Central Authorization System
January 2009
Penn Groups

Penn Profile

- Private research university founded in 1740
- 259 buildings, 283 acres located in West Philadelphia
- 10,345 undergraduates; 12,103 graduate and professional students (as of Fall 2007) enrolled into twelve graduate/professional schools
- Over 20,000 employees, including 14,000+ in University Health System
- University (including health system) operating budget of four billion dollars
- Central IT in a decentralized environment
  - Twelve schools and multiple administrative centers operate with autonomy
  - Most schools and centers have their own IT department
  - Central IT provides university-wide applications and infrastructure
Identity Management at Penn

Goal: To increase protection of the confidential and sensitive information at Penn by:

- Uniquely identifying entities associated with Penn
- Providing access to appropriate facilities, services, and systems
- Preventing unauthorized access to facilities, services, and systems
Components of identity management

- Penn Community – central repository for a person’s bio/demo data as fed by core business systems (SRS, HR/Payroll, Atlas, UPHS) and entered directly for ancillary affiliates
- Penn Directory – system that holds the preferred name and contact info for all Penn affiliates
- Penn Card – system used to generate the physical ID card that is used for building access and commercial transactions across the university
- PennNames - system used to associate a unique username to each individual at Penn, providing a common and consistent University namespace for online services
- PennKey – unique identifier for Penn’s central authentication system; with associated password, provides an electronic means to authenticate an individual and provide access to systems across the university
- PennGroups – system for creating and managing groups to facilitate authorization decisions by applications with hooks to LDAP or web services
Penn’s Identity Management Strategy

Penn Community

Penn Directory

PennGroups

PennCard

PennKey

Ancillary Affiliates (Temp, VFAC, CHOP, etc.)

HR

SRS

UPHS

Atlas

In-House App

3rd Party App

AuthZ Decisions via LDAP or WS

Central Authorization at the University of Pennsylvania
PennGroups is derived from the Internet2 open source Grouper initiative

Has been adopted and deployed at many other universities (Brown, Cornell, Yale)

Penn has worked with the Grouper team to enhance the baseline product (UI, web services, SQL loaded groups)
  - Better meets the needs of Penn
  - Provides additional useful functionality to other grouper users
  - Allows Penn to benefit from future grouper enhancements without maintaining a separate source code instance
Benefits

- Facilitates consistent application of University business rules
  - Managed through a common UI and web services

- Streamlines maintenance of authorization data
  - Brings scattered redundant groups together for re-use
  - Allows useful actions on these groups -- group math, group nesting, exclusion criteria

- Leverages Penn Community data for accurate, up to date authorization decisions
  - Can leverage existing attribute information

- Distributed/delegated model of control
  - Supports the creation of new groups by schools and centers
Authorization by application

After authentication the application can interrogate PennGroups for access to group membership data

- Web services
- LDAP

Changes to group membership are reflected automatically and propagate to the application dynamically
Two modes for creating and managing groups

- Automated
  - Web services - build and run a query from your data store and send group membership information to PennGroups via the web service API
  - SQL loaded groups – Configure a SQL query within the PennGroups UI to run on a scheduled basis to modify group membership

- Manual
  - UI – log onto the PennGroups UI to manually manage your group membership
    - You cannot manually add members to or remove members from a group that is managed in an automated fashion
    - You can simulate this with include/exclude composite groups
PennGroups Hierarchy
PennGroups in a Decentralized Environment

- When School/Center is purchasing or developing a new system
  - LSP (local support provider)/application developer contacts Central IT
  - LSP/developer and Central IT collaborate to:
    - Establish authorization use cases for the specific application
    - Determine access method (LDAP or Web Services)
    - Determine best approach for group creation and maintenance
  - School/Center fills out access forms
  - Central IT consults with LSP/developer on group hierarchy structure
Use Cases

- **PTO – Paid Time Off**
  - Self service system used to request/track vacation/sick time
  - Penn Groups provides the flexibility so that the user selects their approver for time off.
  - Time off can be routed and approved by other than a direct supervisor

- **Warehouse Apps**
  - Penn groups provides a feed for org based security based on active status

- **Abramson's Cancer Center**
  - Builds custom research related applications and needs a means to confirm that users who log in currently have an active status

