Grouper Role and Permission Management

Wiki Home	Grouper Release Announcements	Grouper Guides	Grouper Deployment Guide	Community Contributions	Internal Developer Resources

This topic is discussed in the "Grouper Permissions" training video and in the "LITE UI Permissions - PART 1" training video and in the "LITE UI Permissions - PART 2" training video.

Diagrams

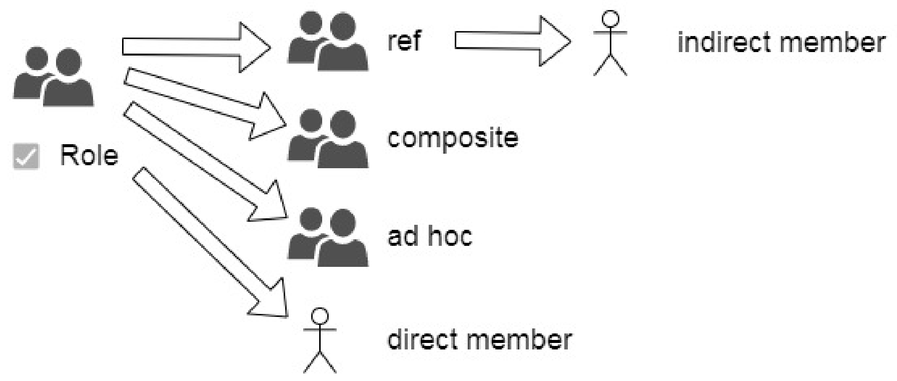
Note: there are 4 levels of hierarchies in Grouper permissions.

1. Indirect group membership to have a role
2. Permissions that imply other permissions (so you can assign one permissions and get a lot of rights)
3. Actions that imply other actions (e.g. admin implies all other actions)
4. Role inheritance so a role can inherit all permissions from another role, and add some more

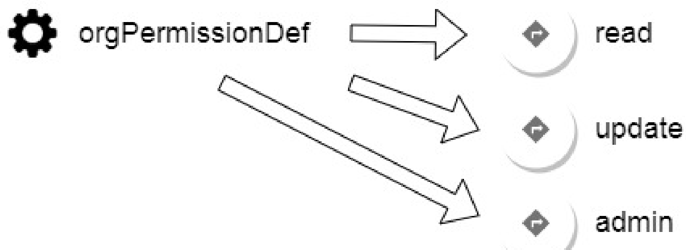
Here are examples and diagrams of Grouper permissions

1. Take a policy group (e.g. in an app folder) and mark it as a "role". A "role" is a group that can have permissions assigned

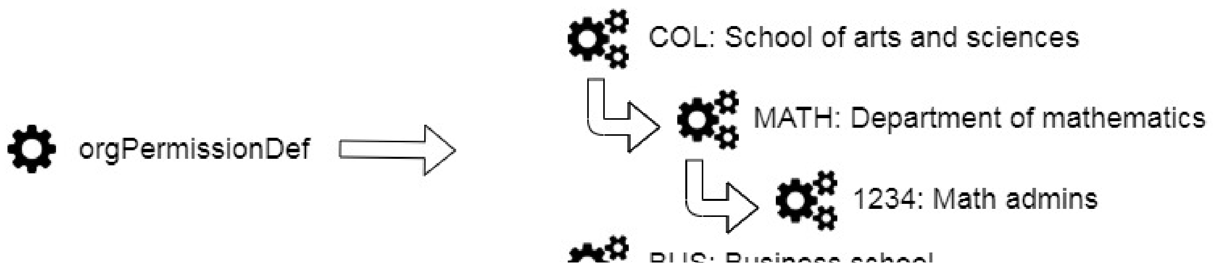
2. Just like any policy group, it consists of ref groups, ad hocs, composites, etc. The app could have one or many roles. Any member of a group which is a member of the role "has the role"



