Grouper attribute framework

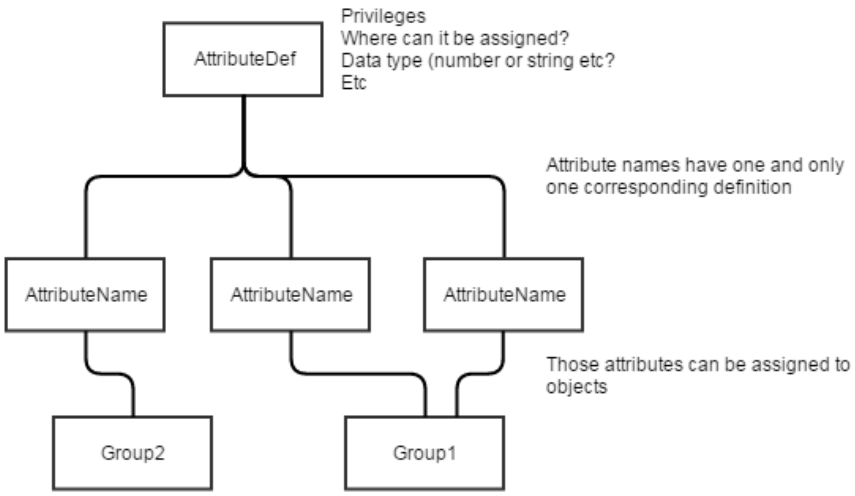
The Attribute Framework

The attribute framework is used for attaching metadata to various objects in the registry.

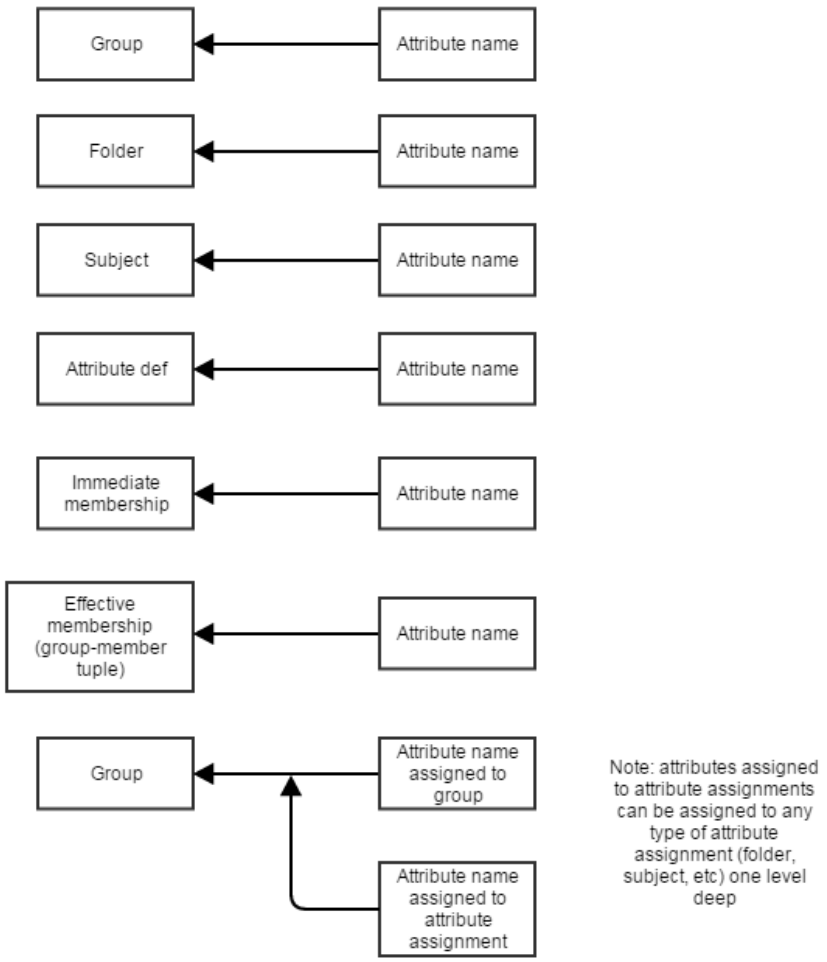
For information about the Attribute Framework UI, see this page

