Grouper and the CMU Billing Use Case

Grouper Solution to the CMU Billing Use Case

To summarize, this is related to permissions for if someone can read someone else's bill. Here are the basic rules:

1. University wide admins can read all bills
2. Students can read their own bill
3. Students can delegate access to read their bill to someone else
4. Local billing administrators can read bills for students in their organization

Command line CMU billing decision assigner and maker

This program (one java class, several hundred lines of code), used with the grouperClient (rest web service client jar), demonstrates how Grouper can fulfill these access management requirements remotely using web services. Just fill in the web service credentials in the grouper.client.properties file, and it is already pointed to the Grouper demo server where all this test data lives. Note, the client only have to make max two web service calls for decision making: one to get all permissions for a user for the applicable roles, and if not able to read bills, then another one to see which orgs the student is in. All the decisions can be made from the results of those two calls.

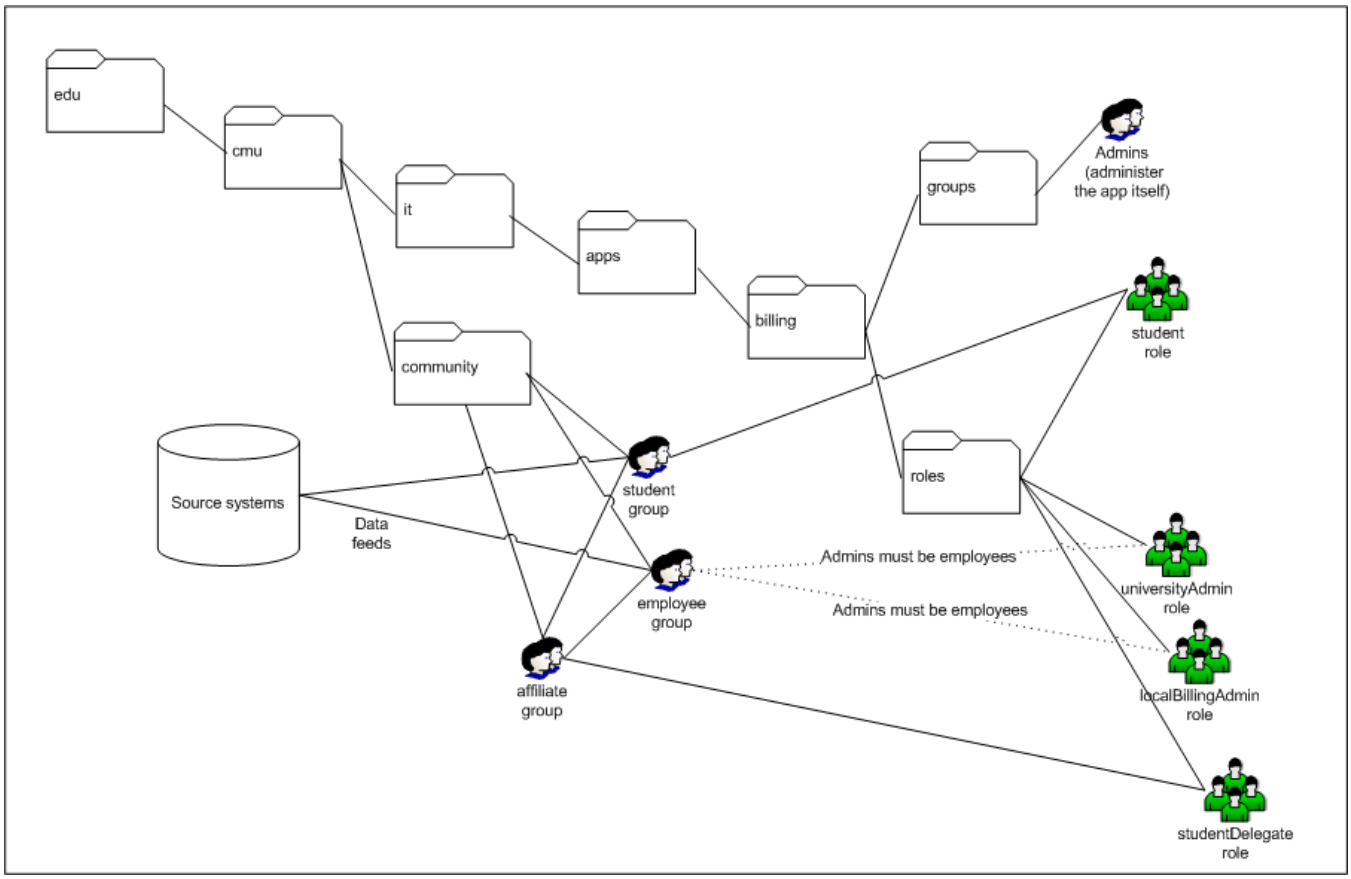
Download here (source in cmuBilling.jar file)

```
C:\temp\cmu>dir

10/02/2010  11:55 PM            14,680 cmuBilling.jar
10/02/2010  12:52 PM            17,532 grouper.client.properties
06/01/2010  06:12 AM         2,619,667 grouperClient.jar

C:\temp\cmu>

########### No permissions

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=babl --personWithAccess=elbl

Has allBills permission? false
Is checking own bill? false
Has studentDelegate permission? false
Person is not local admin on any orgs
Can read bill? false

########### Assign university-wide permission

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=assignUniversityAdmin --personWithAccess=fibl

Assign university admin: SUCCESS_ALREADY_EXISTED


########### University-wide permissions

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=babl --personWithAccess=fibl

Has allBills permission? true
Can read bill? true

########### Checking own bill failure

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=haed --personWithAccess=haed

Has allBills permission? false
Is checking own bill? true
Has checkOwnBill permission? false
Has studentDelegate permission? false
Person is not local admin on any orgs
Can read bill? false

########### Checking own bill success
```

```
C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=babu --personWithAccess=babu

Has allBills permission? false
Is checking own bill? true
Has checkOwnBill permission? true
Can read bill? true

########### Assign bill in org (hierarchy)

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=assignLocalAdmin --personWithAccess=hato --orgName=edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
BIOB

Assign local admin role: SUCCESS_ALREADY_EXISTED
Assign org edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB, changed? T

########### Check bill for local admin in org (hierarchy),
########### student kebr is a major in org 0103, which is a grandchild of the org BIOB
########### hato can read all of org BIOB (which includes 0103 obviously)

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=kebr --personWithAccess=hato

Has allBills permission? false
Is checking own bill? false
Has studentDelegate permission? false
Person is local admin on orgs: 0103, 0105, 0333, BIOB, BIOL, BIOT
Student has majors: 0103
Can read bill? true

########### Delegate to someone else

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=assignDelegate --personWithAccess=babl --studentId=stto

Has studentDelegate permission? false
Already is delegate? false
Assign student delegate role: SUCCESS
Already had permission: studentDelegate: false
Assigned permission studentDelegate
Assigned delegate for student: changed? T, delegateId changed: T

Or, run it again:

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=assignDelegate --personWithAccess=babl --studentId=stto

Has studentDelegate permission? true
Person has been assigned delegate from: stto
Already is delegate? true

########### Check if delegate to someone else

C:\temp\cmu>java -cp cmuBilling.jar;grouperClient.jar edu.cmu.it.apps.billing.CmuBilling --
operation=canReadBill --studentToCheck=stto --personWithAccess=babl

Has allBills permission? false
Is checking own bill? false
Has studentDelegate permission? true
Person has been assigned delegate from: stto
Can read bill? true

C:\temp\cmu>
```