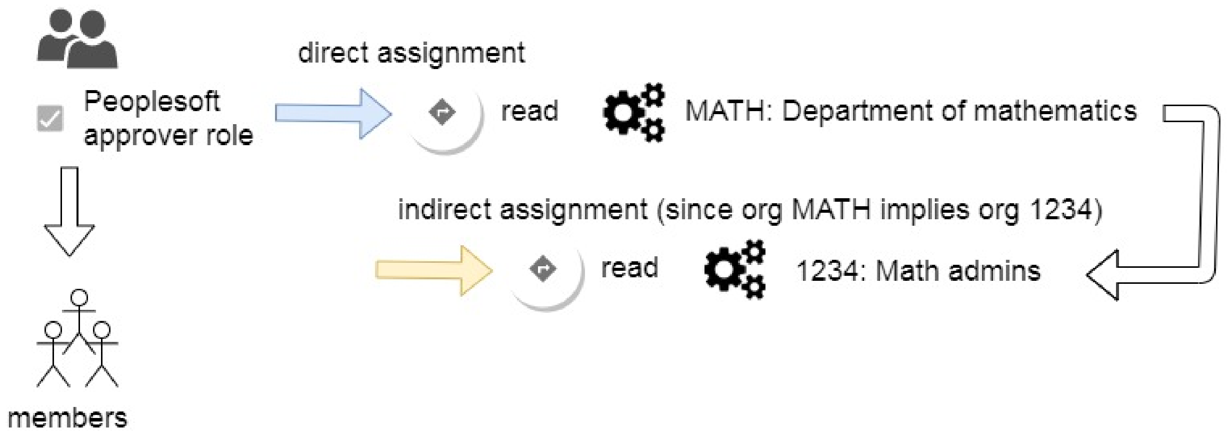
3. Permission definitions (under the covers are Grouper attribute definitions) configure the permissions. Specifically we need "actions" (part of the 'triple' permission assignment. Note: for this example we are securing data at a row level by org. So the org list can be maintained centrally and used by multiple applications. This list of actions is ad hoc depending on the needs of the application



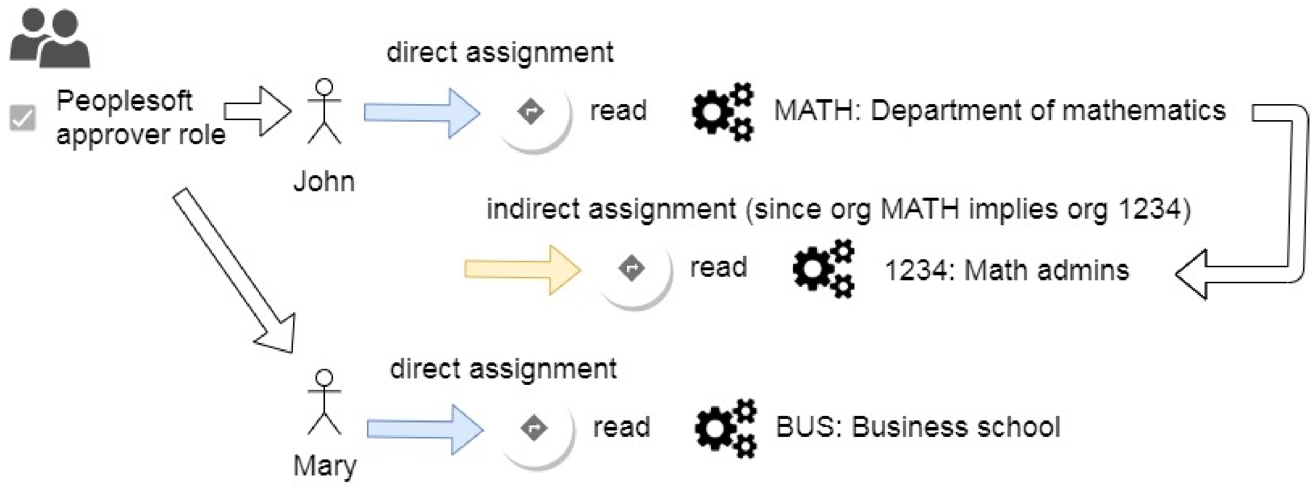
4. Permission resources (or "permissions") (under the covers are Grouper attribute names) are things you can assign to roles or users. Permissions can imply other permissions (just as being a member of a group can lead to other indirect group memberships). In this case the list of org permissions is maintained by a loader job. Each permission is configured by its permission definition. Note: just like groups, the Grouper folder location of permissions is not related to if permissions imply each other. Permissions could be located in the same folder or whatever folder makes sense. For orgs, it is recommended to not use folders to organize any hierarchy since it might change.


