IBM SecureWay Directory Version 3.2.1: Access Control Lists

Preface

This is a temporary document intended to make Access Control List (ACL) information available to users without having to access the Directory Management Tool help panels. This information will be incorporated into a formal Administration Guide at a later release.

Contents

Access Control Lists

Access Control Lists (ACL) provide a means to protect information stored in a LDAP directory. Administrators use ACLs to restrict access to different portions of the directory, or specific directory entries. LDAP directory entries are related to each other by a hierarchical tree structure. Each directory entry (or object) contains the distinguished name of the object as well as a set of attributes and their corresponding values.

The object attributes associated with access control, such as owner, ownerSource,ownerPropagate, acl, aclSource and aclPropagate are unusual in that they are logically associated with each object, but can have values that depend upon other objects higher in the tree. Depending upon how they are established, these attribute values can be explicit to an object or inherited from an ancestor.

The access control model defines two sets of attributes: The Access Control Information (ACI) and the entryOwner Information. The ACI specifically defines a subject's permission to perform a given operation against certain LDAP objects. The entryOwner information controls which subjects can define the ACIs. The entryOwnership also acquires full access rights to the target object.

Using Access Control Information (ACI), administrators can restrict access to different portions of the directory, specific directory entries and, based on the attribute name or attribute access class, the attributes contained in the entries. Each entry within the LDAP directory has a set of associated ACI. In conformance with the LDAP model,the ACI and entryOwner informationis represented as attribute-value pairs. Furthermore, the ldif syntax is used to administer these values. The attributes are:

- aclEntry
- aclPropagate
- entryOwner
- ownerPropagate

The Access Control Attribute Syntax

Each of these attributes can be managed using LDIF notation. The following defines the syntax for the ACI and entryOwner attributes using BNF.

```
<aclEntry> ::=  <subject> [ ":"  <rights> ]

  <aclPropagate> ::=  "true" | "false"

  <entryOwner> ::=  <subject>

  <ownerPropagate> ::= "true" | "false"

  <subject> ::= <subjectDnType> ':' <subjectDn> | <pseudoDn>

  <subjectDnType> ::= "role" | "group" | "access-id"

  <subjectDn> ::= <DN>

  <DN> ::= distinguished name as described in RFC 2251, section 4.1.3.

  <pseudoDn> ::= "group:cn=anybody" | "group:cn=authenticated" | "access-id:cn=this"
```

```
<rights> ::= <accessList> [":" <rights> ]

<accessList> ::=  <objectAccess> | <attributeAccess> | <attributeClassAccess>

<objectAccess> ::= "object:" [<action> ":"]  <objectPermissions>

<action> ::= "grant" | "deny"

<objectPermisssions> ::=  <objectPermission> [ <objectPermissions> ]

<objectPermission> ::= "a" | "d" |  ""

<attributeAccess> ::= "at." <attributeName> ":" [<action> ":"]  <attributePermissions>

<attributeName> ::= attributeType name as described in RFC 2251, section 4.1.4. (OID or
alpha-numeric string with leading alphabet, "-" and ";" allowed)

<attributePermissions> ::=  <attributePermission> [<attributePermissions>]

<attributePermission> ::=  "r" | "w" | "s" | "c" |  ""

<attributeClassAccess> ::= <class> ":" [<action> ":"] <attributePermissions>

<class> ::= "normal" | "sensitive" | "critical"
```

AclEntry

Subject

A subject (the entity requesting access to operate on an object) consists of the combination of a DN (Distinguished Name) type and a DN. The valid DN types are: access Id, Group and Role.

The DN identifies a particular access-id, role or group. For example, a subject might be access-id: cn=personA, o=IBM or group: cn=deptXYZ, o=IBM.

Because the field delimiter is the colon (':'), a DN containing colon(s) must be double-quoted. And a double-quoted DN must escape any double-quote sign with the escape character '\', should it be present in the DN.

All directory groups can be used in access control.