Features

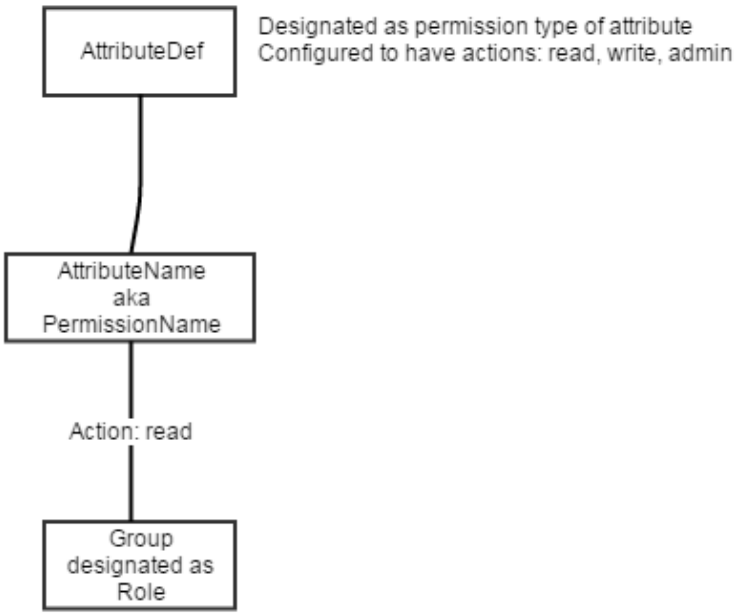
- Attributes have a definition (attributeDef), and a name (attributeName or aka attributeDefName). There is a one-to-many relationship between attributeDef and attributeName. The definition has all the configuration, and the name is assigned to the object



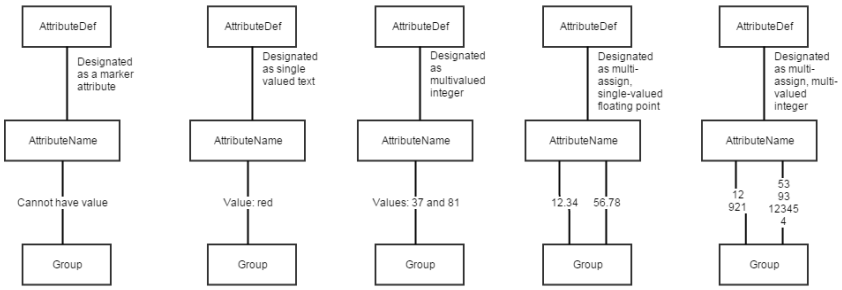
- Attributes can be assigned to groups, memberships (immediate or effective), members (i.e. subjects), folders, other attributes, and attribute assignments (one level deep)



- Attribute assignments that are permissions have an optional "action" qualifier for permission type attributes. This is a free form string which is configured per attribute definition



- Attributes could have free-form values, multi-valued, multi-assigned, limits as to where they can be applied, validation on values (TODO on validation), etc



- Attributes can have enabled/disabled dates.