Namespace

Start with the folder structure, groups, roles, etc



First create a section of the folder namespace for this application, e.g.

edu:cmu:it:apps:billing

GSH (note, you could do this with UI/WS too):

```
gsh 0% grouperSession = GrouperSession.startRootSession();
gsh 1% billingFolder = new StemSave(grouperSession).assignName("edu:cmu:it:apps:billing").
assignCreateParentStemsIfNotExist(true).save();
```

Grant that folder to the development team for this application, and for the service principal the application uses to access the deployment of Grouper. I recommend making an admin group for this application, and adding that group to the the proper privilege lists. e.g. edu:cmu:it:apps:billing:groups:admins

GSH (note, you could do this with UI/WS too):

```
gsh 2% billingAdmins = new GroupSave(grouperSession).assignName("edu:cmu:it:apps:billing:groups:admins").
assignCreateParentStemsIfNotExist(true).save();
gsh 3% billingFolder.grantPriv(billingAdmins.toSubject(), NamingPrivilege.STEM);
gsh 4% billingFolder.grantPriv(billingAdmins.toSubject(), NamingPrivilege.CREATE);
gsh 5% tgd = SubjectFinder.findById("tgd", true);
gsh 6% billingAdmins.addMember(tgd);
```

Roles

The application would need the following roles:

- edu:cmu:it:apps:billing:roles:universityBillingAdministrator
- edu:cmu:it:apps:billing:roles:student
- edu:cmu:it:apps:billing:roles:studentDelegate
- edu:cmu:it:apps:billing:roles:localBillingAdministrator

GSH (note, you could do this with UI/WS too, though you might need GSH to switch types to Role)

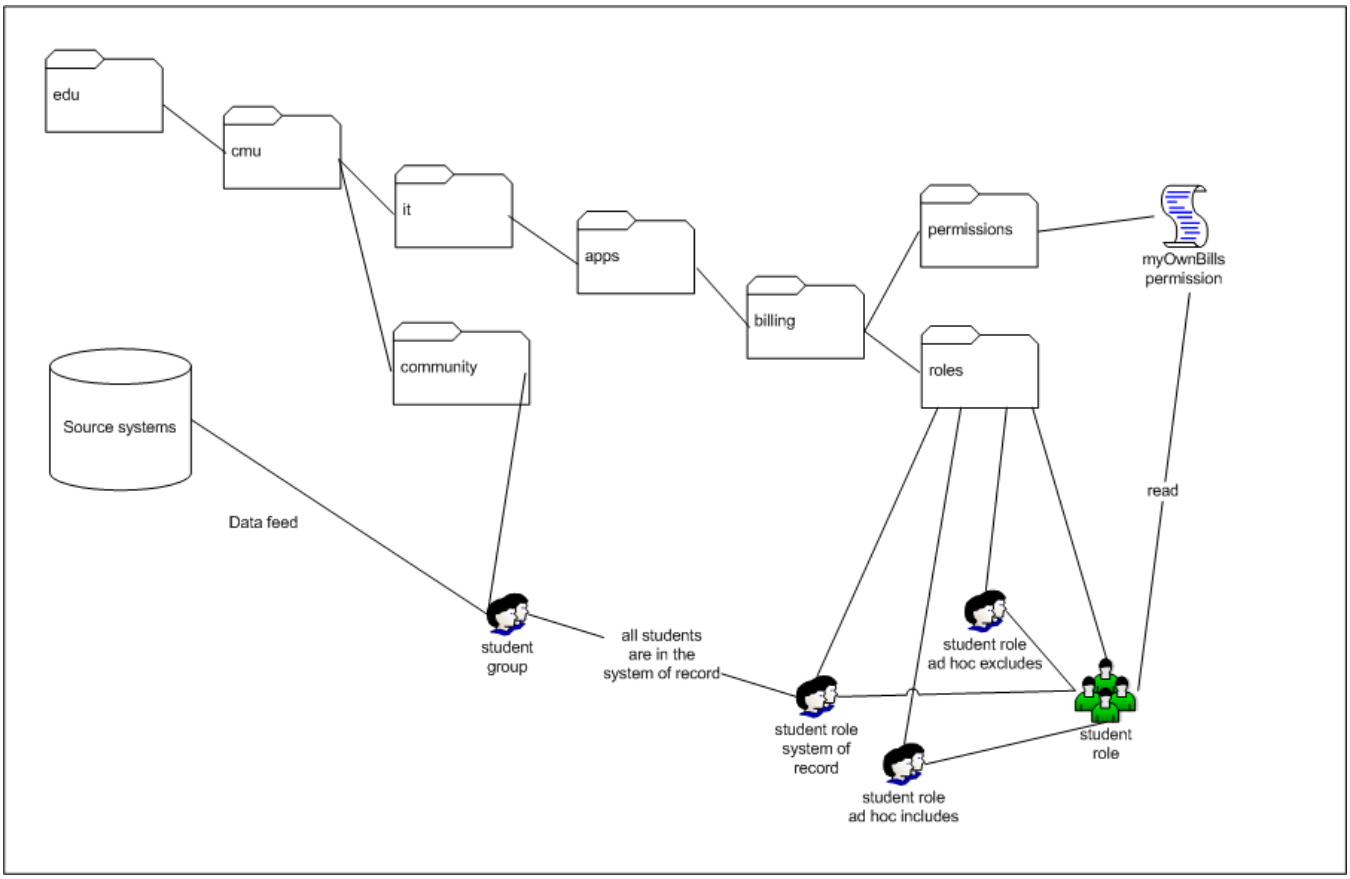
```
gsh 7% universityBillingAdministratorRole = new GroupSave(grouperSession).assignName("edu:cmu:it:apps:billing:
roles:universityBillingAdministrator").assignSaveMode(SaveMode.INSERT_OR_UPDATE).
assignCreateParentStemsIfNotExist(true).assignTypeOfGroup(TypeOfGroup.role).save();
gsh 8% studentRole = new GroupSave(grouperSession).assignName("edu:cmu:it:apps:billing:roles:student").
assignSaveMode(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).assignTypeOfGroup(TypeOfGroup.
role).save();
gsh 9% studentDelegateRole = new GroupSave(grouperSession).assignName("edu:cmu:it:apps:billing:roles:
studentDelegate").assignSaveMode(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).
assignTypeOfGroup(TypeOfGroup.role).save();
gsh 10% localBillingAdministratorRole = new GroupSave(grouperSession).assignName("edu:cmu:it:apps:billing:roles:
localBillingAdministrator").assignSaveMode(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).
assignTypeOfGroup(TypeOfGroup.role).save();
```

Note that security for who can read those roles, or assign, would be given to the group edu.cmu.it.apps.billing.groups.admins.