Note: Version 3.1 and earlier LDAP servers restricted Groups which are to be used in access control to have an objectclass of AccessGroup. This restriction is lifted at Version 3.2. Any group of **AccessGroup**, **GroupOfNames** or **'GroupofUniqueNames'** objectclasses can be used for access control.

Another DN type used within the access control model is role. While roles and groups are similar in implementation, conceptually they are different. When a user is assigned to a role, there is an implicit expectation that the necessary authority has already been set up to perform the job associated with that role. With group membership, there is no built in assumption about what permissions are gained (or denied) by being a member of that group.

Roles are similar to groups in that they are represented in the directory by an object. Additionally, roles contain a group of DNs. Roles that are used in access control must have an objectclass of **AccessRole**.

Pseudo DN

The LDAP directory contains several pseudo DNs. These are used to refer to large numbers of DNs which at bind time share a common characteristic, in relation to either the operation being performed, or the target object on which the operation is being performed.

Currently, three pseudo DNs are defined:

group:cn=anybody
Refers to all subjects, including those that are unauthenticated. All users belong to this group automatically.

group:cn=authenticated
Refers to any DN which has been authenticated to the directory. The method of authentication is not considered.

access-id:cn=this
Refers to the bindDn which matches the target object's DN on which the operation is performed.

Rights

Access rights can apply to an entire object or to attributes of the object. The LDAP access rights are discrete. One right does not imply another right. The rights may be combined together to provide the desired rights list following a set of rules discussed later. Rights can be of null value, which indicates that no access rights are granted to the subject on the target object. The rights consist of three parts:

Action:
Defined values are **grant** or **deny**. If this field is not present, the default is set to **grant**.

Permission:
There are six basic operations that may be performed on a directory object. From these operations, the base set of ACI permissions are taken. These are: add an entry, delete an entry, read an attribute value, write an attribute value, search for an attribute, and compare an attribute value.

The possible attribute permissions are : read (r), write (w), search (s), and compare (c). Additionally, object permissions apply to the entry as a whole. These permissions are add child entries (a) and delete this entry (d).

The following table summarizes the permissions needed to perform each of the LDAP operations.

Table 1.

Operation	Permission Needed
ldapadd	add (on parent)
ldapdelete	delete (on object)
ldapmodify	write (on attributes being modified)

Table 1. (continued)

ldapsearch	• search, read (on attributes in RDN) • search (on attributes specified in the search filter) • search (on attributes returned with just names) • search, read (on attributes returned with values)
ldapmodrdn	write (on RDN attributes)
ldapcompare	compare (on compared attribute)

Note: For search operations, the subject is required to have search (s) access to all the attributes in the search filter or no entries are returned. For returned entries from a search, the subject is required to have search (s) and read (r) access to all the attributes in the RDN of the returned entries or these entries are not returned.

Access Target:

These permissions can be applied to the entire object (add entry, delete entry), to an individual attribute within the entry, or can be applied to groups of attributes (Attribute Access Classes) as described in the following.

Attributes requiring similar permissions for access are grouped together in classes. Attributes are mapped to their attribute classes in the directory schema file. These classes are discrete; access to one class does not imply access to another class. Permissions are set with regard to the attribute access class as a whole. The permissions set on a particular attribute class apply to all attributes within that access class unless the individual attribute access permissions are specified.

IBM defines three attribute classes that are used in evaluation of access to user attributes: **normal**, **sensitive**, and **critical**. For example, attribute **commonName** falls into the normal class, and attribute userpassword belongs to the critical class. User defined attributes belong to the normal access class unless otherwise specified.

Two other access classes are also defined: system and restricted. The system class attributes are **creatorsName**, **modifiersName**, **createTimestamp**, **modifyTimestamp**, **ownerSource** and **aclSource**. These are attributes maintained by the LDAP server and read only to the directory users. **OwnerSource** and **aclSource** are described in the Propagation section. The attributes that define the access control, namely **aclEntry**, **aclPropagate**, **entryOwner** and **ownerPropagate**, are in the restricted class. All users have read access to the restricted attributes but only **entryOwners** can create, modify, and delete these attributes.