- **School of Engineering and Applied Science**
  - Affiliate level groups - faculty members, staff members, students, undergrads, grads, PhD students
  - Class level groups - everyone enrolled in every SEAS course, and several ad-hoc groups.
  - Kept up to date via a SEAS data store and propagated to PennGroups via the SQL loader
  - Group hierarchy (groups such as freshman, sophomore, etc are members in the group uGrad).
  - Ad hoc groups generated and maintained via specific applications and business rules.
  - Use of groups to determine access to various resources such as SSH (with different groups allowed to access different machines), IMAP, POP, SMTP, etc.
Penn Groups

PennGroups

Technical Discussion
Penn Groups

Agenda (note: additional information in slide notes)

- PennGroups architecture
- User interfaces
- Web services
- LDAP
- Grouper client
- Grouper loader
- What’s new with Grouper in 1.4
  - Configuration checking
  - Daily report
Penn Groups

PennGroups architecture
Penn Groups

Grouper user interface

- Grouper has a built in user interface
- Penn generally uses the default UI, though:
  - We customized the authentication to use Penn’s single signon
  - We added custom code to require users to be in a grouper group to be able to log in (not everyone allowed)
- Penn did a facelift for the Grouper 1.3 release in Spring 2008, improving the usability and help documentation
- For Grouper 1.4 in January 2009, we added the ability to have tooltips on types and attributes
Penn Groups

Grouper user interface (continued)

- Tooltips configured in nav.properties

```properties
# prefixes for messages
message.Message=Note:
massage.ErrorMessage=Error:
massage.WarningMessage=Warning:

tooltipTargetted.groupTypes.grouperLoader=Group membership automatically managed
tooltipTargetted.groupFields.grouperLoaderDbName=For sql based loader, this is
tooltipTargetted.groupFields.grouperLoaderIntervalSeconds=If a START_TO_START:
tooltipTargetted.groupFields.grouperLoaderPriority=The loader has a max number
tooltipTargetted.groupFields.grouperLoaderQuartzCron=Quartz cron-like string (:)
tooltipTargetted.groupFields.grouperLoaderQuery=This is the query to run in the
tooltipTargetted.groupFields.grouperLoaderScheduleType=CRON: This is a cron-like
tooltipTargetted.groupFields.grouperLoaderType=SQL_SIMPLE: a group whose membe
tooltipTargetted.groupFields.grouperLoaderAndGroups=If you want to restrict mem
tooltipTargetted.groupFields.grouperLoaderGroupTypes=If you want types assigned
tooltipTargetted.groupFields.grouperLoaderGroupsLike=If you want the group (if
```
For PennGroups tasks not included in Grouper, we have an ancillary UI for Grouper.

Diagram:
- Browser
- Grouper admins, staff, students
- End user
- Server
- Grouper client
- Web services
- LDAP
- Grouper Registry

Two way data flow
One way data flow
Currently we only have one task, registering an LDAP login.
Penn/Internet2 spent a lot of effort in winter/spring 2008 to help create the Grouper web services

- They can be REST or SOAP
- They can be simple “Lite” calls, or batched
- REST accepts formats: XML, XHTML, JSON, HTTP params

There are a dozen operations exposed, including managing:
  - Groups
  - Memberships
  - Permissions
  - Folders

Penn uses HTTP credentials sent to kerberos and penn:etc:webServiceUsers group required for authorization
Grouper web services (continued)
There are hundreds of samples to manage

Custom sample generator is a harness which runs all samples, and stores them in CVS:

- Listens on TCP port, forwards to web service
- Makes web service request to the listener
- Captures request and response
  - Indents the XML or JSON
  - Masks sensitive data (e.g. authentication credentials)
- Captures stdout and stderr
- Collates everything including source of sample, saves file in CVS
- Runs each sample for all different formats, web service types, etc.
- 163 total sample files
### Index of /grouper-ws/grouper-ws/doc/samples/addMember

Files shown: 13  
(Show 17 dead files)

<table>
<thead>
<tr>
<th>File</th>
<th>Rev.</th>
<th>Age</th>
<th>Author</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>WsSampleAddMemberLite_soap.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite2_withInput_http_xhtml.txt</td>
<td>1.8</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite2_withInput_json.txt</td>
<td>1.8</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite2_withInput_xhtml.txt</td>
<td>1.8</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite2_withInput_xml.txt</td>
<td>1.8</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite_http_xhtml.txt</td>
<td>1.9</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRestLite_json.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRest_xhtml.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRest_xml.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberLite_json.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRest_xhtml.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMemberRest_xml.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
<tr>
<td>WsSampleAddMember_soap.txt</td>
<td>1.10</td>
<td>5 days</td>
<td>mchyzer</td>
<td></td>
</tr>
</tbody>
</table>
Grouper web services (continued)