GSH (note, you could do this with UI/WS too):

```
gsh 11% universityBillingAdministratorRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.READ);
gsh 12% universityBillingAdministratorRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.UPDATE);
gsh 13% studentRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.READ);
gsh 14% studentRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.UPDATE);
gsh 15% studentDelegateRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.READ);
gsh 16% studentDelegateRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.UPDATE);
gsh 17% localBillingAdministratorRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.READ);
gsh 18% localBillingAdministratorRole.grantPriv(billingAdmins.toSubject(), AccessPrivilege.UPDATE);
```

You might assign the student role to the cmu overall student group (e.g. edu:cmu:community:students). Then as new students are provisioned into that group from the student system, they will automatically be assigned the role in this application. And as students fall out of the overall students group, they will lose the role in this application. If there are exceptions, you can make a group that is Grouper includeExclude so that you can have additions or restrictions from the system of record group. The students group could be provisioned via the Grouper Loader, or other means.



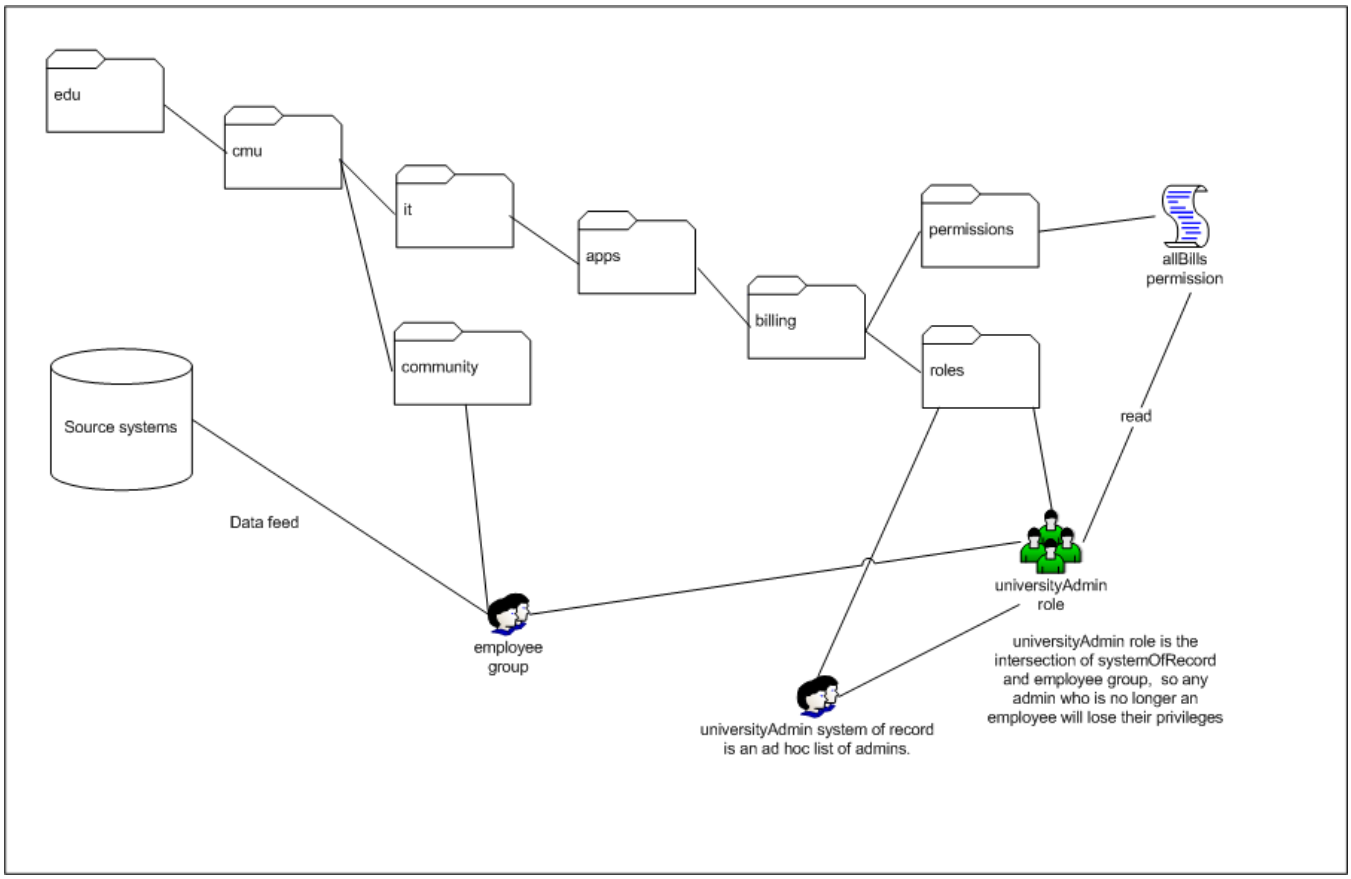
GSH / SQL (note, the Grouper commands can be done with the UI / WS / client). Create the students group, synced from a source system (simulated by using a static sql table)

```
Create a table for students (note, in reality this would be a view)

CREATE TABLE cmu_student (
  student_id varchar(50) NOT NULL,
  PRIMARY KEY  (student_id)
);

Populate with some students:

insert into cmu_student (student_id) values ('babl');
insert into cmu_student (student_id) values ('mchyzer');
insert into cmu_student (student_id) values ('babr');
insert into cmu_student (student_id) values ('babu');
commit;

Create a student group:

gsh 18% studentsGroup = new GroupSave(grouperSession).assignName("edu:cmu:community:students").assignSaveMode
(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).save();

Add a loader job to auto-populate students:

gsh 19% groupAddType("edu:cmu:community:students", "grouperLoader");
gsh 20% setGroupAttr("edu:cmu:community:students", "grouperLoaderType", "SQL_SIMPLE");
gsh 21% setGroupAttr("edu:cmu:community:students", "grouperLoaderDbName", "grouper");
gsh 22% setGroupAttr("edu:cmu:community:students", "grouperLoaderScheduleType", "CRON");
gsh 23% setGroupAttr("edu:cmu:community:students", "grouperLoaderQuartzCron", "0 0 7 * * ?");
gsh 24% setGroupAttr("edu:cmu:community:students", "grouperLoaderQuery", "SELECT student_id AS subject_id,
'jdbc' AS subject_source_id FROM cmu_student");

Run the loader one time to init:

gsh 25% studentsGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:students", true);
gsh 26% loaderRunOneJob(studentsGroup);

Kick off the job and run loader automatically so it runs daily:

[appadmin@i2midev1 bin]$ cd /opt/grouper/1.6.1/grouper.apiBinary-1.6.1/bin
[appadmin@i2midev1 bin]$ ./gsh.sh -loader > /tmp/grouper.1.6.1_loader.log 2>&1 &

See that the group has the people it should:

gsh 27% studentsGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:students", true);

gsh 28% studentsGroup.getMembers();
member: id='babl' type='person' source='jdbc' uuid='39ce9502bf2d4924bd920611b31832b3'
member: id='babr' type='person' source='jdbc' uuid='4117432df1244536bbc49562b988d7f8'
member: id='babu' type='person' source='jdbc' uuid='db2ec9441c2f42d3b42fcaf8f56d6217'
member: id='mchyzer' type='person' source='jdbc' uuid='b00465a3015547388caa3b422ea0c659'
```

Make the studentRole an includeExclude, and add the student group to the system of record part. This means that generally students in the students group have the role, though you could ad hoc add or subtract from that withotu affecting the loaded group. This can also be done with the UI, WS, or client

```
gsh 5% groupAddType("edu:cmu:it:apps:billing:roles:student", "addIncludeExclude");
gsh 6% addMember("edu:cmu:it:apps:billing:roles:student_systemOfRecord", "edu:cmu:community:students");
```

For the university wide billing administrator and the local billing administrator roles, you should probably create a Grouper restrictGroup where you look at what the person must be to have the role. If what they must be is an active staff member, then setup the restrictGroup so that if the user ever stops being an activeStaff member, they automatically fall out of the university or local billing administrator role. If it is wider, then make it an active CMU member, etc.