EntryOwner

The entry owners have complete permissions to perform any operation on the object regardless of the aclEntry. Additionally, the entry owners are the only ones who are permitted to administer the aclEntries for that object. EntryOwner is an access control subject, it can be defined as individuals, groups or roles.

Note: The directory administrator is one of the entryOwners for all objects in the directory by default, and the directory administrator's entryOwnership can not be removed from any object.

Propagation

Entries on which an aclEntry has been placed are considered to have an explicit **aclEntry**. Similarly, if the **entryOwner** has been set on a particular entry, that entry has an explicit owner. The two are not intertwined, an entry with an explicit owner may or may not have an explicit **aclEntry**, and an entry with an explicit **aclEntry** might have an explicit owner. If either of these values is not explicitly present on an entry, the missing value is inherited from an ancestor node in the directory tree.

Each explicit **aclEntry** or **entryOwner** applies to the entry on which it is set. Additionally, the value might apply to all descendants that do not have an explicitly set value. These values are considered propagated; their values propagate through the directory tree. Propagation of a particular value continues until another propagating value is reached.

AclEntry and **entryOwner** can be set to apply to just a particular entry with the propagation value set to ″false″, or an entry and its subtree with the propagation value set to ″true″. Although both **aclEntry** and **entryOwner** can propagate, their propagation is not linked in anyway.

The **aclEntry** and **entryOwner** attributes allow multi-values, however, the propagation attributes (**aclPropagate** and **ownerPropagate**) can only have a single value for all **aclEntry** or **entryOwner** attribute values within the same entry.

The system attributes **aclSource** and **ownerSource** contain the DN of the effective node from which the **aclEntry** or **entryOwner** are evaluated, respectively. If no such node exists, the value **default** is assigned.

An object's effective access control definitions can be derived by the following logic:

- If there is a set of explicit access control attributes at the object, then that is the object's access control definition.
- If there is no explicitly defined access control attributes, then traverse the directory tree upwards until an ancestor node is reached with a set of propagating access control attributes.
- If no such ancestor node is found, the default access described below is granted to the subject.

Access Evaluation

Access for a particular operation is granted or denied based on the subject's bind DN for that operation on the target object. Processing stops as soon as access can be determined.

The checks for access are done by first finding the effective **entryOwnership** and **ACI** definition, checking for entry ownership, and then by evaluating the object's ACI values.

By default, the directory administrator and the master server (for replication) get full access rights to all objects in the directory except write access to system attributes. Other **entryOwners** get full access rights to the objects under their

ownership except write access to system attributes. All users have read access rights to system and restricted attributes. These predefined rights cannot be altered. If the requesting subject has **entryOwnership**, access is determined by the above default settings and access processing stops.

If the requesting subject is not an entryOwner, then the ACI values for the object entries are checked. The access rights as defined in the ACIs for the target object are calculated by the specificity and combinatory rules.

Specificity Rule

The most specific aclEntry definitions are the ones used in the evaluation of permissions granted/denied to a user. The levels of specificity are:

- Access-id is more specific than group or role. Groups and roles are on the same level.
- Within the same **dnType** level, individual attribute level permissions are more specific than attribute class level permissions.
- Within the same attribute or attribute class level, **deny** is more specific than **grant**.

Combinatory Rule

Permissions granted to subjects of equal specificity are combined. If the access cannot be determined within the same specificity level, the access definitions of lesser specific level are used. If the access is not determined after all defined ACIs are applied, the access is denied.

Note: After a matching access-id level **aclEntry** is found in access evaluation, the group level aclEntries are not included in access calculation. The exception is that if the matching access-id level **aclEntries** are all defined under cn=this, then all matching group level **aclEntries** are also combined in the evaluation.