Security on Attributes

- Attributes have security, coupled with other objects' security. Each attribute has the following lists:
  - ATTR_VIEW: can see that the attribute exists
  - ATTR_READ: can see the attribute assignment
  - ATTR_UPDATE: can assign or unassign the attribute
  - ATTR_ADMIN: can rename the attribute or assign security
  - ATTR_OPTIN: can assign the attribute to one's self
  - ATTR_OPTOUT: can unassign the attribute to one's self
  - ATTR_DEF_ATTR_READ (v2.2+): can see attributes assigned to this attribute definition (note you need READ on the assigned attributes definition too)
  - ATTR_DEF_ATTR_UPDATE (v2.2+): can assign attributes to this attribute definition (note you need UPDATE on the assigned attributes definition too)
- In order to perform operations on attributes, more security is needed on the underlying objects. For example, to assign an attribute to a group, you need ATTR_UPDATE on the attribute and GROUP_ATTR_UPDATE on the group. In order to assign an attribute to a membership, you need ATTR_UPDATE on the attribute, and UPDATE on the group.
- To make attributes easier to use, you can set these settings in the grouper.properties to make attributes "public". This means that if you have the appropriate security on the underlying object, you can add / edit / delete attributes from the object
- Attribute assignments can be delegatable. You can identify as assignment as true for delegatable, false for not delegatable, or "grant" which means the user can delegate it, and set the delegate flag.

```
attributeDefs.create.grant.all.attrRead          = true|false
attributeDefs.create.grant.all.attrUpdate        = true|false
```

- Many attributes can share the same security settings. An attribute consists of an attribute definition (where all the rules and security are applied), and an attribute name (just has a friendly and system name, and a description)

New Privileges for Attributes (in Grouper 2.2+)

In Grouper 2.2, new privileges have been added to determine if a subject has access to read or update attributes:

1. New privileges for groups — groupAttrRead and groupAttrUpdate.
2. New privileges for stems -- stemAttrRead and stemAttrUpdate.
3. New privileges for attributeDefs - attrDefAttrRead and attrDefAttrUpdate.

For group privileges:

1. To be able to read an attribute assigned to a group, you need attrRead (or attrAdmin) on the attribute and groupAttrRead (or admin) on the group.
2. To be able to update an attribute assigned to a group, you need attrUpdate (or attrAdmin) on the attribute and groupAttrUpdate (or admin) on the group.
3. To be able to read an attribute assigned to an attribute assigned to a group, you need attrRead (or attrAdmin) on both attributes and groupAttrRead (or admin) on the group.
4. To be able to update an attribute assigned to an attribute assigned to a group, you need attrUpdate (or attrAdmin) on both attributes and groupAttrUpdate (or admin) on the group.

For stem privileges:

1. To be able to read an attribute assigned to a stem, you need attrRead (or attrAdmin) on the attribute and stemAttrRead (or create or stem) on the stem.
2. To be able to update an attribute assigned to a stem , you need attrUpdate (or attrAdmin) on the attribute and stemAttrUpdate (or create or stem) on the stem.
3. To be able to read an attribute assigned to an attribute assigned to a stem, you need attrRead (or attrAdmin) on both attributes and stemAttrRead (or create or stem) on the stem.
4. To be able to update an attribute assigned to an attribute assigned to a stem, you need attrUpdate (or attrAdmin) on both attributes and stemAttrUpdate (or create or stem) on the stem.

For attributeDef privileges:

1. To be able to read an attribute assigned to an attributeDef, you need attrRead (or attrAdmin) on the attribute and attrDefAttrRead (or attrAdmin) on the attributeDef.
2. To be able to update an attribute assigned to an attributeDef , you need attrUpdate (or attrAdmin) on the attribute and attrDefAttrUpdate (or attrAdmin) on the attributeDef.
3. To be able to read an attribute assigned to an attribute assigned to an attributeDef, you need attrRead (or attrAdmin) on both attributes and attrDefAttrRead (or attrAdmin) on the attributeDef.
4. To be able to update an attribute assigned to an attribute assigned to an attributeDef, you need attrUpdate (or attrAdmin) on both attributes and attrDefAttrUpdate (or attrAdmin) on the attributeDef.

These changes affect permission assignments as well since they are just a special form of attribute assignments. Privileges for permissions will work as follows:

1. To be able to read a permission, you need attrRead (or attrAdmin) on the permissionDef and groupAttrRead (or admin) on the role. Note that the permissionDef and role used here may not be the ones that are actually associated with the attribute assignment due to inheritance.
2. To be able to assign/unassign a permission, you need attrUpdate (or attrAdmin) on the permissionDef and groupAttrUpdate (or admin) on the group.
3. In both cases, the same is true regardless of whether the permission is assigned to a role or a subject in a role.