GSH / SQL (note, the Grouper commands can be done with the UI / WS / client). Create the employees group, synced from a source system (simulated by using a static sql table)

```
Create a table for employees (note, in reality this would be a view)

CREATE TABLE cmu_employee (
  employee_id varchar(50) NOT NULL,
  PRIMARY KEY  (employee_id)
);

Populate with some employees:

insert into cmu_employee (employee_id) values ('elbl');
insert into cmu_employee (employee_id) values ('dousti');
insert into cmu_employee (employee_id) values ('elbr');
insert into cmu_employee (employee_id) values ('elbu');
insert into cmu_employee (employee_id) values ('ben');
insert into cmu_employee (employee_id) values ('fibe');
insert into cmu_employee (employee_id) values ('fibl');
insert into cmu_employee (employee_id) values ('fibr');
commit;

Create an employee group:

gsh 18% employeesGroup = new GroupSave(grouperSession).assignName("edu:cmu:community:employees").assignSaveMode
(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).save();

Add a loader job to auto-populate students:

gsh 19% groupAddType("edu:cmu:community:employees", "grouperLoader");
gsh 20% setGroupAttr("edu:cmu:community:employees", "grouperLoaderType", "SQL_SIMPLE");
gsh 21% setGroupAttr("edu:cmu:community:employees", "grouperLoaderDbName", "grouper");
gsh 22% setGroupAttr("edu:cmu:community:employees", "grouperLoaderScheduleType", "CRON");
gsh 23% setGroupAttr("edu:cmu:community:employees", "grouperLoaderQuartzCron", "0 0 7 * * ?");
gsh 24% setGroupAttr("edu:cmu:community:employees", "grouperLoaderQuery", "SELECT employee_id AS subject_id,
'jdbc' AS subject_source_id FROM cmu_employee");

Run the loader one time to init:

gsh 25% employeesGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:employees", true);
gsh 26% loaderRunOneJob(employeesGroup);

Kick off the job and run loader automatically so it runs daily:

[appadmin@i2midev1 bin]$ cd /opt/grouper/1.6.1/grouper.apiBinary-1.6.1/bin
[appadmin@i2midev1 bin]$ ./gsh.sh -loader > /tmp/grouper.1.6.1_loader.log 2>&1 &

See that the group has the people it should:

gsh 27% employeesGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:employees", true);

gsh 28% employeesGroup.getMembers();
member: id='elbl' type='person' source='jdbc' uuid='079fb8bed3264e15aec231310e6135e0'
member: id='dousti' type='person' source='jdbc' uuid='09d7fcdb9e3b497d8711bb23ae6f80f4'
member: id='fibr' type='person' source='jdbc' uuid='172e3cd41e9d43ed8ac080e521e2e1e0'
member: id='elbu' type='person' source='jdbc' uuid='2da8882b79384bdc88a4e2ebcf6131dc'
member: id='ben' type='person' source='jdbc' uuid='35c959d61fb0458f9ba10cfe8ec2619b'
member: id='fibe' type='person' source='jdbc' uuid='b8faa3b714c24acf8967d0d35c3f44e1'
member: id='fibl' type='person' source='jdbc' uuid='dc4f29774c9b46db9053eb8e9522935a'
member: id='elbr' type='person' source='jdbc' uuid='fc76b63e08cf426a94823197ace6c97a'
```

Make the localBillingAdministratorRole and universityBillingAdministratorRole a requireGroup, and add the employee group as the "required" group. This means that if a user who is a local or university admin, then if they are ever not an employee anymore, then they will fall out of the overall group and also not have their permissions anymore. This can also be done with the UI, WS, or client

```
gsh 19% groupAddType("edu:cmu:it:apps:billing:roles:localBillingAdministrator", "requireInGroups");
gsh 20% setGroupAttr("edu:cmu:it:apps:billing:roles:localBillingAdministrator", "requireAlsoInGroups", "edu:cmu:
community:employees");
gsh 21% groupAddType("edu:cmu:it:apps:billing:roles:universityBillingAdministrator","requireInGroups");
gsh 22% setGroupAttr("edu:cmu:it:apps:billing:roles:universityBillingAdministrator", "requireAlsoInGroups",
"edu:cmu:community:employees");
gsh 23% addMember("edu:cmu:it:apps:billing:roles:localBillingAdministrator_systemOfRecord","elbl");
gsh 24% addMember("edu:cmu:it:apps:billing:roles:localBillingAdministrator_systemOfRecord","dousti");
gsh 25% addMember("edu:cmu:it:apps:billing:roles:localBillingAdministrator_systemOfRecord","elbr");
gsh 26% addMember("edu:cmu:it:apps:billing:roles:localBillingAdministrator_systemOfRecord","elbu");
gsh 27% addMember("edu:cmu:it:apps:billing:roles:universityBillingAdministrator_systemOfRecord","ben");
gsh 28% addMember("edu:cmu:it:apps:billing:roles:universityBillingAdministrator_systemOfRecord","fibr");
gsh 29% addMember("edu:cmu:it:apps:billing:roles:universityBillingAdministrator_systemOfRecord","fibl");
gsh 30% addMember("edu:cmu:it:apps:billing:roles:universityBillingAdministrator_systemOfRecord","fibe");
```