In other words, within the object entry, if a defined ACI entry contains an access-id subject DN that matches the bind DN, then the permissions are first evaluated based on that aclEntry. Under the same subject DN, if matching attribute level permissions are defined, they supersede any permissions defined under the attribute classes. Under the same attribute or attribute class level definition, if conflicting permissions are present, denied permissions override granted permissions.

Note: A defined null value permission prevents the inclusion of less specific permission definitions.

If access still can not be determined and all found matching aclEntries are defined under "cn=this", then group membership is evaluated. If a user belongs to more than one groups, the user receives the combined permissions from these groups. Additionally, the user automatically belongs to the cn=Anybody group and possibly the cn=Authenticated group if the user did an authenticated bind. If permissions are defined for those groups, the user receives the specified permissions.

Note: Group and Role membership is determined at bind time and last until either another bind takes place, or until an unbind request is received. Nested groups and roles, that is a group or role defined as a member of another group or role, are not resolved in membership determination nor in access evaluation.

For example, assume attribute1 is in the sensitive attribute class, and user cn=Person A, o=IBM belongs to both group1 and group2 with the following aclEntries defined:

1. aclEntry: access-id: cn=Person A, o=IBM: at.attributel:grant:rsc:sensitive:deny:rsc
2. aclEntry: group: cn=group1,o=IBM:critical:deny:rwsc
3. aclEntry: group: cn=group2,o=IBM:critical:grant:r:normal:grant:rsc

This user gets:

- Access of 'rsc' to attribute1, (from 1. Attribute level definition supersedes attribute class level definition).
- No access to other sensitive class attributes in the target object, (from 1).
- No other rights are granted (2 and 3 are NOT included in access evaluation).

For another example, with the following aclEntries:

1. aclEntry: access-id: cn=this: sensitive
2. aclEntry: group: cn=group1,o=IBM:sensitive:grant:rsc:normal:grant:rsc

The user has:

- no access to sensitive class attributes, (from 1. Null value defined under access-id prevents the inclusion of permissions to sensitive class attributes from group1).
- and access of 'rsc' to normal class attributes (from 2).

Defining the ACIs and Entry Owners

The following two examples show an administrative subdomain being established. The first example shows a single user being assigned as the entryOwner for the entire domain. The second example shows a group assigned as the entryOwner.

```
entryOwner: access-id:cn=Person A,o=IBM
ownerPropagate: true

entryOwner: group:cn=System Owners, o=IBM
ownerPropagate: true
```

The next example shows how a group "cn=Dept XYZ, o=IBM" is being given permissions to read, search and compare attribute1. The permission applies to the entire subtree below the node containing this ACI.

```
aclEntry: group:cn=Dept XYZ,o=IBM:at.attribute1:grant:rsc
aclPropagate: true
```

The next example shows how a role "cn=System Admins,o=IBM" is being given permissions to add objects below this node, and read, search and compare attribute2 and the critical attribute class. The permission applies only to the node containing this ACI.

```
aclEntry: role:cn=System Admins,o=IBM:object:grant:a:at.attribute2:grant:rsc:critical:grant:rsc
aclPropagate: false
```

Modifying the ACI and Entry Owner Values

Modify-replace

Modify-replace works the same way as all other attributes. If the attribute value does not exist, create the value. If the attribute value exists, replace the value.

Given the following ACIs for an entry:

```
aclEntry: group:cn=Dept ABC,o=IBM:normal:grant:rsc
aclPropagate: true
```

perform the following change:

```
dn: cn=some entry
changetype: modify
replace: aclEntry
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rsc
```

The resulting ACI is:

```
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rsc
aclPropagate: true
```

ACI values for Dept ABC are lost through the replace.