orgPermissionDef
COL: School of arts and sciences
MATH: Department of mathematics
1234: Math admins

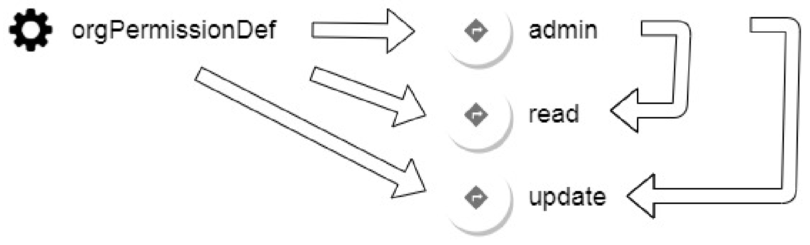
5. Assign a permission to a role. This is a "triple" assignment since there is also an action. Note that membership is a "tuple" assignment, group and entity. Any entity in the "role" can perform the action on the resource, in the context of the role. "In the context of the role" means only while considering that role. In another application in another role, the user cannot perform that action. Anyone in role "Peoplesoft approver" can read orgs "MATH" and "1234".

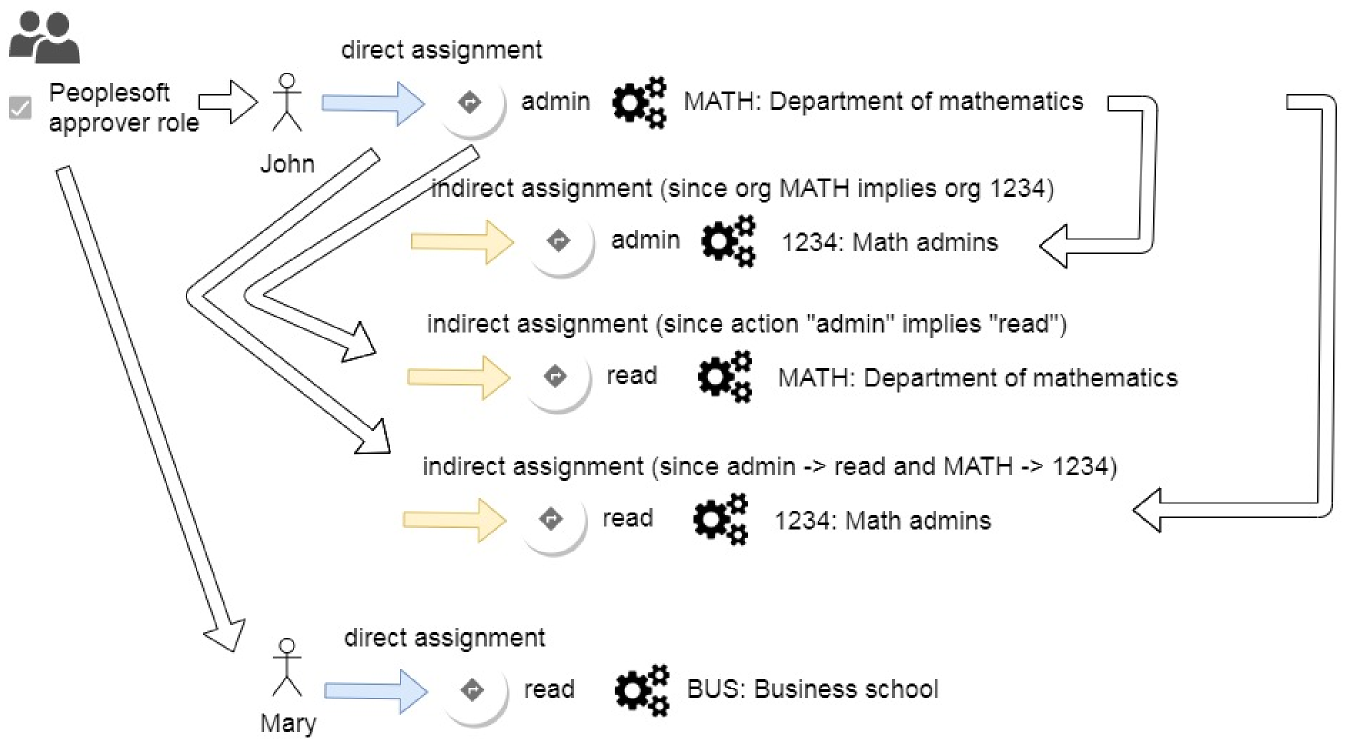

Peoplesoft approver role
direct assignment
read
MATH: Department of mathematics
indirect assignment (since org MATH implies org 1234)
read
1234: Math admins
members

6. Assign a permission to a user in a role. This means that while the user has the role, and while in the context of the role, the user has the permission (and implied permission), but other users in the role do not (unless that have an assignment to it). If the user is deprovisioned from the role, they lose the permissions. Note: the user has the permissions for role-wide permissions in addition to individual permissions.

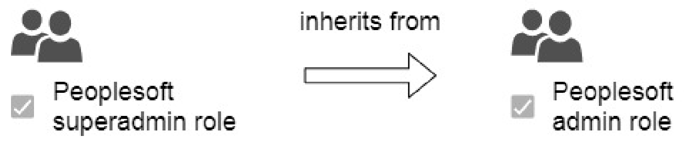


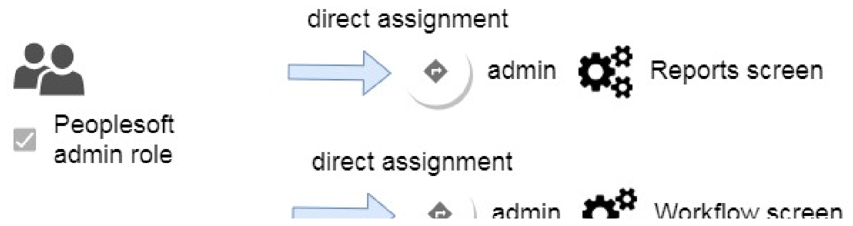
7. Actions can have hierarchies (well, actually they are directed graphs). Configure an action to imply other actions. In this case, if someone has "admin" then they can also "read" and "write"