Privileges for attributes on memberships and members have not changed:

For members:

1. To be able to read an attribute assigned to a member, you need attrRead (or attrAdmin) on the attribute.
2. Only wheel members and GrouperSystem can update attributes assigned to a member.

For memberships (immediate and effective):

1. To be able to read an attribute assigned to an effective membership, you need attrRead (or attrAdmin) on the attribute and read (or admin) on the group.
2. To be able to update an attribute assigned to an effective membership, you need attrUpdate (or attrAdmin) on the attribute and update (or admin) on the group.

Object types

Attribute definition (attributeDef):

An attribute definition holds the type of attribute (attribute, permission, limit, etc), privileges (who can assign or see assignments), restrictions about where it can be assigned (e.g. which folder), which owner types it can be assigned to (e.g. group, folder, etc), which type of value (e.g. text, numeric, etc), if it is multi-valued, if it can be assigned to the same owner object more than once at a time, etc. For permissions, this also holds which actions (in the triple) are available for this permission (e.g. READ, WRITE, ADMIN, etc). The attributeDef and attributeDefName live in a folder (stem) for namespace reasons. If you can CREATE in the stem, you can create attribute objects. It is up to you if you keep those centrally in a top level folder in Grouper or closer to the leaf where the application lives.

Attribute definition name (attributeDefName):

An attribute definition name has a many-to-one relationship to attributeDef. This is the thing that is assigned to the owner. It is similar to a list-of-values for the attribute. You cannot assign an attribute without having an attributeDefName. This allows you to share common settings among multiple attributes without having to reconfigure each value. You might have only one attributeDefName for an attributeDef, but generally you will have more than 1.

Attribute assignment:

This is the relationship between the owner (e.g. Group, Folder), and the attributeDefName. This might have a start/end timestamp, and other metadata (e.g. for permissions if it is delegatable, if it is allowed or forbidden, etc)

Attribute assignment value:

If an attributeDef is not of value type 'marker', then it can have a value, or multiple values (of the same type) if the attributeDef is labeled as multi-valued. The attribute assign value is a many to one on the attribute assignment. The value can have a type (e.g. text, integer, floating, timestamp, etc), and at some point it would be nice if it had the capability to have validations (e.g. a regex).

GSH commands

Create an attribute definition in GSH:

```
addRootStem("school","school");
addStem("school", "attr", "attr");
addStem("school:attr", "students", "students");
grouperSession = GrouperSession.startRootSession();
attrStudentsStem = StemFinder.findByName(grouperSession, "school:attr:students");
studentsAttrDef = attrStudentsStem.addChildAttributeDef("students", AttributeDefType.attr);
studentsAttrDef.setAssignToGroup(true);
studentsAttrDef.store();
```

Create an attribute name

```
attrArtsAndSciences = attrStudentsStem.addChildAttributeDefName(studentsAttrDef, "artsAndSciences",
"artsAndSciences");
```

Assign an attribute name to an object:

```
addStem("school", "math", "math");
groupBrainProject = addGroup("school:math", "brainProject", "brainProject");
groupBrainProject.getAttributeDelegate().assignAttribute(attrArtsAndSciences);
```

Set the security of an attribute definition

```
groupStudents = addGroup("school", "students", "students");
studentsAttrDef.getPrivilegeDelegate().grantPriv(groupStudents.toSubject(), AttributeDefPrivilege.ATTR_READ,
false);
```

Here is how to assign the same attribute multiple times (v1.6+) [note, needed to restart GSH to clear cache)

```
studentsAttrDef.setMultiAssignable(true);
studentsAttrDef.store();
-- RESTART GSH --
grouperSession = GrouperSession.startRootSession();
groupBrainProject = GroupFinder.findByName(grouperSession, "school:math:brainProject", true);
attrArtsAndSciences = AttributeDefNameFinder.findByName("school:attr:students:artsAndSciences", true);
groupBrainProject.getAttributeDelegate().addAttribute(attrArtsAndSciences);
groupBrainProject.getAttributeDelegate().addAttribute(attrArtsAndSciences);
groupBrainProject.getAttributeDelegate().addAttribute(attrArtsAndSciences);
groupBrainProject.getAttributeDelegate().retrieveAssignments(attrArtsAndSciences);
```