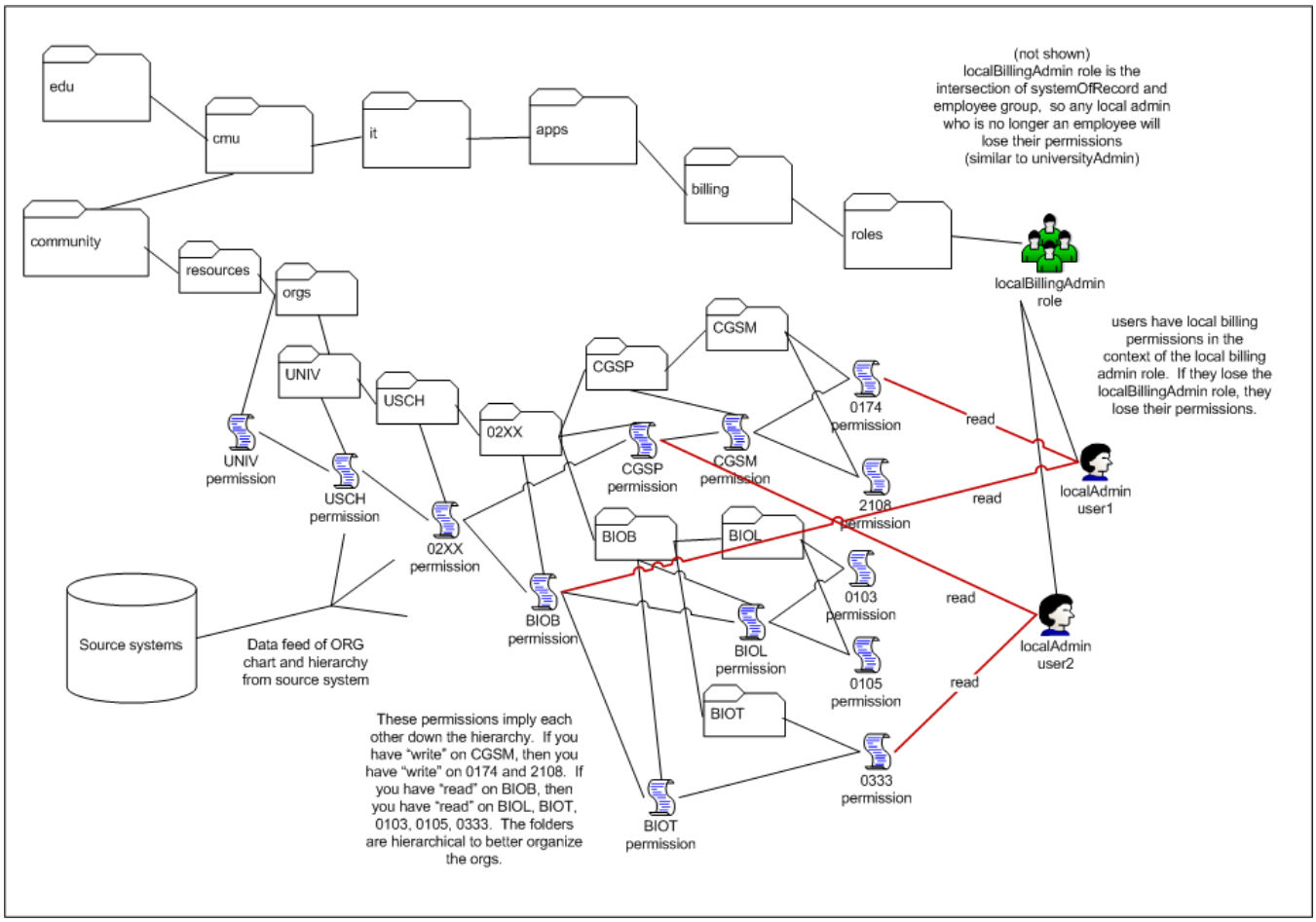
Assignment to the universityBillingAdministrator role could have start and end dates, or could be provisioned with workflow and approvals.

Permission resources

There are two tracks of permission resources, the ones specific to this application, and potentially the re-use the organization-wide org chart. If the school has a university wide org chart of resources which matches the requirements, it can be re-used. This is what Penn is doing.

So the org chart looks like this:

edu:cmu:community:resources:orgs:UNIV:USCH:02XX
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:0174
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:2108
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0103
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0105
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT
edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT:0333


edu
cmu
it
apps
billing
roles
community
resources
orgs
UNIV
USCH
02XX
CGSP
CGSM
0174 permission
2108 permission
UNIV permission
USCH permission
02XX permission
CGSP permission
CGSM permission
BIOB
BIOL
BIOT
0103 permission
0105 permission
0333 permission
BIOB permission
BIOL permission
BIOT permission
Source systems
Data feed of ORG chart and hierarchy from source system
These permissions imply each other down the hierarchy. If you have "write" on CGSM, then you have "write" on 0174 and 2108. If you have "read" on BIOB, then you have "read" on BIOL, BIOT, 0103, 0105, 0333. The folders are hierarchical to better organize the orgs.
(not shown)
localBillingAdmin role is the intersection of systemOfRecord and employee group, so any local admin who is no longer an employee will lose their permissions
(similar to universityAdmin)
localBillingAdmin role
users have local billing permissions in the context of the local billing admin role. If they lose the localBillingAdmin role, they lose their permissions.
read
read
read
read
localAdmin user1
localAdmin user2

```
The orgs and org hierarchy can be loaded from a database view.  Some tables will simulate that.

CREATE TABLE cmu_org_permission_name (
  permission_name varchar(200) NOT NULL,
  permission_display_name varchar(200) default NULL,
  PRIMARY KEY  (permission_name) );

insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX - School or Arts and
Sciences');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:CGSP', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP - LPS BA Services');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:CGSP:CGSM', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM - LPS
Masters Programs');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:0174', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:0174
- Master of Science in Applied Geoscience');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:2108', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:2108
- Master of Applied Positive Psychology');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB - Biology Dept BA
Services');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB:BIOL', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL - Biology
Parent');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0103', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0103
- Biology');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0105', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:0105
- Biochemistry');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB:BIOT', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT - Biology
BA Other Orgs');
insert into cmu_org_permission_name (permission_name, permission_display_name) values ('edu:cmu:community:
resources:orgs:UNIV:USCH:02XX:BIOB:BIOT:0333', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOT:BIOT:0333
- Biology Business Administration Services');
commit;
```

