* (2002). These are alternatively called static roles, or role groups by other sources.

<sup>3</sup> A basic role could also serve as a functional role, should the security policy permit.

# DRAFT

236 roles) cannot occur until the session is established, and authorization to establish the  
237 session may occur outside of the application authorization functions. To accomplish this  
238 basic connect function, the user would possess, in addition to authentication information,  
239 some set of basic (static) roles that would be prerequisites to a user's being authorized to  
240 "connect" to the task or workflow containing the session (functional) roles. An access  
241 control enforcement function would have the responsibility to grant or deny the session  
242 based on the basic role.

243

244 Basic roles therefore contain permission to participate in specific workflows or tasks that  
245 require access to an information object such as a database that is managed by session  
246 oriented functional roles. Basic roles allow basic "connect" permission to task-related  
247 information stores. Basic roles would be typically managed in identity certificates or  
248 directories.

### 249 **8.1.3 Basic Roles vs Organizational Roles**

250 Organizational roles are those roles that reflect the organization chart of an enterprise.<sup>4</sup>  
251 In some cases, organizational roles and basic roles may be conterminous. The distinction  
252 between the two is that organizational roles are taken from the organizational structure.  
253 Basic roles may or may not also be organizational roles.

### 254 **8.1.4 Roles in ASTM Healthcare Policy and Standard Guide**

255 Basic Role groups may be found as categories of subscribers for healthcare certificates in  
256 the ASTM digital certificate policy (ASTM E 2212)<sup>5</sup>. This policy provides for roles and  
257 credentials for healthcare organizations to use that are part of non-critical extensions to  
258 an X.509 v3 PKI certificate. ASTM views these roles as basic roles. They are "static"  
259 since they are part of the identity certificate, and exist as long-term attributes of the user.

260

261 The ASTM Standard Guide for Information Access Privileges to Health Information (E  
262 1986)<sup>6</sup> represents healthcare basic roles suitable for use in this standard. Some  
263 healthcare basic role examples include: Physician, Pharmacist, Advanced Practice  
264 Registered Nurse, and Ward Clerk. These are basic roles suitable for session connect  
265 privilege that do not necessarily specify what the user can do once connected.

266

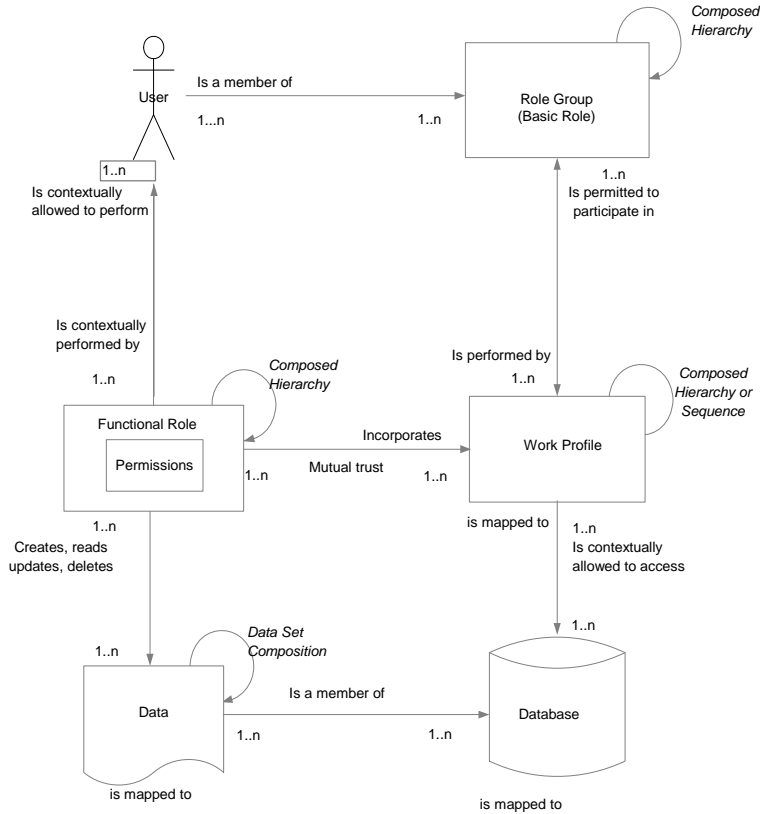
267 While these ASTM standards are oriented to healthcare, the concepts pertain to any  
268 business area.