Here is how to assign a value (v1.6+) [note, needed to restart GSH to clear cache)

```
studentsAttrDef.setValueType(AttributeDefValueType.string);
studentsAttrDef.store();
groupBrainProject.getAttributeDelegate().retrieveAssignments(attrArtsAndSciences).iterator().next().
getValueDelegate().assignValue("hey");

//or, assign directly from group based on name of attributeDefName

//note, you get a result that has the assignment object, and whether this was a new assignment, or already
existed
//this is true for the value delegate, or the delegate on the assignment above
groupBrainProject.getAttributeValueDelegate().assignValue("school:attr:students:artsAndSciences", "hey");
```

Here is how to assign multiple values to an assignment (v1.6+) [note, needed to restart GSH to clear cache).

```
studentsAttrDef.setMultiValued(true);
studentsAttrDef.store();

//again, this returns a result object that has the object
groupBrainProject.getAttributeDelegate().retrieveAssignments(attrArtsAndSciences).iterator().next().
getValueDelegate().addValue("there");
groupBrainProject.getAttributeDelegate().retrieveAssignments(attrArtsAndSciences).iterator().next().
getValueDelegate().retrieveValuesString();
java.util.ArrayList: [there, hey]

//you could also do this directly with the value delegate
groupBrainProject.getAttributeValueDelegate().addValue("school:attr:students:artsAndSciences", "there");
groupBrainProject.getAttributeValueDelegate().retrieveValuesString("school:attr:students:artsAndSciences");
java.util.ArrayList: [there, there]
```

Example

Add a multi-valued attribute to a group, say mailAlternateAddress with two values "foo@memphis.edu" and "bar@memphis.edu". Here is the API code:

```
AttributeDef attributeDef = stem.addChildAttributeDef("someName", AttributeDefType.attr); attributeDef.
setAssignToGroup(true);
attributeDef.setMultiValued(true);
attributeDef.setValueType(AttributeDefValueType.string);
attributeDef.store();

AttributeDefName attributeDefName = stem.addChildAttributeDefName(attributeDef, "mailAlternateAddress",
"mailAlternateAddress");

group.getAttributeValueDelegate().assignValuesString(
        attributeDefName.getName(), GrouperUtil.toSet("foo@memphis.edu", "bar@memphis.edu"), true);
```