Grouper web service sample of service: addMember, WsSampleAddMemberLite, code generated classes,

```
POST /grouperWs/services/GrouperService HTTP/1.1
Content-Type: application/soap+xml; charset=UTF-8; action="urn:addMemberLite"
User-Agent: Axis2
Authorization: Basic xxxxxxxxxxxxxxxx==
Host: localhost:8092
Transfer-Encoding: chunked

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:addMemberLite xmlns:ns1="http://soap.ws.grouper.middleware.internet2.edu/xsd">
      <ns1:clientVersion>v1_4_000</ns1:clientVersion>
      <ns1:groupName>aTest:aGroup</ns1:groupName>
      <ns1:groupId>/ns1:groupId
```
There is a Grouper LDAP provisioning connector called LDAPPC, though Penn does not use this.

We have some simple triggers in Oracle which add records to a change log.

Then a process pulls records off of that table to send diffs to openLDAP (runs every 10 minutes).

Daily all records are refreshed.

Only users in penn:etc:ldapUsers can login to ldap.

Users can only read group membership lists they have privileges to read in Grouper.
LDAP and web services are low level

Grouper client exposes Grouper LDAP and web services to a command line API or a Java library

It can also be used to generate custom web service samples (can log requests and responses)

Institutions can customize the client before distributing so the LDAP config is done (e.g. Penn allows ID lookups)

Callers aren’t tied to output, they can tell the client the output format that is expected
Grouper client (continued)
Sample LDAP config:

```java
ldapSearchAttribute.operationName.2 = hasMemberLdap
ldapSearchAttribute.ldapName.2 = ou=groups
ldapSearchAttribute.matchingAttributes.2 = cn, hasMember
ldapSearchAttribute.matchingAttributeLabels.2 = groupName, pennnameToCheck
ldapSearchAttribute.returningAttributes.2 = cn
ldapSearchAttribute.outputTemplate.2 = hasMember: ${resultBoolean}
ldapSearchAttribute.resultType.2 = BOOLEAN
```

Sample LDAP command line call:

c:\grouper> java -jar grouperClient.jar --operation=hasMemberLdap
--groupName=penn:myfolder:mygroup --pennnameToCheck=jsmith

hasMember: true
Sample command line web service call:
```bash
c:\grouper> java -jar grouperClient.jar --operation=getMembersWs
   --groupNames=aStem:aGroup --outputTemplate=${index}: ${subject.id}
```

0: 12345
1: 23456

c:\grouper>

Sample Java web service call:
```java
WsAddMemberResults wsAddMemberResults =
   new GcAddMember().assignGroupName("aStem:aGroup")
   .addSubjectId("12345").execute();
```
Penn contributed the “Grouper loader” in spring 2008

This keeps groups in sync with results of sql queries
SQL> select * from authz_employee_active_v where rownum < 10

<table>
<thead>
<tr>
<th>PENN_ID</th>
<th>PENN_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>jsmith</td>
</tr>
<tr>
<td>12346</td>
<td>asmith</td>
</tr>
<tr>
<td>12347</td>
<td>bsmith</td>
</tr>
<tr>
<td>12348</td>
<td>rjohnson</td>
</tr>
<tr>
<td>12349</td>
<td>sjohnson</td>
</tr>
<tr>
<td>12350</td>
<td>tjohnson</td>
</tr>
<tr>
<td>12351</td>
<td>ajones</td>
</tr>
<tr>
<td>12352</td>
<td>bjones</td>
</tr>
<tr>
<td>12353</td>
<td>cjones</td>
</tr>
</tbody>
</table>
**Current location is:**

