Role Based Access Control (RBAC) is an authorization strategy in which an entity’s permission to access and manipulate targeted resources is determined by the entity’s role or function within a certain organizational context. RBAC’s principal motivation is to streamline security policy administration. Many discrete authorizations can be aggregated within a defined role. One or many roles may be assigned or attributed to individuals. This draft describes LDAP object classes and attributes which support RBAC. Adoption of this schema across multiple LDAP implementations will enable RBAC interoperability among heterogeneous underlying directory services.
1. Introduction

The Lightweight Directory Access Protocol (LDAP) [6,7] is rapidly becoming the ubiquitous mechanism for accessing and manipulating directory data. Many diverse directory implementations, data stores, client applications, and API suites are acquiring LDAP interfaces and functionality.

This widespread LDAP protocol compliance enables interoperability. But protocol compliance in the absence of architecture still leaves us a long way from integration. Unless LDAP-compliant implementations can rely upon each other to provide and interpret meaningful content, little is achieved.

Common schema implementations provide architectural elements in the form of LDAP object classes and attributes. Widespread adoption and support of these architectural elements will add value to interoperable implementations, yielding potential for integration of heterogeneous underlying strategies. This integration is realizable on enterprise and Internet scales.

The scope of functionality addressed by this draft is authorization and access control.

Role Based Access Control (RBAC) [1,2,3] is an authorization strategy in which an entity’s permission to access and manipulate targeted resources is determined by the entity’s role or function within a
certain organizational context. RBAC’s principal motivation is to streamline security policy administration.

Under RBAC, many discrete authorizations can be aggregated within a defined role. One or many roles may be assigned or attributed to individuals. Roles may be constructed or attributed hierarchically, so that one role may be constructed of several other roles, even further streamlining administrative overhead. Some roles are also mutually exclusive or conflicting, implying a requirement for defining such relationships within the architecture.

This draft describes LDAP object classes and attributes which support RBAC. Adoption of this schema across multiple LDAP implementations will enable RBAC interoperability among heterogeneous underlying directory services.

This LDAP RBAC strategy is called "hyperDRIVE" [13], as a working name and convenient discussion handle. The working name will be used throughout this document.

2. Object Class Model

The discovery and definition of the hyperDRIVE objects focuses upon three principal responsibilities of an integrated security function, namely authentication, authorization, and access.

We define each responsibility:

2.1 Access

Access, in the hyperDRIVE context, encompasses the location of target objects, and the methods and modes of access to, and of, network computing resources. Access answers the questions, "What is the target? To which asset(s) are we selectively allowing access? What is the nature of the the access?" For some, the familiar acronym CRUD (create, read, update, delete) is included in this concept, as describing particular modes of access. In RBAC terminology, this object points to an operation or privilege.

The hyperDRIVE object "operationAccessor" (see 4.2) fulfills this responsibility. The operationAccessor’s premier attribute is a URL type. The URL represents a globally unique access reference to the precise target data and functionality intended by the implementor.
2.2 Authorization

Authorization, in the RBAC context, is the mapping or aggregation of one or more operations or privileges to a defined role. Also included are the related constructs of role hierarchies, mutually exclusive roles, and statically defined separation of duties.

The hyperDRIVE object "role" (see 4.3) fulfills this responsibility.

In hyperDRIVE, roles are assigned or mapped to persons, not vice versa as some RBAC literature describes. A person potentially possesses many roles. Roles do not possess persons.

2.3 Authentication

Authentication and authorization are certainly separate responsibilities and separate activities. Much of the published RBAC literature treats the choice of authentication strategy as inconsequential with respect to the functionality of the RBAC strategy. hyperDRIVE, on the other hand, specifies X.509 [12] certificate-based authentication as an integral and indispensible component.

The X.509 certificate’s "subject" attribute (or field), is the Distinguished Name (DN) of the certificate’s possessor, which is generally a person. The DN is a unique direct index into the same LDAP directory information base which holds the operationAccessor and role objects. The X.509 "subject" DN binds the authenticated identity (of the entity who possesses and presents the certificate at run time) to an object in the directory which describes the person. The hyperDRIVE object "hyperDrivePerson" (see 4.4) extends the directory’s person object. The hyperDrivePerson accomplishes the binding of the authenticated identity to the assigned role(s).