Example 2

Put an attribute on a membership

```
gsh 0%
gsh 1% grouperSession = GrouperSession.startRootSession();
edu.internet2.middleware.grouper.GrouperSession: 3c50fffcfb104b39adbd55d05d47355 7,'GrouperSystem','application'
gsh 2% folder = StemFinder.findByName(grouperSession, "Communities:LVC:LSC:MOU:UWM:UWMGroupAttributes",true)
stem: name='Communities:LVC:LSC:MOU:UWM:UWMGroupAttributes' displayName='Communi ties:LVC:LSC:MOU:UWM:
UWMGroupAttributes' uuid='97f4c134149941fbad8906d3a1ed2340'
gsh 3% attributeDef = folder.addChildAttributeDef("attr01", AttributeDefType.attr);
edu.internet2.middleware.grouper.attr.AttributeDef: AttributeDef[name=Communities:LVC:LSC:MOU:UWM:
UWMGroupAttributes:attr01,uuid=f8756aac777947ff9ae386786de4a287]
gsh 4% attributeDef.setAssignToImmMembership(true);
gsh 5% attributeDef.store();
gsh 6% myAttributeName = folder.addChildAttributeDefName(attributeDef,"myAttributeName","myAttributeName");
edu.internet2.middleware.grouper.attr.AttributeDefName: AttributeDefName[name=Communities:LVC:LSC:MOU:UWM:
UWMGroupAttributes:myAttributeName ,uuid=db8d3c93b30e4e2f96d3cdd9ae1af737]
gsh 7% subject = SubjectFinder.findById("scott.koranda@LIGO.ORG", true);
subject: id='scott.koranda@LIGO.ORG' type='person' source='ligo' name='Scott Koranda'
gsh 8% member = MemberFinder.findBySubject(grouperSession, subject, true);
member: id='scott.koranda@LIGO.ORG' type='person' source='ligo' uuid='56246fe035bd4266bc92abb617430033'
gsh 9% group = GroupFinder.findByName(grouperSession, "Communities:LVC:LSC:MOU:UWM:UWMGroupMembers");
group: name='Communities:LVC:LSC:MOU:UWM:UWMGroupMembers' displayName='Communities:LVC:LSC:MOU:UWM:
UWMGroupMembers' uuid='00918b49-ad44-49aa-8b13-49d8a1aa459c'
gsh 10% membership = MembershipFinder.findImmediateMembership(grouperSession, group,subject,Group.
getDefaultList(), true)
edu.internet2.middleware.grouper.Membership: Membership[createTime=1270587726584,
creatorUuid=e00f1b26f1c340db8845e1dfe297f01b,depth=0,listName=members,listType=list,
memberUuid=56246fe035bd4266bc92abb617430033,groupId=00918b49-ad44-49aa-8b13-49d8a1aa459c,type=immediate,
uuid=faefbff7e2ce4561b86c7e070fcd0ac9:a5b12a759438462a996842dca313ccfc]
gsh 11% membership.getAttributeDelegate().assignAttribute(myAttributeName);
edu.internet2.middleware.grouper.attr.assign.AttributeAssignResult: edu.internet2.middleware.grouper.attr.
assign.AttributeAssignResult@488e753c
gsh 12% membership.getAttributeDelegate().retrieveAssignments(myAttributeName);
edu.internet2.middleware.grouper.attr.assign.AttributeAssign: AttributeAssign
[id=350594e5dc39431ea56c17635eab253f,action=assign,attributeDefName=Communities:LVC:LSC:MOU:UWM:
UWMGroupAttributes:evil,membershipId=faefbff7e2ce4561b86c7e070fcd0ac9]
gsh 13%
```

Exclude attributes from audits, changelog, and point in time

```
# comma separated names of attribute defs will not be audited or change log or point in time
# same as ${edu.internet2.middleware.grouper.cfg.GrouperConfig.retrieveConfig().propertyValueStringRequired
('grouper.attribute.rootStem')}
grouper.attribute.namesOfAttributeDefsToIgnoreAuditsChangeLogPit.elConfig = $$grouper.attribute.rootStem$$:
userData:grouperUserDataValueDef,$$grouper.attribute.rootStem$$:instrumentationData:
instrumentationDataInstanceCountsDef,$$grouper.attribute.rootStem$$:instrumentationData:
instrumentationDataInstanceDetailsDef,$$grouper.attribute.rootStem$$:instrumentationData:
instrumentationDataCollectorDetailsDef

# comma separated names of attribute def names will not be audited or change log or point in time
grouper.attribute.namesOfAttributeDefNamesToIgnoreAuditsChangeLogPit.elConfig = $$grouper.attribute.rootStem$$:
attestation:attestationCalculatedDaysLeft,$$grouper.attribute.rootStem$$:attestation:
attestationLastEmailedDate,$$grouper.attribute.rootStem$$:loaderMetadata:
grouperLoaderMetadataLastFullMillisSince1970,$$grouper.attribute.rootStem$$:loaderMetadata:
grouperLoaderMetadataLastIncrementalMillisSince1970,$$grouper.attribute.rootStem$$:loaderMetadata:
grouperLoaderMetadataLastSummary
```

See Also

- Presentation that includes an example of the attribute framework
- Attribute Framework UI