- Root: ⌂
- penn: ⌂
- community: ⌂
- employee: ⌂
- orgGroups

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>orgGroups</td>
</tr>
<tr>
<td>Description</td>
<td>dynamic group with configs for org groups (do not add members to this)</td>
</tr>
<tr>
<td>ID</td>
<td>orgGroups</td>
</tr>
<tr>
<td>ID_Path</td>
<td>penn:community:employee:orgGroups</td>
</tr>
<tr>
<td>UUID</td>
<td>6f75f636-ad7c-4d50-a02f-c779b94c4aa8</td>
</tr>
<tr>
<td>Types</td>
<td></td>
</tr>
<tr>
<td>grouperLoader</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderAndGroups</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderDbName</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderGroupQuery</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderGroupQueryTypes</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderGroupsLike</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderIntervalSeconds</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderPriority</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderQuartzCron</td>
<td></td>
</tr>
<tr>
<td>grouperLoaderQuery</td>
<td>select subject_id, group_name from EMPLOYEE_ORG_GROUPS_V</td>
</tr>
<tr>
<td></td>
<td>CRON</td>
</tr>
<tr>
<td>grouperLoaderScheduleType</td>
<td>SQL_GROUP_LIST</td>
</tr>
<tr>
<td>grouperLoaderType</td>
<td>SQL_GROUP_LIST</td>
</tr>
<tr>
<td>SUBJECT_ID</td>
<td>GROUP_NAME</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>12345</td>
<td>penn:community:employee:orgs:employeeOrg123</td>
</tr>
<tr>
<td>12346</td>
<td>penn:community:employee:orgs:employeeOrg123</td>
</tr>
<tr>
<td>12347</td>
<td>penn:community:employee:orgs:employeeOrg123</td>
</tr>
<tr>
<td>12348</td>
<td>penn:community:employee:orgs:employeeOrg124</td>
</tr>
<tr>
<td>12349</td>
<td>penn:community:employee:orgs:employeeOrg124</td>
</tr>
<tr>
<td>12350</td>
<td>penn:community:employee:orgs:employeeOrg124</td>
</tr>
<tr>
<td>12351</td>
<td>penn:community:employee:orgs:employeeOrg128</td>
</tr>
<tr>
<td>12352</td>
<td>penn:community:employee:orgs:employeeOrg128</td>
</tr>
<tr>
<td>12353</td>
<td>penn:community:employee:orgs:employeeOrg128</td>
</tr>
</tbody>
</table>
If grouper is not configured correctly, it sometimes did not give descriptive errors

With 1.4, on startup, it will verify its configuration and give descriptive errors

It checks:

- All DBs connectivity
- Config file validity (including data types)
- Subject API queries
- System groups exist (auto-create)
Print out useful grouper info on startup

Grouper starting up: version: 1.4.0, build date: 11/2/2008, env: DEV
grouper.properties read from: C:\grouper\build\grouper.properties
Grouper current directory is: C:\grouper
log4j.properties read from: C:\grouper\build\log4j.properties
Grouper is logging to file: console, at min level WARN for package: edu.internet2.middleware.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\grouper\grouper.hibernate.properties
grouper.hibernate.properties: jdbc:mysql://localhost:3306/grouper
sources.xml read from: C:\grouper\build\sources.xml
sources.xml jdbc source id: penpperson: GrouperJdbcConnectionProvider
sources.xml groupersource id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider
Grouper daily report

- With Grouper 1.4 there is a daily report
- This is emailed out every morning to grouper admins
- Includes a state of the registry:
  - E.g. number of new / total groups and memberships
- Loader job reports
  - Number of successes and failures, inserts/updates/deletes
- Registry health
  - Unresolvable subjects, bad memberships
- Stores history of reports on file system
Subject: Grouper report

OVERALL:

environment: PROD
memberships: 135,280
groups: 20
members: 56,207
folders: 17
unresolvable subjects: 1,197
bad memberships: 0

----------

WITHIN LAST DAY:

new memberships: 66
new groups: 0
updated groups: 0
new folders: 0
Grouper binary distribution

- Grouper used to be distributed as source that needed to be built with ant and a java compiler
- Now with grouper 1.4 there is a binary build which is the java libraries
- All that is required is a java runtime
- An HSQL database is included, you can unzip, init the db, and run grouper shell (GSH)
- There is also a grouper client binary distribution
[mchyzer@ellis temp]$ tar xzf grouper.binary.1.4.0.tar.gz

[mchyzer@ellis bin]$ ./gsh.sh -registry -runscript

Grouper starting up: version: 1.4.0...

Are you sure you want to schemaexport db user 'sa', db url 'jdbc:hsqldb:/temp/.../grouper;create=true'? (y|n):

Y

Continuing...

Script was executed successfully

[mchyzer@ellis bin]$ ./gsh.sh

Grouper starting up: version: 1.4.0...