3. Object Relationships

This ASCII representation of a Unified Modeling Language (UML) [8] schematic illustrates the static object relationships. Observe that the "Role" object possesses an attribute "operations", which is an aggregation of references to operationAccessor objects. Likewise, the hyperDrivePerson object possesses an attribute "roles", which is an aggregation of references to "Role" objects.
4. Object Class Specifications

4.1. ASN.1 representation

\begin{verbatim}
id-hyprDrv OBJECT IDENTIFIER ::= { /* to be APPLIED-FOR */ }

/**********************************************************
* hyperDRIVE Definitions
* Internal Revenue Service, U.S. Department of the Treasury
* October 10, 1997
* -- Note: borrowed attributes:
* Attribute Source
* commonName X.500 Standard Attributes
* description X.500 Standard Attributes
* member X.500 Standard Attributes
* userCertificate X.500 Standard Attributes
* manager COSINE/Internet X.500 Pilot Attributes
* labeledURL University of Michigan
* ...
*
**********************************************************/

/**********************************************************
* hyperDRIVE base OIDs
**********************************************************/

id-hD OBJECT IDENTIFIER ::= { id-hyprDrv 0 }
id-hd-at OBJECT IDENTIFIER ::= { id-hD 0 }
id-hd-oc OBJECT IDENTIFIER ::= { id-hD 1 }
id-hd-mr OBJECT IDENTIFIER ::= { id-hD 2 }

/**********************************************************
* hyperDRIVE Attributes
**********************************************************/

id-at-hD-organizationalUnitDN OBJECT IDENTIFIER ::= {id-hD-at 0}
id-at-hD-objectOwner OBJECT IDENTIFIER ::= {id-hD-at 1}
id-at-hD-reviewAndApprovalDN OBJECT IDENTIFIER ::= {id-hD-at 2}
id-at-hD-maintenanceDN OBJECT IDENTIFIER ::= {id-hD-at 3}
id-at-hD-roleOwner OBJECT IDENTIFIER ::= {id-hD-at 4}
id-at-hD-locationDN OBJECT IDENTIFIER ::= {id-hD-at 5}
id-at-hD-includedRole OBJECT IDENTIFIER ::= {id-hD-at 6}
\end{verbatim}
id-at-hD-conflictingRole OBJECT IDENTIFIER ::= {id-hD-at 7}
id-at-hD-roles OBJECT IDENTIFIER ::= {id-hD-at 8}
id-at-hD-operations OBJECT IDENTIFIER ::= {id-hD-at 9}
id-at-hD-hyperDriveObjectType OBJECT IDENTIFIER ::= {id-hD-at 10}

/***********************************************************
* hyperDRIVE Structural Object Classes                     *
***********************************************************/
id-oc-hD-operationAccessor OBJECT IDENTIFIER ::= {id-hD-oc 0 }
id-oc-hD-role OBJECT IDENTIFIER ::= {id-hD-oc 1 }

/***********************************************************
* hyperDRIVE NON-Structural Object Classes                *
********************************************************]*)
id-oc-hD-hyperDrivePerson OBJECT IDENTIFIER ::= {id-hD-oc 2 }
id-oc-hD-hyperDriveLocality OBJECT IDENTIFIER ::= {id-hD-oc 3 }
id-oc-hD-hyperDriveOrganizationalUnit OBJECT IDENTIFIER ::= {id-hD-oc 4 }

/***********************************************************
* hyperDRIVE Attribute Definitions and Matching Rules    *
***********************************************************/
organizationalUnitDN ATTRIBUTE ::=  
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID { id-at-hD-organizationalUnitDN }

objectOwner ATTRIBUTE ::=  
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID { id-at-hD-objectOwner }

reviewAndApprovalDN ATTRIBUTE ::=  
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID { id-at-hD-reviewAndApprovalDN }

maintenanceDN ATTRIBUTE ::=  
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID { id-at-hD-maintenanceDN }

roleOwner ATTRIBUTE ::=  
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID { id-at-hD-roleOwner }
locationDN ATTRIBUTE ::= {
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID   { id-at-hD-locationDN } }

includedRole ATTRIBUTE ::= {
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID   { id-at-hD-includedRole } }

conflictingRole ATTRIBUTE ::= {
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID   { id-at-hD-conflictingRole } }

roles ATTRIBUTE ::= {
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID   { id-at-hD-roles } }

operations ATTRIBUTE ::= {
  WITH SYNTAX DistinguishedName
  EQUALITY MATCHING RULE distinguishedNameMatch
  ID   { id-at-hD-operations } }

hyperDriveObjectType ATTRIBUTE ::= {
  WITH SYNTAX DirectoryString
  EQUALITY MATCHING RULE caseIgnoreMatch
  SUBSTRINGS MATCHING RULE caseIgnoreSubstringsMatch
  ID                  { id-at-hD-hyperDriveObjectType } }