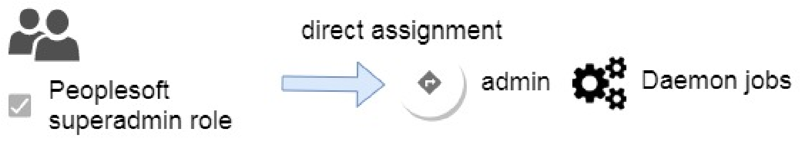


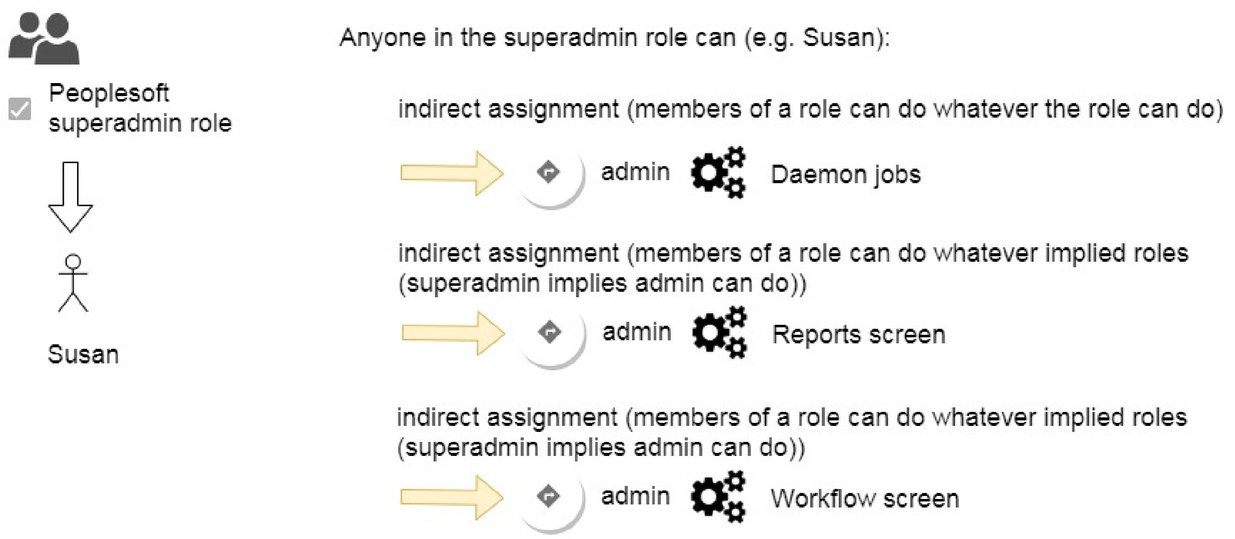

Peoplesoft approver role
John
direct assignment
admin
MATH: Department of mathematics
indirect assignment (since org MATH implies org 1234)
admin
1234: Math admins
indirect assignment (since action "admin" implies "read")
read
MATH: Department of mathematics
indirect assignment (since admin -> read and MATH -> 1234)
read
1234: Math admins
Mary
direct assignment
read
BUS: Business school

8. Roles can have hierarchies. This is an RBAC concept (like many of the other concepts). It doesnt affect membership of the roles. It means that a role inherits permissions that another role has. So if the "admin role" can do "XYZ", and the "superadmin role" inherits permissions from the "admin role", then anyone in the "superadmin role" can do XYZ, in