Type help() for instructions

**gsh 1%** addRootStem("myschool", "myschool");

stem: name='myschool' displayName='myschool' uuid='abcde'

**gsh 2%** addGroup("myschool", "agroup", "agroup");

group: name='myschool:agroup' displayName='myschool:agroup' uuid='abcdf'
Grouper encrypted passwords

- Grouper database passwords can now be encrypted and stored in external files to the normal config files
  - Grouper / loader DB’s
  - Subject API
  - Grouper client LDAP and web service

- There is a stand-alone Internet2 library: morphString.jar (can easily be reused in other projects)

- Facilitates:
  - Non-cleartext passwords
  - Sanitized config files (for email or source control)
**Grouper hooks**

- Grouper 1.4 has 100 hook points built in to the data layer API.
- You can get the data to do something (notification), add more queries to the transaction (audit), or veto the transaction.
- Currently Grouper ships with some default implementations of hooks:
  - Group name and attribute validator regex (e.g. alphanumeric)
  - Group type edit security (e.g. only let admins edit grouper loader attributes)
  - Include/exclude auto-create
  - Require groups auto-create
For technical documentation see the Internet2 Grouper wiki at:

- Grouper product
  - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Project
- Grouper project
  - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Project
- Web services info
  - https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+-+Web+Services
Grouper DDL management

- Grouper used to use Hibernate schemaexport
- Switched to a custom method built on Jakarta ddlutils
- Supports hsql, oracle, mysql, and postgres (and probably other untested db’s)
- Supports tables, views, comments, indices, foreign keys, data massaging
- Knows when the database is out of sync (keeps state in DB table), and logs to ERROR that update needed
- If you drop a column of a table, and run “deep” ddl registry check, it will generate DDL to recreate it
### Grouper DDL management (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>PARENT_STEM</th>
<th>CREATOR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>711a6173-2b45-4c5d-a814-e45669c13793</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>644e26b0-0295-4b54-b7e4-6e09b1e7716e</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>54e35f33-ab76-4894-9372-f807c1972a29</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>937a8933-e457-4039-8033-cb30eb3cc2d</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>7fe9678c-e92b-48a8-91ab-82d02220d38e</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>7f2f0d0f-eeb4-413b-a4e-a113-5503a6b5263</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>4813a3b3-7647-4956-a81d-8fa03cd5a3d</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>53fe910a-057e-4291-9384-44a450a238ce</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>60a954ac-feb1-46b3-bf6e-9ef03eefeb09947</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>740f2098-0215-4756-9f8f-f2563d7f20f</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>04c22387-e5a1-4858-9765-d7e0fdca256cf</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>ada0dc1f-7a3c-4e93-a1eb-bf178002d6b</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>913f95d0-cbe4-4bce-a1a6-15428810cc66</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>e81d3421-05f0-d482-b0d9-55f6f113990</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
<tr>
<td>093b398e-2cd4-46b0-89e5-0e78062d38c</td>
<td>42891ef7-74e6-4cbd-987e-451620b25988</td>
<td>540f6f10-784c-4396-a0d</td>
</tr>
</tbody>
</table>
### Grouper DDL management (continued)