And there is a hierarchy where if you are assigned CGSP, then you also have the decendents of that node. Note, grouper can help manage the hierarchy table if the org list exists. Documented here

```
--table that holds the permission hierarchy relationships

CREATE TABLE cmu_org_permission_hierarchy (
  if_has_permission_name varchar(200) NOT NULL,
  then_has_permission_name varchar(200) NOT NULL,
  PRIMARY KEY  (if_has_permission_name,then_has_permission_name),
  KEY FK_cmu_org_permission_hier2 (then_has_permission_name),
  CONSTRAINT FK_cmu_org_permission_hier2 FOREIGN KEY (then_has_permission_name) REFERENCES
cmu_org_permission_name (permission_name),
  CONSTRAINT FK_cmu_org_permission_hier1 FOREIGN KEY (if_has_permission_name) REFERENCES
cmu_org_permission_name (permission_name)
);

INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:CGSP', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:
0174');
```

```
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:CGSP:CGSM:
2108');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:BIOB', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:
0103');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOL:
0105');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:BIOB', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT');
INSERT INTO cmu_org_permission_hierarchy (if_has_permission_name, then_has_permission_name) VALUES ('edu:cmu:
community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT', 'edu:cmu:community:resources:orgs:UNIV:USCH:02XX:BIOB:BIOT:
0333');
COMMIT;

# create the permission definition, configure the actions available to be "read" and "write"

orgPermissionDef = new AttributeDefSave(grouperSession).assignCreateParentStemsIfNotExist(true).assignName("edu:
cmu:community:resources:permissionDefinition").assignAttributeDefType(AttributeDefType.perm).
assignToEffMembership(true).assignToGroup(true).save();

# make a loader job to automatically keep the org permission hierarchy in sync

orgPermissionDef.getAttributeDelegate().assignAttributeByName(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoader");
orgPermissionDef.getAttributeValueDelegate().assignValue(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoaderType", "ATTR_SQL_SIMPLE");
orgPermissionDef.getAttributeValueDelegate().assignValue(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoaderQuartzCron", "0 0 7 * * ?");
orgPermissionDef.getAttributeValueDelegate().assignValue(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoaderAttrsLike", "%");
orgPermissionDef.getAttributeValueDelegate().assignValue(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoaderAttrQuery", "SELECT permission_name attr_name, permission_display_name attr_display_name FROM
cmu_org_permission_name");
orgPermissionDef.getAttributeValueDelegate().assignValue(GrouperCheckConfig.attributeLoaderStemName() + ":
attributeLoaderAttrSetQuery", "SELECT if_has_permission_name if_has_attr_name, then_has_permission_name
then_has_attr_name FROM cmu_org_permission_hierarchy");

# kick it off one time
loaderRunOneJobAttr(orgPermissionDef);

# restart the loader so it updates daily
[appadmin@i2midev1 bin]$ cd /opt/grouper/1.6.1/grouper.apiBinary-1.6.1/bin
[appadmin@i2midev1 bin]$ ./gsh.sh -loader > /tmp/grouper.1.6.1_loader.log 2>&1 &
```

Allow the app admins to read and/or assign these privilege resources. To keep things simple, we will use the university admins, though could be a different group/role

```
orgPermissionDef.getPrivilegeDelegate().grantPriv(universityBillingAdministratorRole.toSubject(),
AttributeDefPrivilege.ATTR_READ, false);
orgPermissionDef.getPrivilegeDelegate().grantPriv(universityBillingAdministratorRole.toSubject(),
AttributeDefPrivilege.ATTR_UPDATE, false);
```

Limit the assignments of this permission to the local billing admin role

```
orgPermissionDef.getAttributeDefScopeDelegate().assignOwnerGroup(localBillingAdministratorRole);
```

The permission resource should have READ/WRITE action options

```
orgPermissionDef.getAttributeDefActionDelegate().configureActionList(GrouperUtil.toSet(new Object[]{"read",
"write"}));
```

When specifying a local billing administrator, the system wide admin would assign the role edu.cmu.it.apps.billing.roles.localBillingAdministrator to the user, and will assign READ to one of the orgs in the global shared org chart.

Lets assign this permission to some individuals (note this can be done via GSH, WS, or grouperClient)

```
gsh 33% subjectElbl = SubjectFinder.findById("elbl", true);
gsh 34% subjectDousti = SubjectFinder.findById("dousti", true);
gsh 35% subjectElbr = SubjectFinder.findById("elbr", true);
gsh 36% subjectElbu = SubjectFinder.findById("elbu", true);
gsh 37% permission02XX = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX",
true);
gsh 38% permissionBIOL = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
BIOB:BIOL", true);
gsh 39% permission0333 = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
BIOB:BIOT:0333", true);
gsh 40% permission0174 = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
CGSP:CGSM:0174", true);
gsh 41% permissionCGSP = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
CGSP", true);
gsh 52% permissionCGSM = AttributeDefNameFinder.findByName("edu:cmu:community:resources:orgs:UNIV:USCH:02XX:
CGSP:CGSM", true);
gsh 45% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permission02XX, subjectElbl);
gsh 46% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permissionBIOL, subjectDousti);
gsh 47% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permission0333, subjectElbr);
gsh 48% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permission0174, subjectElbu);
gsh 49% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permissionCGSP, subjectDousti);
gsh 50% localBillingAdministratorRole.getPermissionRoleDelegate().assignSubjectRolePermission("read",
permissionCGSM, subjectElbr);
```

Create the other permissions in the application

```
# definition for the permissions
gsh 54% billingPermissionsDefinition = new AttributeDefSave(grouperSession).assignCreateParentStemsIfNotExist
(true).assignName("edu:cmu:it:apps:billing:permissionDefs:billingPermissions").assignAttributeDefType
(AttributeDefType.perm).assignToEffMembership(true).assignToGroup(true).save();
```

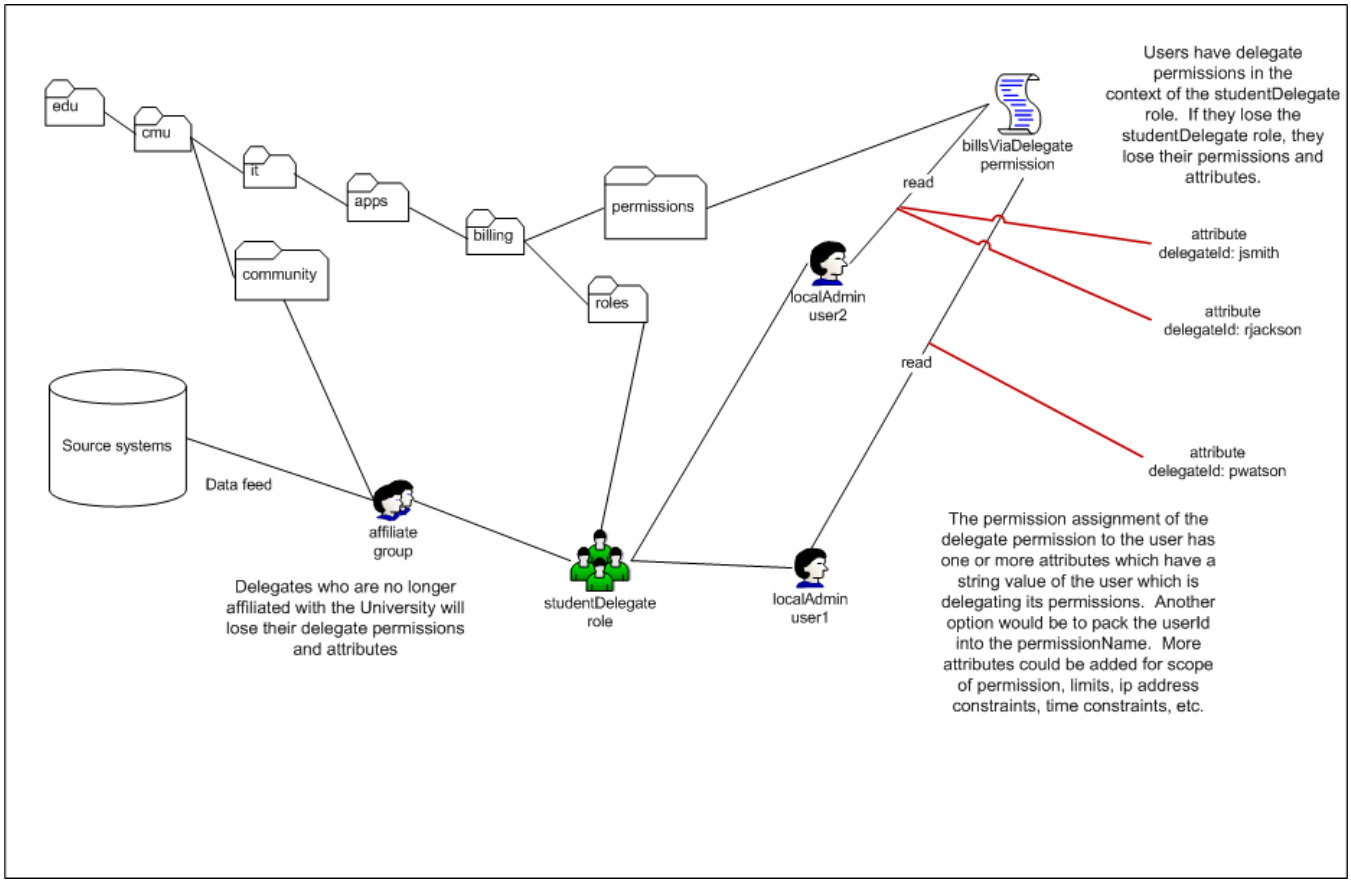
The permission resource should have READ/WRITE action options

```
 gsh 55% billingPermissionsDefinition.getAttributeDefActionDelegate().configureActionList(GrouperUtil.toSet
(newObject[]{"read","write"}));
```