Modify-add

During an ldapmodify-add, if the ACI or entryOwner does not exist, the ACI or entryOwner with the specific values is created. If the ACI or entryOwner exists, then add the specified values to the given ACI or entryOwner. For example, given the ACI:

```
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rsc
```

with a modification:

```
dn: cn=some entry
changetype: modify
add: aclEntry
aclEntry: group:cn=Dept ABC,o=IBM:at.attribute1:grant:rsc
```

would yield an multi-valued aclEntry of:

```
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rsc
aclEntry: group:cn=Dept ABC,o=IBM:at.attribute1:grant:rsc
```

The permissions under the same attribute or attribute class are considered as the basic building blocks and the actions are considered as the qualifiers. If the same permission value is being added more than once, only one value is stored. If the same permission value is being added more than once with different action values, the last action value is used. If the resulting permission field is empty (""), this permission value is set to null and the action value is set to **grant**.

For example, given the following ACI:

```
aclEntry: group:cn=Dept XYZ,O=IBM:normal:grant:rsc
```

with a modification:

```
dn: cn=some entry
changetype: modify
add: aclEntry
aclEntry: group:cn=Dept XYZ,o=IBM:normal:deny:r:critical:deny::sensitive:grant:r
```

yields an aclEntry of:

```
aclEntry: group:cn=Dept XYZ,O=IBM:normal:grant:sc:normal:deny:r:critical:grant::sensitive:gra
```

Modify-delete

To delete a particular ACI value, use the regular ldapmodify-delete syntax.

Given an ACI of:

```
aclEntry: group:cn=Dept XYZ,o=IBM:object:grant:ad
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rwsc
```

```
dn: cn = some entry
changetype: modify
delete: aclEntry
aclEntry: group:cn=Dept XYZ,o=IBM:object:grant:ad
```

yields a remaining ACI on the server of

```
aclEntry: group:cn=Dept XYZ,o=IBM:normal:grant:rwsc
```

Deleting an ACI or entryOwner value that does not exist results in an unchanged ACI or entryOwner and a return code specifying that the attribute value does not exist.

Deleting the ACI/Entry Owner Values

With the ldapmodify-delete operation, the entryOwner can be deleted by specifying

```
dn: cn = some entry
changetype: modify
delete: entryOwner
```

In this case, the entry would then have no explicit entryOwner. The ownerPropagate is also removed automatically. This entry would inherit its entryOwner from the ancestor node in the directory tree following the propagation rule.

The same can be done to delete aclEntry completely:

```
dn: cn = some entry
changetype: modify
delete: aclEntry
```

Deleting the last ACI or entryOwner value from an entry is not the same as deleting the ACI or entryOwner. It is possible for an entry to contain an ACI or entryOwner with no values. In this case, nothing is returned to the client when querying the ACI or entryOwner and the setting propagates to the descendent nodes until it is overridden. To prevent dangling entries that nobody can access, the directory administrator always has full access to an entry even if the entry has a null ACI or entryOwner value.

Retrieving the ACI/Entry Owner Values

The effective ACI or entryOwner values can be retrieved by simply specifying the desired ACL or entryOwner attributes in a search, for example,

```
ldapsearch -b "cn=object A, o=ibm" -s base "objectclass=*"
    aclentry aclpropagate aclsource entryowner ownerpropagate ownersource
```

returns all ACL or entryOwner information that is used in access evaluation on object A. Note that the returned values might not look exactly the same as they are first defined. The values are the equivalent of the original form.

Working with Access Control Lists

Follow these steps to use the Directory Management Tool utility to work with ACLs.

1. If you have not done so already, expand the Directory tree category in the navigation area, then click **Browse tree**.
2. Select a directory entry. For example, cn=John Doe,ou=Advertising,o=ibm,c=US.

3. Click the ACL icon on the toolbar.

An entry can either have an explicitly defined ACL or inherit an ACL from a parent object.

The ACL panel contains two tabs:

- ACLs
- Owners

After modifying the settings on the ACLs and Owners tabs, click **OK** to add the ACL to the selected entry.