/***********************************************************
*  hyperDRIVE Object Class Definitions                      *
***********************************************************/

operationAccessor OBJECT-CLASS ::= {
  SUBCLASS OF top
  MUST CONTAIN commonName
  MAY CONTAIN { description | objectOwner | organizationalUnitDN | labeledURL | userCertificate | reviewAndApprovalDN | maintenanceDN | member | hyperDriveObjectType }
  ID   { id-oc-hD-operationAccessor } }
role OBJECT-CLASS ::= {
  SUBCLASS OF top
  MUST CONTAIN commonName
  MAY CONTAIN { description |
    roleOwner | organizationalUnitDN |
    reviewAndApprovalDN |
    maintenanceDN |
    includedRole |
    conflictingRole |
    operations }
  ID { id-oc-hD-role } }

hyperDrivePerson OBJECT-CLASS ::= {
  SUBCLASS OF person
  MAY CONTAIN { objectOwner |
    reviewAndApprovalDN |
    maintenanceDN |
    locationDN |
    organizationalUnitDN |
    roles }
  ID { id-oc-hD-hyperDrivePerson } }

hyperDriveLocality OBJECT-CLASS ::= {
  SUBCLASS OF locality
  MAY CONTAIN { labeledURL |
    hyperDriveLocality |
    organizationalUnitDN |
    member }
  ID { id-oc-hD-hyperDriveLocality } }

hyperDriveOrganizationalUnit OBJECT-CLASS ::= {
  SUBCLASS OF organizationalUnit
  MAY CONTAIN { manager |
    labeledURL |
    hyperDriveOrganizationalUnit |
    locationDN |
    member }
  ID { id-oc-hD-hyperDriveOrganizationalUnit } }

/***********************************************************
*  END hyperDRIVE Definitions                             *
***********************************************************/

/*----------------------------------------------------------------------------------------*/
*   END hyperDRIVE Definitions                                                            *
*----------------------------------------------------------------------------------------*/
4.2. operationAccessor

Description:

The operationAccessor object class is the directory’s representation of an operation (referenced by the URL) which requires restricted access authorization and navigational representation in the directory.

Attributes:

- **commonName**: the full, formal name of the operationAccessor.
- **description**: mildly verbose description.
- **objectOwner**: distinguished name(s) of the persons or role(s) which "own" the right to permit or deny authorization to access and manipulate the target object/operation.
- **organizationalUnitDN**: the distinguished name of the highest-level organizational unit to which the target object/operation belongs.
- **execURL**: the URL through which the target object/operation is accessed, executed, manipulated.
- **applicationCertificate**: the X.509 certificate, containing public key and other attributes, which the target object/operation uses to identify itself and its actions.
- **review+ApprovalDN**: the DN of the operationAccessor through which the owner’s right and responsibility is exercised.
- **maintenanceDN**: the DN of the operationAccessor through which maintenance on this object is performed.
- **member**: distinguished name(s) of directory object(s) which are permitted access to the target.
- **hyperDriveObjectType**: a string value which indicates what general type of object is represented. Suggested values include: "production", "information", "administrative", and "review+approval".

Naming Rules:

The commonName attribute must be used for naming.
Structure Rules:

A directory entry for operationAccessor must have an immediately superior entry of organizationalUnit.

Relationships:

organizationalUnit
role

4.3. role

Description:

The role object is a container for aggregations of operationAccessor objects.

Attributes:

- commonName: the full, formal name of the role
- description: mildly verbose description
- roleOwner: distinguished name(s) of the persons, or roles which "own" the right to permit or deny assignment of the role to persons.
- organizationUnitDN: the distinguished name of the highest-level organizational unit to which the role object belongs.
- review+ApprovalDN: the DN of the operationAccessor object through which the owner's right and responsibility is exercised.
- maintenanceDN: the DN of the operationAccessor object through which maintenance on this object is performed.
- includedRole: a list of one or more distinguished names (DN) of role object. This attribute supports the implementation of role hierarchies.
- conflictingRole: a list of one or more distinguished names (DN) of role object. This attribute supports the implementation of mutually exclusive role policies.
- operations: a list of one or more distinguished names (DN) of operationAccessor object.

Naming Rules:

The commonName attribute must be used for naming.

Structure Rules:

A directory entry for operationAccessor must have an immediately superior entry of organizationalUnit.

Relationships:

organizationalUnit
operationAccessor
hyperDrivePerson

4.4. hyperDrivePerson

Description:

The hyperDrivePerson object is an extension of the directory’s person object, and is a container for aggregations of role objects.

Attributes:

- objectOwner: distinguished name(s) of the person(s), or role(s) which "owns" the right to permit or deny authorization to access and manipulate this object.

- review+ApprovalDN: the DN of the operationAccessor through which the owner’s right and responsibility is exercised.

- maintenanceDN: the DN of the operationAccessor through which maintenance on this object is performed.

- roles: a list of one or more distinguished names (DN) of role object. This attribute accomplishes the assignment of role objects to the person.