This definition has several resource names associated with it, e.g. my own bills, all bills, and delegate

```
gsh 55% myOwnBillsPermission = new AttributeDefNameSave(grouperSession, billingPermissionsDefinition).assignName
("edu:cmu:it:apps:billing:permissions:myOwnBills").assignCreateParentStemsIfNotExist(true).save();
gsh 56% allBillsPermission = new AttributeDefNameSave(grouperSession, billingPermissionsDefinition).assignName
("edu:cmu:it:apps:billing:permissions:allBills").assignCreateParentStemsIfNotExist(true).save();
gsh 57% billsViaDelegatePermission = new AttributeDefNameSave(grouperSession, billingPermissionsDefinition).
assignName("edu:cmu:it:apps:billing:permissions:billsViaDelegate").assignCreateParentStemsIfNotExist(true).
save();

# Assign two of these to the application roles (RBAC)
gsh 58% universityBillingAdministratorRole.getPermissionRoleDelegate().assignRolePermission("read",
allBillsPermission);
gsh 59% studentRole.getPermissionRoleDelegate().assignRolePermission("read", myOwnBillsPermission);
```

For the delegation, we could pack the id of the user you are the delegate of into the permission resource name, or you could just put an attribute on that permission assignment. Lets go the route of the attribute on the assignment just to show how to put metadata on permission assignments (e.g. scopes, time range, ip address, etc).



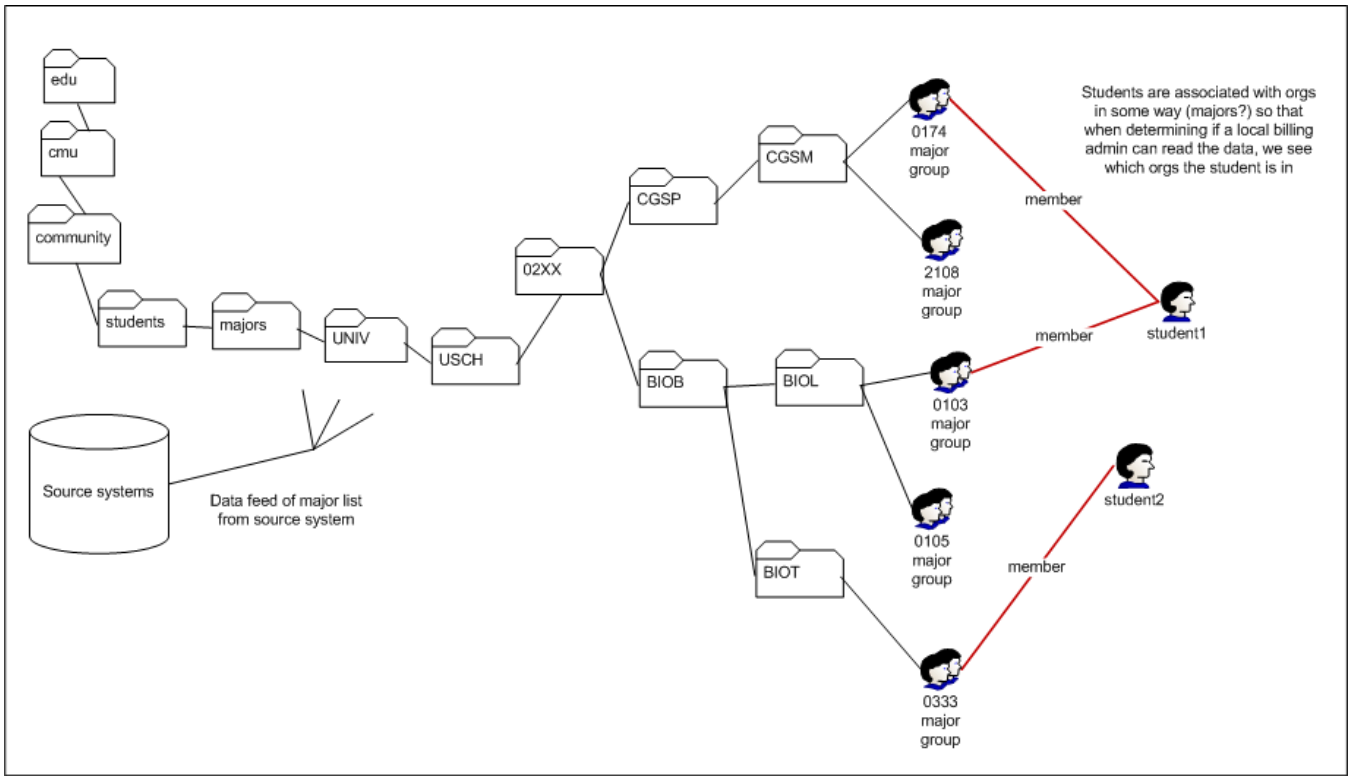
This is the attribute definition for that attribute

```
# attribute definition
gsh 61% delegateIdDefinition = new AttributeDefSave(grouperSession).assignCreateParentStemsIfNotExist(true).
assignName("edu:cmu:it:apps:billing:permissionDefs:delegateIdDef").assignAttributeDefType(AttributeDefType.
attr).setAssignToEffMembershipAssn(true).save();

# attribute name
gsh 62% delegateIdAttribute = new AttributeDefNameSave(grouperSession, delegateIdDefinition).assignName("edu:
cmu:it:apps:billing:attributes:delegateId").assignCreateParentStemsIfNotExist(true).save();

# the attribute is single valued, and multi assignable (can be delegate of multiple), and string valued (string
is ID you are delegate of)
gsh 63% delegateIdDefinition.setMultiAssignable(true);
gsh 64% delegateIdDefinition.setMultiValued(false);
gsh 65% delegateIdDefinition.setValueType(AttributeDefValueType.string);
gsh 66% delegateIdDefinition.store();

# We need a role for people who can be studentDelegates, so lets make a cmu affiliate group (employees and
students for now), and then assign that to the studentDelegate role
affiliatesGroup = new GroupSave(grouperSession).assignName("edu:cmu:community:affiliates").assignSaveMode
(SaveMode.INSERT_OR_UPDATE).assignCreateParentStemsIfNotExist(true).save();

gsh 68% studentsGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:students", true);
gsh 69% employeesGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:employees", true);
gsh 70% affiliatesGroup.addMember(studentsGroup.toSubject(), false);
gsh 71% affiliatesGroup.addMember(employeesGroup.toSubject(), false);
gsh 72% studentDelegateRole = GroupFinder.findByName(grouperSession, "edu:cmu:it:apps:billing:roles:
studentDelegate", true);
gsh 73% studentDelegateRole.addMember(affiliatesGroup.toSubject());
gsh 74% groupAddType("edu:cmu:it:apps:billing:roles:studentDelegate", "requireInGroups");
gsh 75% setGroupAttr("edu:cmu:it:apps:billing:roles:studentDelegate", "requireAlsoInGroups", "edu:cmu:community:
affiliates");

# lets delegate a bill for student gsh 1% grouperSession = GrouperSession.startRootSession();
gsh 2% studentDelegateRole = GroupFinder.findByName(grouperSession, "edu:cmu:it:apps:billing:roles:
studentDelegate", true);
gsh 3% subjectFibe = SubjectFinder.findById("fibe", true);
gsh 4% billsViaDelegatePermission = AttributeDefNameFinder.findByName("edu:cmu:it:apps:billing:permissions:
billsViaDelegate", true);
gsh 5% attributeAssignResult = studentDelegateRole.getPermissionRoleDelegate().assignSubjectRolePermission
("read", billsViaDelegatePermission , subjectFibe);
gsh 6% attributeAssign = attributeAssignResult.getAttributeAssign();
gsh 8% delegateIdAttribute = AttributeDefNameFinder.findByName("edu:cmu:it:apps:billing:attributes:delegateId",
true);
gsh 9% attributeAssignResultDelegate = attributeAssign.getAttributeDelegate().addAttribute("read",
delegateIdAttribute);
gsh 10% attributeAssignDelegate = attributeAssignResultDelegate.getAttributeAssign();
gsh 11% valueDelegate = attributeAssignDelegate.getValueDelegate();
gsh 12% valueDelegate.addValue("babr");
```