addition to any permissions assigned to the "superadmin role". Note: this applies to permissions assigned to the role level, not to the individual in the role.









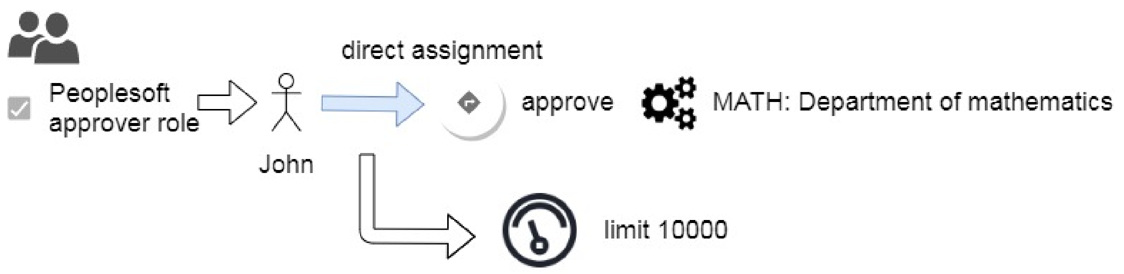
Anyone in the superadmin role can (e.g. Susan):

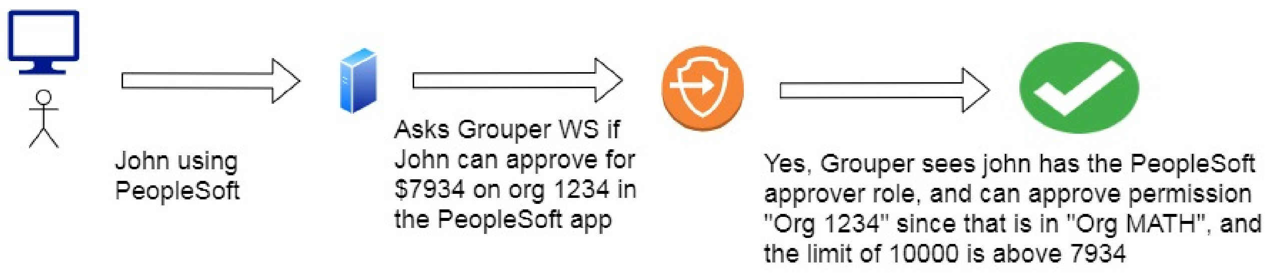
indirect assignment (members of a role can do whatever the role can do)

indirect assignment (members of a role can do whatever implied roles (superadmin implies admin can do))

indirect assignment (members of a role can do whatever implied roles (superadmin implies admin can do))

9. Permission assignments can have limits. These can an amount, or a time of day, or various things. The Grouper web services can accept an input and give an answer. Or the metadata can be exported to the application and it can make the decision.





