Agenda

• Grouper Loader LDAP example
• Naming best practice (folders, grouper, roles)
• Setting up reference groups (via loader)
• Composite group setup and management
• Resource/permission inheritance (Penn's unix/tomcat example)
Agenda - continued

• Logical progression from basic to production
• Managing Grouper in multiple environments
Grouper Loader LDAP example

- Searched internet for public LDAP
- ldap.andrew.cmu.edu
- ou=person
- guid=
- cn=John Smith
Grouper Loader LDAP example (continued)

• Need a source with the users in there (normally your installation will already have this)

• Get sources.xml from wiki
Grouper Loader LDAP example (continued)

- Create folder/group test:testGroup
- Use new attribute framework to assign ldap loader
### Grouper Loader LDAP example (continued)

#### Attribute assignments

<table>
<thead>
<tr>
<th>Owner group</th>
<th>Attribute name</th>
<th>Enabled?</th>
<th>Assignment values</th>
<th>Attribute definition</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>testGroup</td>
<td>Grouper loader LDAP</td>
<td>enabled</td>
<td></td>
<td>grouperLoaderLdapDef</td>
<td></td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP subject attribute name</td>
<td>enabled</td>
<td>guild</td>
<td>grouperLoaderLdapDefValueDef</td>
<td>68658...</td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP search base DN</td>
<td>enabled</td>
<td>ou=person</td>
<td>grouperLoaderLdapValueDef</td>
<td>66e3...</td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP quartz cron</td>
<td>enabled</td>
<td>0 8 * ?</td>
<td>grouperLoaderLdapValueDef</td>
<td>76620...</td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP filter</td>
<td>enabled</td>
<td></td>
<td>grouperLoaderLdapValueDef</td>
<td>7cb2...</td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP server ID</td>
<td>enabled</td>
<td>personLdap</td>
<td>grouperLoaderLdapValueDef</td>
<td>accff...</td>
</tr>
<tr>
<td>Metadata on assignment</td>
<td>Grouper loader LDAP type</td>
<td>enabled</td>
<td>LDAP_SIMPLE</td>
<td>grouperLoaderLdapValueDef</td>
<td>52eaf...</td>
</tr>
</tbody>
</table>
Grouper Loader LDAP example (continued)

- You can debug the loader
- log4j.properties
- Run GSH: C:\grouper\bin> gsh
gsh 0% grouperSession = GrouperSession.startRootSession();
gsh 1% loaderGroup = GroupFinder.findByName(grouperSession, "test:testGroup");
gsh 2% loaderRunOneJob(loaderGroup);
Naming best practices

• Might want to have a top level folder for your institution, something short
  • E.g. at Penn, it is penn:
  • E.g. at Chicago, it is uc:
• This will make group names generally globally unique
• At Penn we also have a top level folder “test:”
• Our “test” grouper instance is for testing new upgrades to grouper, the “test” folder in prod is for clients’ test environments. Not for load
Naming best practices (continued)

• Folder structure matches the privilege delegation

• For instance, your top level folders (under the institution folder) might be schools and centers in the institution
Naming best practices (continued)

You can look for groups throughout the hierarchy. (You might not be able to see some groups if you lack appropriate privileges.)

Browse or list groups

Current location is:

Root: penn

- community
- etc
- evp
- gse
- isc
- library
- med
- pennlaw
- presidentsOffice
- sas
- seas
- uphs
- wharton
Naming best practices (continued)

- Keep groups / roles / permissions organized in separate folders
Naming best practices (continued)

- Keep groups / roles / permissions organized in separate folders
Naming best practices (continued)

- Enforce a policy on which characters are allowed
- Keep in mind down-stream systems
Naming best practices (continued)

• Could start with extensions that are the same as display extensions
  • Some people like spaces and title case instead of camel case
Naming best practices (continued)

• Have a high-level apps folder
  • Note: Penn doesn’t do this, though some institutions recommend it

• Have a high-level community folder
  • Commonly used groups generally from loader

• Descriptive extensions
  • Some screens only show the extension
  • Instead of “admins”, use “ptoAdmins”
Reference groups via loader

- Have a high-level community folder
  - Commonly used groups from loader
Reference groups via loader (continued)

Current location is:
- Root: penn: community

- alumni
- employee
- student
- activeNonAlumniWithPennname
- affiliate
- courtesyAffiliation
- employee
- employeeActiveFullTimePartTime
- employeeIncludingUphs
- employeeNonTemp
- employeeOrServiceProvider
- employeeServiceProvider
- faculty
- facultyStudentStaff
- lockout
- overseersTrustees
- religiousCommunities
- researchers
- serviceProviders
- staff
- student
- uphs
- visitingFaculty
- visitingScholar

Grouper is sponsored by

INTERNET2
Reference groups via loader (continued)

- Courses
- Could have include/exclude
- Could filter which courses are needed
- Each course should be a folder
- Course list, instructors, guests, etc
Reference groups via loader (continued)
Reference groups via loader (continued)

- Employee orgs similar to courses
- Should organize such that changes in org namespace do not affect group names (been burned)
Reference groups via loader (continued)
Reference groups via loader (continued)

• Employee orgs can have rollups based on descendant orgs
Reference groups via loader (continued)

Reference groups via loader (continued)
Reference groups via loader (continued)
Reference groups via loader (continued)

- Loader has 5 categories
  - SQL_SIMPLE
  - SQL_GROUP_LIST
  - LDAP_SIMPLE
  - LDAP_GROUP_LIST
  - LDAP_GROUPS_FROM_ATTRIBUTES
- See grouper loader wiki and intro images
Composite groups