So when a student wants to delegate the viewing of bills, they click the button on the screen to delegate, and:

- The system sees if the user to be delegated to, has the role: edu:cmu:it:apps:billing:roles:studentDelegate
    - If not, then it either assigns that role or determines the user is not affiliated
- The system sees if the permission is assigned, and if not, will assign the permission to the user being delegated to in the context of the role: edu.cmu.it.apps.billing.roles.studentDelegate
- The system sees if the attribute with correct value exists, and if not assigns an attribute which is the user id of the user delegating

Note: a privilege resource assignment in Grouper can be marked with a delegation flag of T (true), F (false), or G (grant, it is true, and the recipient can grant the delegation flag to whom they delegate to). Not sure if this is useful for this application or not.



Students majors

When looking up permissions of if someone can see a student's bill, we need to know which org the student is affiliated with. We can represent that with groups which are majors. Lets make a table which simulated a view of the source system.

Setup the loader group (Note, can be done with GSH, API, WS, UI)

```
# create a table to simulate the view on source system

CREATE TABLE cmu_student_major (
  student_id varchar(100) NOT NULL,
  group_name varchar(200) NOT NULL,
  PRIMARY KEY  (student_id,group_name)
);

# insert some test data

INSERT INTO cmu_student_major (student_id, group_name) VALUES ('kebe', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:CGSP:CGSM:0174');
INSERT INTO cmu_student_major (student_id, group_name) VALUES ('kebe', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:BIOB:BIOT:0333');
INSERT INTO cmu_student_major (student_id, group_name) VALUES ('kebl', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:CGSP:CGSM:2108');
INSERT INTO cmu_student_major (student_id, group_name) VALUES ('kebr', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:BIOB:BIOL:0103');
INSERT INTO cmu_student_major (student_id, group_name) VALUES ('kebu', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:BIOB:BIOL:0105');
INSERT INTO cmu_student_major (student_id, group_name) VALUES ('keco', 'edu:cmu:community:student:majors:UNIV:
USCH:02XX:BIOB:BIOT:0333');
commit;

# Create the group with the loader properties, and setup the loader params

gsh 0% grouperSession = GrouperSession.startRootSession();
gsh 1% loaderGroup = new GroupSave(grouperSession).assignName("edu:cmu:community:student:majorLoaderGroup").
assignCreateParentStemsIfNotExist(true).save();
gsh 2% groupAddType("edu:cmu:community:student:majorLoaderGroup", "grouperLoader");
gsh 5% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderType", "SQL_GROUP_LIST");
gsh 6% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderDbName", "grouper");
gsh 7% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderScheduleType", "CRON");
gsh 8% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderQuartzCron", "0 30 7 * * ?");
gsh 9% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderQuery", "SELECT student_id AS
subject_id, 'jdbc' AS subject_source_id, group_name FROM cmu_student_major");
gsh 10% setGroupAttr("edu:cmu:community:student:majorLoaderGroup", "grouperLoaderGroupQuery", "SELECT DISTINCT
group_name FROM cmu_student_major");
gsh 11% loaderGroup = GroupFinder.findByName(grouperSession, "edu:cmu:community:student:majorLoaderGroup");
gsh 12% loaderRunOneJob(loaderGroup);

# Restart the loader
[appadmin@i2midev1 bin]$ ./gsh.sh -loader > /tmp/grouper.1.6.1_loader.log 2>&1 &
```

Checking permissions

Permissions can be checked with GSH, SQL, WS, or client. The application will either need to check permissions dynamically, or export and cache all the relevant permissions periodically, or for a user on their login. The following decision making process needs to occur:

Can user X see bill for user Y:

- If user X has permission: edu.cmu.it.apps.billing.permissions.viewAllBills in the role edu.cmu.it.apps.billing.roles.universityBillingAdministrator, then the answer is YES
- If Y user is the same as X, and X has edu.cmu.it.apps.billing.permissions.viewOwnBills in the role edu.cmu.it.apps.billing.roles.student, then the answer is YES
- If Y has the ID of 99887766, and X has the permission READ for permission edu:cmu:it:apps:billing:permissions:billsViaDelegate in the role edu.cmu.it.apps.billing.roles.studentDelegate, with an attribute on it: delegateId with the value 99887766, then the answer is YES
- Find the orgs that user Y is associated with. Assume it is org A:B:C, and D:E:F. See if X can READ either edu.cmu.community.resources.orgs.UNIV.USCH.A.B.C or edu.cmu.community.resources.orgs.UNIV.USCH.D.E.F for the role: edu.cmu.it.apps.billing.roles.localBillingAdministrator. If so, then the answer is YES
- Else, the answer is no.

The command line program above implements this logic with Grouper WS via the grouperClient API