---

<sup>4</sup> See David F. Ferraiolo, D. Richard Kuhn, and Ramaswamy Chandramouli, *Role-Based Access Control* (2003).

<sup>5</sup> ASTM International E2212-02a Standard Practice for Healthcare Certificate Policy

<sup>6</sup> ASTM International E1986-98(2005)-02a Standard Guide for Information Access Privileges to Health Information



**Figure 1: Role Engineering Model**

269  
270  
271

Figure 1<sup>7</sup> illustrates the relationships between role groups (basic roles, static roles) work profiles, and functional roles (groups of permissions) consistent with the ASTM healthcare policy and the RBAC standard.

272  
273  
274  
275

**Role groups** (basic roles) place people within an organization’s personnel (not necessarily organizational) structure into categories of personnel warranting differing levels of access control. Role groups allow users to participate in the organization’s workflow (e.g., tasks) by job, title, or position but do not specify detailed permissions on specific information objects. As stated earlier, role groups can allow a user to “connect” to a resource but do not necessarily grant finer-grain authorizations on protected information objects.

276  
277  
278  
279  
280  
281  
282  
283

As depicted in Figure 2 (extracted from Figure 1), role groups define what specific work profiles users are allowed to perform, while functional roles define what authorizations are needed by an entity to access protected information technology or application resources.

284  
285  
286  
287

<sup>7</sup> Adapted from Health Level 7 Security Technical Committee

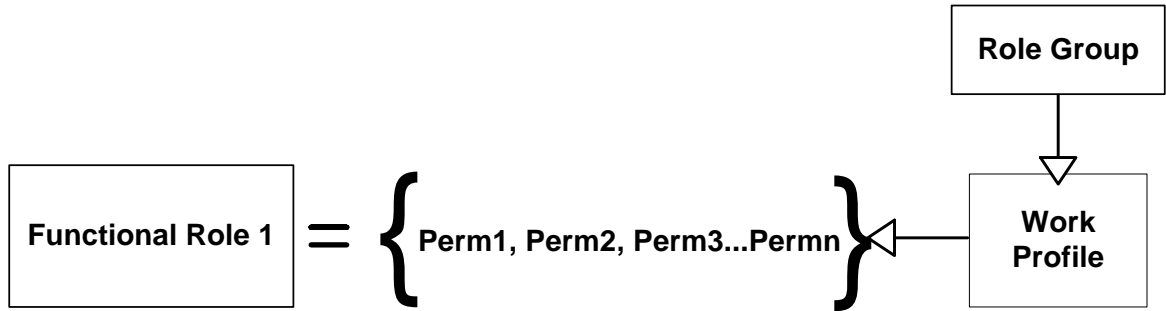


Figure 2: Role Groups and Functional Roles

288  
289  
290

### 291 8.1.5 Conclusion

292 Basic roles can provide the basic connect permissions that precede the user activation of  
293 roles described in the core RBAC and hierarchical RBAC models.

294  
295 (Best practices should be provided in an informative annex, e.g., consistency checking)

- 296
- 297
- 298 OASIS eXtensible Access Control markup Language (XACML) ver0, OASIS Standard,
- 299 1 February 2005
- 300 Core and hierarchical role based access control (RBAC) profile of XACML v2.0, OASIS
- 301 Standard, 1 February 2005
- 302 ISO/IEC 10181-3:1996 Information Technology – Open Systems Interconnection –
- 303 Security framework for open systems: Access Control framework

304  
305  
306  
307

## 306 Appendices

### 308 **Appendix A: Rationale**

- 309
- 310 A.1 <as appropriate for those sections for which explanation is desired>
- 311
- 312 ...
- 313
- 314 A.6.1
- 315
- 316 A.6.2
- 317
- 318 A.6.3...
- 319
- 320