• Three types of composites
  • Union
    • Never use this, just add group as member of another group which is more efficient
  • Intersection
    • Good for requiring members of a group to be members of another group
  • Minus
    • Good for excluding people from a group
Composite groups (continued)

- You can set these up
  - Manually
  - Via loader attributes
  - Via group attributes
Composite groups (continued)

• Composite include/exclude can delegate privileges well

• “System of record” groups is the group used prior to the composite calculation

• Composite groups do not remove the user from the system of record group
Composite groups (continued)

- Rules to the rescue
- Grouper rule can remove user from the system of record group when not employee
- When rehired, user will have to go back through the intake process
- Will not work with loader system of record (should *never* edit that!)
Permissions inheritance

• Penn uses permissions in several apps
• One (which is not quite live yet) is managing unix permissions
Permissions inheritance (continued)

- Support staff for applications have various permissions for various applications
- Restart tomcat
- Stop tomcat
- Start tomcat
- Status tomcat
- Apache configtest
- Apache graceful
- View logs
- Redeploy
Permissions inheritance (continued)

• Users are the unix users
• Role is clusterUser
• Permission is the application
• Action is tomcatRestart / apacheGraceful / etc

• Real time and batch provisioning
Permissions inheritance

- clusteralinux.jar sends XMPP message when relevant Grouper event occurs
- Full refresh kicked off by nightly cron or XMPP message
- Scripts use the permission file when commands are run

Permissions file:
- john can restart tomcat_a
- mary can run status on tomcat_c
- steve can start|restart|stop|status tomcat_d

Linux server that needs permission management on commands
Permissions inheritance (continued)

• Group inheritance
  • Could have a group of student-based applications support staff that all share the same permissions
Permissions inheritance (continued)

• Role inheritance
  • There could be a clusterAdminRole role that inherits everything that clusterRole has, and includes all actions on all applications
Permissions inheritance (continued)

• Action inheritance
  • “tomcatAll” action could include: tomcatStatus, tomcatRestart, tomcatStop, tomcatStart
  • “clusterAll” action could include all actions to give someone full control of app
Permissions inheritance (continued)

• Permission inheritance
  • Can make collections of applications so you can assign permissions to multiple related applications with one assignment
  • E.g. researchApplications could include the five permissions for the five research applications
Progression basic to production

• Start with the installer
• Do manual builds based on installer output
• Tweak some config settings, see changes
Progression basic to production (continued)

• Subject source
  • SQL or LDAP
  • Might have more flexibility with JDBC (make a view or data feed with whatever you want)
  • If everything you need is in JNDI, and you have a highly available env, use that
Progression basic to production (continued)

- Subject source
  - Subjects should “always” be resolvable
  - ID generally is an opaque unchanging permanent id
  - Identifier is a netId, eppn, something that needs to resolve to a subject
Progression basic to production (continued)

• Subject source
  • Description is what is generally shown on screen, at Penn:
  • Michael Christopher Hyzer (mchyzer, 10021368) (active) Staff - Isc Administrative Systems Tools And Technologies - Application Architect (also: Alumni)
Progression basic to production (continued)

- Customize the UI
- At least put a logo (media.properties)
Progression basic to production (continued)

• Customize the UI authentication
• Easy with shib, CAS, cosign
• Web server plugins will work with REMOTE_USER
• Can do a servlet filter with whatever authentication
Progression basic to production (continued)

• Look in media.properties, grouper.properties, grouper-loader.properties, see which settings you want to change
Progression basic to production (continued)

• Provision to LDAP / AD
  • PSP
  • Batch and real-time
Progression basic to production (continued)

- Document your Grouper deployment for your users
- Delegate privileges for high level folders as needed
- Train admins on using Grouper
- Integrate projects
Progression basic to production (continued)

- Decide which environments to have
  - Prod
  - Test
  - Dev?
  - Train?
- See which config settings are different for each environment
- Keep your settings in your revision control
- **Have a build script** to war up your builds
Manage Grouper in multiple environments

- Penn shared an ant build script
- Out of the box builds a dev / test / prod
Manage Grouper in multiple environments
Manage Grouper in multiple environments

- Config files have variables whose values are controlled by the build.properties
Manage Grouper in multiple environments

Here are the grouper.hibernate.properties variables

```java
hibernate.connection.url = @dbUrl@
hibernate.connection.username = @dbUser@
hibernate.connection.password = @dbPass@
```

There are entries per env. Note the passwords are encrypted with the morphString Internet2 library, so the encrypted values are in a file system file (better for storage of config files in CVS nad hiding plaintext passwords)

```java
devDbUrl=jdbc:oracle:thin:@devserver:1521:devsid
testDbUrl=jdbc:oracle:thin:@testserver:1521:testsid
prodDbUrl=jdbc:oracle:thin:@prodserver:1521:prodsid
devDbUser=myuser
testDbUser=myuser
prodDbUser=myuser

localdevDbPass=/home/appadmin/pass/grouper/grouperMorphDev.pass
devDbPass=/home/appadmin/pass/grouper/grouperMorphDev.pass
testDbPass=/home/appadmin/pass/grouper/grouperMorphTest.pass
prodDbPass=/home/appadmin/pass/grouper/grouperMorphProd.pass
```
Thanks!

Further information:

Infosheets, mail lists, wiki, downloads, etc: www.internet2.edu/grouper

Grouper demo server: https://grouperdemo.internet2.edu/