Asks Grouper WS if John can approve for $7934 on org 1234 in the PeopleSoft app

Yes, Grouper sees john has the PeopleSoft approver role, and can approve permission "Org 1234" since that is in "Org MATH", and the limit of 10000 is above 7934

When to use Grouper Permissions

1. When the application can support permissions being provisioned to it
2. Helps if your application has a specific and probably custom UI to assign permissions
    a. E.g. imagine assigning permissions on Confluence pages outside of the Confluence app? Might be difficult to use
    b. Grouper has a permissions UI but it is generic
3. Grouper permissions do not provision with PSPNG. You need to provision permissions to the application using Grouper WS or SQL or Java
4. Grouper permissions will tell you real time when assignments change (for real time provisioning), but it only indicates that a role has changed somehow
    a. If you are doing a change log consumer or messaging, you need to get that indication and do a full sync of permissions for that role

Role and Permission Management as of v2.0 and above

- Grouper permission limits
- Grouper permissions allow and disallow
- Grouper permissions example
- Grouper resource or permission picker

Grouper has the capability to manage external applications' roles and permissions, and can function as a central permission management system.

Note that "privilege" is interchangeable with "permission", but Grouper already has documents about internal Grouper privileges on Groups / folders / etc. so the word "permission" is used here.

- Roles can be stored in Grouper. Roles can be assigned to subjects or groups.
- External application permission objects can be stored in Grouper.
- Permissions can be assigned to roles or to subjects since they are modeled as types of attributes on role memberships or roles.
- Roles can be configured to imply other roles. For example a Senior Loan Administrator is a Loan Administrator, plus a few more security grants. Roles can be connected like a directed graph of role inheritance
- Permissions can grouped into permissionSets. E.g. if an organization hierarchy was represented as permissions, then the higher level organizations can imply the lower level ones. Note this does not have to be a hierarchy
- Permission assignments have an optional "action" qualifier. This is a free form string which is configured per permission definition. E.g. a user can READ certain orgs, and WRITE certain other orgs.
- Permission actions can imply other actions. e.g. Having ADMIN on a permission resource implies being able to READ or WRITE it. Note, there are no built in actions (though a default "assign" exists if none specified). So the actions and action inheritance needs to be defined
- Permission assignments can be ALLOWED or DISALLOWED. With all the inheritance (permission resource, role, action, memberships), if a permission is allowed to a wide population, then it can be narrowed with a disallow. For example, someone could be assigned to READ all orgs in the University in the payroll system, except for the user's own org.
- Permission limits can be assigned to direct permission assignments. The limit can have a value or type string, numeric, date, etc. The limit has logic associated with it to use the optional value and context from the caller to decide if the permission is allowed or not. There are built-in limits for value (e.g. allowed to approve if value less then 50000), time of day (only allowed during business hours), ip address, etc.
- All permissions operations are exposed through the Grouper Lite UI
- Videos:
    - Demo of permissions
    - Attribute definitions (definition holds security, name, actions for all permission names associated)
    - Attribute definition privileges (attribute definition privileges control who can list the permissions, or assign them)
    - Permission name hierarchies
    - Role editor
    - Directed graph visualization
    - Permissions demos of allow/deny
        - Common setup
        - Role assignment vs individual assignment
        - Role assignment vs individual assignment up the hierarchy
        - Role assignment vs individual assignment up the hierarchy2
        - Action directed graph priority
        - Various role assignments
        - Role inheritance
        - Directed graph priority
        - Directed graph priority with tie
        - Resource directed graph tie with different actions
    - Permission limits UI

See also the Overview of Access Management Features page for guidelines of when to use rules, roles, permission limits, and enabled / disabled dates.