#### Table: Grouper_GROUPS_V1

<table>
<thead>
<tr>
<th>Column Name</th>
<th>ID</th>
<th>Data Type</th>
<th>Null?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTENSION</td>
<td>1</td>
<td>VARCHAR2 (1024 Byte)</td>
<td>Y</td>
<td>EXTENSION: part of group name not including path information, e.g. theGroup</td>
</tr>
<tr>
<td>NAME</td>
<td>2</td>
<td>VARCHAR2 (1024 Byte)</td>
<td>Y</td>
<td>NAME: name of the group, e.g. school:stem:theGroup</td>
</tr>
<tr>
<td>DISPLAY_EXTENSION</td>
<td>3</td>
<td>VARCHAR2 (1024 Byte)</td>
<td>Y</td>
<td>DISPLAY_EXTENSION: name for display of the group, e.g. My school:The stem1:</td>
</tr>
<tr>
<td>DISPLAY_NAME</td>
<td>4</td>
<td>VARCHAR2 (1024 Byte)</td>
<td>Y</td>
<td>DISPLAY_NAME: name for display of the group without any path information, e.g.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>5</td>
<td>VARCHAR2 (1024 Byte)</td>
<td>Y</td>
<td>DESCRIPTION: contains user entered information about the group e.g. why it exists</td>
</tr>
<tr>
<td>PARENT_STEM_NAME</td>
<td>6</td>
<td>VARCHAR2 (255 Byte)</td>
<td>N</td>
<td>PARENT_STEM_NAME: name of the stem this group is in, e.g. school:stem1</td>
</tr>
<tr>
<td>GROUP_ID</td>
<td>7</td>
<td>VARCHAR2 (128 Byte)</td>
<td>N</td>
<td>GROUP_ID: uuid unique id of the group</td>
</tr>
<tr>
<td>PARENT_STEM_ID</td>
<td>8</td>
<td>VARCHAR2 (128 Byte)</td>
<td>N</td>
<td>PARENT_STEM_ID: uuid unique id of the stem this group is in</td>
</tr>
<tr>
<td>MODIFIER_SOURCE</td>
<td>9</td>
<td>VARCHAR2 (255 Byte)</td>
<td>Y</td>
<td>MODIFIER_SOURCE: source name of the subject who last modified this group, e.g.</td>
</tr>
<tr>
<td>MODIFIER_SUBJECT_ID</td>
<td>10</td>
<td>VARCHAR2 (255 Byte)</td>
<td>Y</td>
<td>MODIFIER_SUBJECT_ID: subject id of the subject who last modified this group, e.g.</td>
</tr>
<tr>
<td>CREATOR_SOURCE</td>
<td>11</td>
<td>VARCHAR2 (255 Byte)</td>
<td>Y</td>
<td>CREATOR_SOURCE: source name of the subject who created this group, e.g. school:stem1:</td>
</tr>
<tr>
<td>CREATOR_SUBJECT_ID</td>
<td>12</td>
<td>VARCHAR2 (255 Byte)</td>
<td>Y</td>
<td>CREATOR_SUBJECT_ID: subject id of the subject who created this group, e.g. 12:</td>
</tr>
<tr>
<td>IS_COMPOSITE_OWNER</td>
<td>13</td>
<td>CHAR (1 Byte)</td>
<td>Y</td>
<td>IS_COMPOSITE_OWNER: T if this is a result of a composite operation (union, intersection)</td>
</tr>
<tr>
<td>IS_COMPOSITE_FACTOR</td>
<td>14</td>
<td>CHAR (1 Byte)</td>
<td>Y</td>
<td>IS_COMPOSITE_FACTOR: T if this is a member of a composite operation, e.g. one</td>
</tr>
<tr>
<td>CREATOR_ID</td>
<td>15</td>
<td>VARCHAR2 (128 Byte)</td>
<td>N</td>
<td>CREATOR_ID: member id of the subject who created this group, foreign key to group</td>
</tr>
<tr>
<td>CREATE_TIME</td>
<td>16</td>
<td>NUMBER (38)</td>
<td>N</td>
<td>CREATE_TIME: number of milis since 1970 since this group was created</td>
</tr>
<tr>
<td>MODIFIER_ID</td>
<td>17</td>
<td>VARCHAR2 (128 Byte)</td>
<td>Y</td>
<td>MODIFIER_ID: member id of the subject who last modified this group, foreign key</td>
</tr>
<tr>
<td>MODIFY_TIME</td>
<td>18</td>
<td>NUMBER (38)</td>
<td>Y</td>
<td>MODIFY_TIME: number of milis since 1970 since this group was last changed</td>
</tr>
<tr>
<td>HIBERNATE_VERSION_NUMBER</td>
<td>19</td>
<td>NUMBER (38)</td>
<td>Y</td>
<td>HIBERNATE_VERSION_NUMBER: increments by 1 for each update</td>
</tr>
</tbody>
</table>

Contains one record for each group, with friendly names for some attributes and some more information
For the upgrade to Grouper 1.4, we removed some duplicate UUID’s and normalized some tables

- Backups for columns are kept
- Columns are dropped
- SQL to update other cols
- All generated in a DB independent way
- Though can also grouper-export and import in new registry
Some versions of mysql cannot accept indices on cols longer than 1000 bytes

Grouper can accommodate this (even though Jakartaddlutils cannot)

// see if we have a custom script here, do this since some versions of mysql
// cant handle indexes on columns that large
String scriptOverride = ddlVersionBean.isMysql() ? "\nCREATE INDEX attribute_value_idx " + "ON grouper_attributes (value(333));\n" : null;

GrouperDdlUtils.ddlutilsFindOrCreateIndex(database, ddlVersionBean, attributeTable.getName(), "attribute_value_idx", scriptOverride, false, "value");