# University of North Carolina Grouper Project Page





## Grouper Pilot Use Case: Course groups for Office 365

Completed in March 2018 as a pilot project, the Office 365 Course Groups application is an integration of multiple technologies, providing a simple way for class instructors to set up Office 365 groups from their class roster. Instructors can log into a web application to view their assigned course sections for the current term, or for future terms. They can choose from among the sections for a single course, creating an O365 group out of the combined membership of those sections. These groups can be optionally synchronize changes in enrollment from the system of record. Grouper's many components play a central role in tying the system together.

A full write-up can be found at this subpage.

## Custom sync, multiple subject types, building with Maven, deploying to Glassfish

Grouper at UNC-Chapel Hill is used to manage group relationships that are synchronized with both LDAP and AD institutional directories. We upgraded to 2.3.0 in January, 2017. Previously, we had been on v1.6.3 for over 5 years. Certain areas have been customized where necessary, and these are detailed below.

### Subjects and groups

Our sources.xml file defines 4 different sources for subjects. Three of these use our LDAP server and one is from our AD server.

- person (180,000 subjects)
- application (103 subjects)
- legacy application (45 subjects)
- AD service account (7 subjects)

Our groups within Grouper can be separated into three main areas. The organization tree contains a folder for every department, and in each folder is a group for students, staff, faculty, and affiliates. Two additional groups, employee and member, roll up the memberships from certain groups here. The membership in these folders is periodically refreshed based on LDAP queries. The application tree contains functional groups for various purposes, such as authorization to external systems. Membership in these groups is mostly managed through the UI, but some are loader groups refreshed from LDAP or AD queries. The Posix group tree contains a special type of group that includes a GID number attribute managed by IDM. Its sole member is a parallel group in the application tree, where departmental staff have the privileges to manage the membership.

The primary use of Grouper data by applications is authorization checking via its published membership in LDAP and AD. Our Shibboleth SSO configuration also can supply a list of memberships for a logged in user, which web applications can use for authorization purposes.

### Synchronization with LDAP/AD

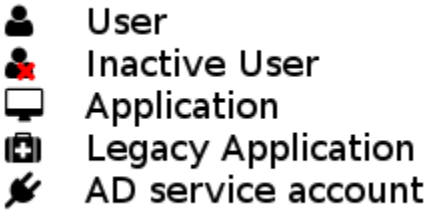
We synchronize between Grouper and LDAP/AD in both directions, via different mechanisms. The processes are mostly custom code. However, in the recent upgrade to 2.3.0, we were successful in converting some of the custom import code to utilize the LDAP Loader process.

For the incoming sync, around 9,000 groups import their membership from LDAP or AD on a periodic basis. Most of these query LDAP; only 90 groups query AD. These groups are set up as LDAP Loader groups, using either the LDAP or AD source we have set up in grouper-loader.properties. Depending on the type of query, some kind of LoaderLdapElUtils class (a standard or an in-house custom one) is usually necessary to convert the query results into subjects. Instead of using the GSH-based daemon to manage these loads, we have a custom daemon that calls the loader individually for each group. That gave us greater control over the scheduling parameters that we needed for our initial rollout.

The outgoing sync is a locally developed solution that originated with our initial 1.x Grouper installation. The software continues to work with 2.3.0, with very few code changes needed. In Grouper, any group can have one or more Boolean attributes set, which indicate the group should push membership to an LDAP group, an AD distribution group, and/or an AD security group. Posix groups have a different set of Boolean attributes that indicate to sync to an LDAP Posix or AD Posix group. The process runs periodically, checking for any groups with changes made since the previous check. A daily process also performs a sync for all groups.

## UI Changes

One notable customization we made to the UI is to display different icons depending on the subject type and subject attributes. Because we have a number of different types, this gives the users a helpful visual cue when managing memberships. Besides the usual user icon for active person subjects, we show a different icon if the subject does not have a current affiliation. This indicates to the membership managers that the subject may be a candidate for removal or further investigation.




## Running on OpenShift cloud hosting with JBoss EAP 7

Instead of an exploded WAR on Tomcat, we were constrained by needing to host this as a packaged WAR file on JBoss EAP 7 in our site-hosted OpenShift cluster. There were a few changes we need to make to the original source in order for a build to work in that environment. Due to a more recent version of the Servlet API, we needed to implement one abstract function of HttpServletResponse, which didn't exist in the servlet version from the Grouper distribution. We also needed to fork some Grouper config reader classes that assumed resources were accessible files; in EAP, all files are on a virtual file system incompatible with the java.io.File class.

## Maven builds

Our local development standards favor Maven over Ant where possible. We were successful in converting the original Grouper distribution into a structure that allowed us to merge our customizations with the official sources, to build a number of artifacts in multiple environments. However, there have been challenges in this process, and these will continue to be hurdles whenever new Grouper versions are incorporated.

**See Also**

Demand Grows for Grouper Groups Management Tool