Naming Rules:
The commonName attribute must be used for naming.

Structure Rules:

hyperDrivePerson is a non-structural object class which extends another person-descriptive object class, such as newPilotPerson.

Relationships:

organizationalUnit

role

5. Behavioral Overview

The runtime behavior of the hyperDRIVE objects is described in terms of a proof-of-concept system- or enterprise-scale framework. A framework named hyperDRIVE was developed to demonstrate the functionality of the RBAC LDAP infrastructure. See [13] for an explicit description.

The hyperDRIVE framework appears to qualify as a system-scale or enterprise-scale framework, as described in [14], "Horizontal-Vertical-Metadata" (HVM) design pattern. As per the HVM design pattern, the LDAP-accessible repositories are Metadata and hyperDRIVE provides a substantial portion of the functionality of the Horizontal framework.

In brief:

hyperDRIVE’s authentication mechanism is via X.509 certificates, which clients and servers exchange and verify to establish mutually authenticated Secure Socket Layer (SSL) [9] sessions.

hyperDRIVE’s authorization model is Role Based Access Control (RBAC). The Lightweight Directory Access Protocol (LDAP) directory contains objects which describe people (hyperDrivePerson) and objects which describe enterprise information resources and the operations which involve those resources (operationAccessor). Additional directory objects (role) are aggregations of operationAccessor. One or more role objects are assigned to each person, thus achieving the RBAC objective.

hyperDRIVE provides its customers with a Java applet, which is a GUI
navigation guide, or menu. The hyperDRIVE navigation guide is con-
structed on the fly from the customer’s LDAP-hosted RBAC profile.

The hyperDRIVE Guide applet establishes distributed object communica-
tion with an object request broker (ORB) server which resides,
according to the Java applet “sandbox” policy [5], on the same host
from which the applet was served.

The client applet requests role objects and operationAccessor objects
which apply to the customer’s DN. We refer to distributed object com-
unication by the name of the Object Management Group’s Common Object
Request Broker Architecture (OMG/CORBA) [11] Internet Inter-ORB pro-
tocol, or IIOP.

The hyperDRIVE proof-of-concept framework uses a freeware ORB, known
as HORB [10]. The development roadmap for hyperDRIVE portends a tran-
sition to CORBA compliance and IIOP.

The ORB contacts the well-known LDAP server, requesting role and
operationAccessor objects which apply to the customer’s DN.

The LDAP server is "well-known" to the ORB server because its name
was given to the ORB server in a list at the time the ORB server was
instantiated. In hyperDRIVE, ORB servers and active objects are pro-
vided with lists of trusted LDAP servers, just as SSL servers are
provided with a list of trusted certificate authorities.

The LDAP server provides the requested role and operationAccessor
objects.

The ORB provides the requested directory objects. The hyperDRIVE
Guide applet can now act as a navigation tool for the customer,
displaying names, descriptions, and links which describe operations
for which the customer is authorized.

Through the hyperDRIVE Guide applet, the customer invokes an opera-
tion. The customer’s web client suite invokes the URL against the
target, while simultaneously attaining a mutually authenticated SSL
session with the web server which hosts the targeted operation.

The targeted server and active objects are (via SSL) provided with
the customer’s authenticated identity. Just as the hyperDRIVE Guide
applet, the targeted services will use the customer’s DN (from the
X.509 certificate) as an index into the LDAP RBAC information base.

hyperDRIVE provides servers, applications, and active objects with
capabilities and facilities to consult the LDAP-hosted RBAC data.
Through this consultation, the entities assure themselves of the
appropriateness of customer requests. hyperDRIVE empowers servers to protect their resources. It empowers applications and objects to protect themselves, in the manner of [4].

6. Security Considerations

This draft describes a directory infrastructure which may be used to implement and manage a security policy. The data described by this schema should be protected from casual observance (i.e. "browsing") and must be protected from anonymous or unauthorized manipulation. Implementors must exercise due diligence in assuring the authenticated identity of any entities which are allowed to access and manipulate the data described by this schema. The degree of rigor applied to the authentication process must be commensurate with the sensitivity of the data or processes which are represented by the schema’s accessor objects.

To this end, the authorization parameters of the directory implementation underlying the LDAP interface, as well as the authorization policies of the LDAP interface itself, should be set to the maximum level of restriction which allows the intended functionality.

Failure to apply this strategy with due diligence may result in exposure of the assets the strategy is intended to shield.

7. References


URL:http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html


8. Author’s Address

Larry Bartz  
Internal Revenue Service  
575 N. Pennsylvania Street  
Attn: Stop 15  
Indianapolis, IN 46204  
USA

Phone: +1 317 226-7060  
Email: lbartz@infostream.cr.irs.gov