The ACLs tab contains four sections:

- The DN entry section displays:
  - **ACL source** - The ACL source is the source of current ACL for the selected entry. If the entry does not have an ACL, it inherits an ACL from parent objects based on the ACL settings of the parent objects.
  - Select either:
    - **Inherit from ACL source** - to inherit ACLs from the ACL source. In the example used above, cn=John Doe,ou=Advertising,o=IBM,c=US inherits from ou=Advertising,o=IBM,c=US

      Note: If this option is selected, and you modify the Rights or Security class, you are modifying the ACL of the ACL source (for example, ou=Advertising,o=IBM,c=US, not the ACL of the selected entry (for example cn=John Doe).
    - **Do NOT inherit from ACL source** - to explicitly define an ACL for the selected entry.
  - **Allow descendant entries to inherit from this entry** - Select the check box to allow descendants without an explicitly defined ACL to inherit from this entry. If the check box is not selected, descendant entries without an explicitly defined ACL inherits ACLs from a parent of this entry that has this option enabled.
  - **Remove ACL and inherit from ACL source** - Select this check box to remove the explicitly defined ACL for this entry and inherit from the ACL source.
- The Subject section displays:
  - The Distinguished Name (DN) of the entity requesting access to perform operations on the selected entry. This can also be a pseudo DN.You can:
    - Select a DN from the drop-down list.
    - Add a new DN to the list.
      1. Replace the entry displayed in the DN field. For example, highlight the text, click the space bar to erase it, then type the DN you want to add, for example, cn=Marketing Group. Do not delete the entry from the drop-down list using Delete.
      2. Select the **Type** of entry for the DN. For example, select access-id if the DN is a user.
      3. Click **Add** to add the specified DN to the drop-down list.
    - Delete a DN from the list.
      1. Select the **DN** from the drop-down list.
      2. Click **Delete**.
    - Click **List all** to display all subjects and their rights and permissions in a tabular format.

– The **Rights** section displays the addition and deletion rights of the subject.
  - **Add child** grants or denies the subject the right to add a directory entry beneath the selected entry.
  - **Delete entry** grants or denies the subject the right to delete the selected entry. In the previous example , it grants or denies cn=Marketing Group the ability to delete cn=John Doe.
– The **Security** class section defines permissions for security classes. Attributes are grouped into security classes:
  - **Normal** - Normal attribute classes require the least security, for example, the attribute commonName.
  - **Sensitive** - Sensitive attribute classes require a moderate amount of security, for example homePhone.
  - **Critical** - Critical attribute classes require the most security, for example, the attribute userpassword.

  Each security class has permissions associated with it.
  - **Read** - the subject can read attributes.
  - **Write** - the subject can modify the attributes.
  - **Search** - the subject can search attributes.
  - **Compare** - the subject can compare attributes.

  Additionally, you may specify permissions based on the attribute instead of the security class to which the attribute belongs.
  1. Select an attribute from the Define an attribute drop-down list.
  2. Click **Define**.
  3. Defined attributes are listed below the **Critical** security class. To remove an attribute, simply select **Unspecified** for all permissions, then click **OK**.

The Owners tab contains two sections:

- The DN entry section displays:
  – The **Owner source** is the source of the current Owner for the selected entry.
  – Select either:
    - **Remove this list from the entry and use propagated values** - to inherit Owner
    - **Descendant directory tree entries inherit from the entry** - to define an Owner for the entry and apply this owner to all descendants that do not have an explicitly set Owner.
- The **Subject** section displays:
  – The **Distinguished name (DN)** of the Owner of the entry. Owners have complete access to all attributes for the entry. If the entry does not have an Owner, it inherits an Owner from parent objects based on the Owner settings of the parent objects. You can:
    - Select a DN from the drop-down list.
    - Add a new DN to the drop-down list:
      1. Replace the entry displayed in the DN field. For example, highlight the text, click the space bar to erase it, then type the DN you want to add, for example, cn=Marketing Group. Do not delete the entry from the drop-down list using Delete.
      2. Select the **Type** of entry for the DN. For example, select access-id if the DN is a user.
      3. Click **Add** to add the specified DN to the drop-down list.

- Delete a DN from the drop-down list:
  1. Select the **DN** from the drop-down list.
  2. Click **Delete**.