GSH commands

Sample

```
import edu.internet2.middleware.grouper.permissions.*;
import edu.internet2.middleware.grouper.permissions.PermissionEntry.PermissionType;

GrouperSession grouperSession = GrouperSession.startRootSession();

Group test = new GroupFinder().addGroupName("test:test").findGroup();
AttributeDefName perm = AttributeDefNameFinder.findByName("test:permName", true);

test.getPermissionRoleDelegate().assignRolePermission(perm);
Subject subject = SubjectFinder.findByIdAndSource("test.subject.0", "jdbc", true);
test.getPermissionRoleDelegate().assignSubjectRolePermission(perm, subject);

for (PermissionEntry permissionEntry : new PermissionFinder().assignPermissionType(PermissionType.role).
assignImmediateOnly(true).addRole("test:test").findPermissions()) {      System.out.println(permissionEntry.
getAttributeDefNameName());    }
```

Create a role

```
gsh 30% userSharerRole = rolesStem.addChildRole("userSharer", "userSharer");
```

Add a member to a role (in this case a group)

```
gsh 38% userSharerRole.addMember(studentsGroup.toSubject());
```

Create a permission definition

```
gsh 51% resourcesDef = resourcesStem.addChildAttributeDef("secureShareWebResources", AttributeDefType.perm);
```

Add one permissions resource name to another (permissionSet)

```
gsh 63% receiveSetResource.getAttributeDefNameSetDelegate().addToAttributeDefNameSet(splashResource);
```

Assign a permission to a role

```
gsh 70% userSharerRole.getPermissionRoleDelegate().assignRolePermission(sendSetResource);
```

Assign a permission to a member in a role

```
gsh 73% adminRole.getPermissionRoleDelegate().assignSubjectRolePermission(adminEmailButtonResource,
schleindMember);
```

Get the permission assignments (not necessarily active or allowed), assigned to a role, immediate, based on role name, print these out

```
gsh 123% for (PermissionEntry permissionEntry : new PermissionFinder().assignPermissionType(edu.internet2.
middleware.grouper.permissions.PermissionEntry.PermissionType.role).assignImmediateOnly(true).addRole("a:b").
findPermissions()) {      System.out.println(permissionEntry.getAttributeDefNameName());    }
    for (PermissionEntry permissionEntry : new PermissionFinder().assignPermissionType(edu.internet2.middleware.
grouper.permissions.PermissionEntry.PermissionType.role).assignImmediateOnly(true).addRole("a:b").
findPermissions()) {      System.out.println(permissionEntry.getAttributeDefNameName());    }
```

Get the permission assignments (not necessarily active or allowed), assigned to a role, immediate, based on permission name, print these out

```
for (PermissionEntry permissionEntry : new PermissionFinder().assignPermissionType(edu.internet2.middleware.
grouper.permissions.PermissionEntry.PermissionType.role).assignImmediateOnly(true).addPermissionName("a:b").
findPermissions()) {      System.out.println(permissionEntry.getRoleName());    }
```

sdf

SQL interface

The view for permissions is grouper_perms_all_v. Note, results here need to be processed is allow/disallow is used, also you should take into account if the records are active or not

get all attributes assigned to a role, assuming direct assignment (unassignable)

```
SELECT GPAV.ATTRIBUTE_DEF_NAME_NAME
  FROM grouper_perms_all_v gpav
 WHERE     GPAV.ROLE_NAME = 'a:b'
       AND gpav.permission_type = 'role'
       AND GPAV.ROLE_SET_DEPTH = 0
       AND GPAV.ATTR_ASSIGN_ACTION_SET_DEPTH = 0
       AND GPAV.ATTR_DEF_NAME_SET_DEPTH = 0
       AND GPAV.MEMBERSHIP_DEPTH = 0
```

get all roles that are assigned a given attribute, assuming direct assignment (unassignable)

```
SELECT GPAV.role_name
  FROM grouper_perms_all_v gpav
 WHERE     GPAV.ATTRIBUTE_DEF_NAME_NAME = 'a:b'
       AND gpav.permission_type = 'role'
       AND GPAV.ROLE_SET_DEPTH = 0
       AND GPAV.ATTR_ASSIGN_ACTION_SET_DEPTH = 0
       AND GPAV.ATTR_DEF_NAME_SET_DEPTH = 0
       AND GPAV.MEMBERSHIP_DEPTH = 0
```

See also

Access Management Features Overview

Grouper New Template Wizard

Training Slides, pages 31-38