UserClasses

> **Work in progress**
>
> This site is in the process of being reviewed and updated.

What are User Classes?

A large part of managing access control information involves the specification of **who** can perform **which** operation on **what** protected resource (entries, attributes, values etc). At evaluation time a requestor of an operation is known. The identity of the requestor is checked to see if it falls into the set of users authorized to perform the operation. User classes are hence definitions of a set of zero or more users permissions apply to. Several constructs exist for specifying a user class.

Simple User Classes

There are 3 really simple constructs for specifying the user. These constructs are listed in the table below:

User Class Construct	Meaning	Example
allUsers	Any user with possible requirements for AuthenticationLevel	**allUsers**
thisEntry	The user with the same DN as the entry being accessed	**thisEntry**
name	The user with the specified DN	**name** { "uid=admin,ou=system" }

These are pretty intuitive. Two other user classes may be a bit less easy to understand or may require some explanation. For these we discuss them in the sections below.

User Class: userGroup

The **userGroup** user class construct is also pretty intuitive. It does however require some background information about how group membership is determined for this purpose.

ApacheDS associates users within a group using the **groupOfNames** and **groupOfUniqueNames** objectClasses. To define groups an entry of either of these objectClasses is added anywhere in the server's DIT. **member** or **uniqueMember** attributes whose values are the DN of user entries are present within the entry to represent membership within the group.

> Although such group entries can be added anywhere within the DIT to be recognized by the Authorization subsystem, a recommended convention exists. Use the 'ou=groups' container under a namingContext/partition within the server to localize groups. Most of the time group information can be stored under 'ou=groups,ou=system'.

Just like the **name** construct, the **userGroup** construct takes a single parameter: the DN of the group entry. During ACI evaluation ApacheDS checks to see if the requestor's DN is contained within the group. Below is a section from X.501 specification which explains just how this is done:

Section 18.4.2.5 Determining group membership (b)

In order to determine whether the requestor is a member of a userGroup user class, the following criteria apply:

- The entry named by the userGroup specification shall be an instance of the object class groupOfNames or groupOfUniqueNames.
- The name of the requestor shall be a value of the member or uniqueMember attribute of that entry.

User Class: subtree

Here the user class specification construct is a subtree specification without a refinement filter. Such a specification is simple yet very powerful. The subtree defines a collection of entries. During ACI evaluation, ApacheDS will check to see if the requestor's DN is included by this collection.

For more information on how to define a subtreeSpecification please see Subentries and the Administrative Model.

> For this purpose a subtree is not refined. Meaning it does not evaluate refinement filters. This is to restrict the information needed to make a determination to just the DN of the requestor and not the entry of the requestor.

Combining Multiple UserClass Specification Mechanisms

The same userClass mechanism can be specified more than once if it makes sense. There is no reason to specify allUsers more than once. More than one type of user class mechanism can be used as well. Again some combinations just will not make sense like having a name based userClass then allUsers. The following ACIItem grants delete abilities to a set of users using more than one machanism. It allows jbean, jdoe, all users in the Administrators group to delete entries. It also allows requestors to delete their own user entry.

```
{ identificationTag "deleteAci"
  precedence 255,
  authenticationLevel simple,
  itemOrUserFirst userFirst:
    {
      userClasses
        {
           thisEntry,
           name { "uid=jbean,ou=users,ou=system" },
           name { "uid=jdoe,ou=users,ou=system" },
           userGroup { "cn=Administrators,ou=groups,ou=system" }
        },
      userPermissions { { protectedItems {entry}, grantsAndDenials { grantRemove } } }
    }
}
```