Custom Group Types, Fields, Attributes, Lists

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

Custom Group Types and Fields have been deprecated. It is recommended to use the Attribute Framework capabilities instead of the Custom Group Types and Fields. As of Grouper 2.2, Custom Group Types and legacy attributes represented as Fields are no longer part of Grouper. The old APIs continue to work for now; they just call the Attribute Framework. The upgrade procedure for Grouper 2.2 includes steps on the migration.

Custom Group Types and Fields

As shipped, Grouper groups have 6 attributes and 1 list, and every Grouper group has those 7 fields without exception (see the Grouper Glossary for a list of them and other Grouper-specific terms used in this section). Prior to Grouper version 2, deployers could augment the pool of available fields with their own ***Custom Fields***. These were gathered into sets of custom fields to form a ***Group Type***, and all defined group types were available to be assigned to individual groups by their ADMINs. Assignment of a group type to a group added the associated set of fields to that group. These custom fields could then be managed or accessed by the API or the UI, appear in XML exports, etc.

In this section we described two methods for loading group types, i.e., collections of custom fields, into your Groups Registry, and one method for deleting them. Before that, however, there is a bit more about Grouper fields.

Fields come in two flavors: ***attributes*** and ***lists***. **(NOTE: As if Grouper 2.2, fields as lists are still valid. Fields as attributes are not. Use the Attribute Framework for attributes)** Attributes have a single, string value. List fields are lists of subjects. Each field must have a declared ***Access Privilege*** necessary to read the field, and likewise an access privilege declared that's needed to write the field. And some fields are "required" - the required fields associated with a given group must all have a value, a requirement that is only enforced by an API caller (including the Grouper UI).

So, to create a custom group type is to declare its name, identify the names of fields associated with that type, define the type of each field (attribute or list), its read and write access privileges, and whether or not it is required. Described below are two means for so doing.

Using the XML Import tool to add a group type (Not valid in 2.2+)

The XML Import tool's metadata import capability can be used to load custom group types. Below is an example XML file that declares the group type named 'teaching' and its three associated fields, 'academic-staff', 'clerical-staff', and 'faculty_code'. The first two of these are lists and the third is an attribute, and the privileges necessary to read or write them is also declared. None of these fields are in fact required.

```
<?xml version="1.0" encoding="UTF-8"?>
<registry>
  <metadata>
    <groupTypesMetaData>
    <groupTypeDef name='teaching'>
      <field name='academic-staff' required='false' type='list' readPriv='read' writePriv='update'/>
      <field name='clerical-staff' required='false' type='list' readPriv='read writePriv='update'/>
      <field name='faculty_code' required='false' type='attribute' readPriv='read' writePriv='admin'/>
    </groupTypeDef>
    </groupTypesMetaData>
  </metadata>
</registry>
```

To successfully import this XML into your Groups Registry, the import.properties file must include the following declarations:

```
import.metadata.group-types=true
import.metadata.group-type-attributes=true
```

Please refer to the XML Import/Export Tool documentation for details on how to load this XML.

Using GrouperShell to create a group type

The Grouper API's GroupType method can be used directly to create types and fields. Here's an example of using the GrouperShell to create the "teaching" group type presented in the XML above:

```
gsh-0.0.1 0% GrouperSession.startRootSession()
subject: id='GrouperSystem' type='application' source='g:isa' name='GrouperSysAdmin'
gsh-0.0.1 1% subj=SubjectFinder.findById("GrouperSystem")
subject: id='GrouperSystem' type='application' source='g:isa' name='GrouperSysAdmin'
gsh-0.0.1 2% sess=GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession: e161f71d-19f3-4cef-b6b8-
f0de9b594aab,'GrouperSystem','application'
gsh-0.0.1 3% type=GroupType.createType(sess, "teaching")
edu.internet2.middleware.grouper.GroupType: teaching
gsh-0.0.1 4% read=Privilege.getInstance("read")
edu.internet2.middleware.grouper.Privilege: read
gsh-0.0.1 5% update=Privilege.getInstance("update")
edu.internet2.middleware.grouper.Privilege: update
gsh-0.0.1 6% admin=Privilege.getInstance("admin")
edu.internet2.middleware.grouper.Privilege: admin
gsh-0.0.1 7% type.addList(sess, "academic-staff", read, update)
edu.internet2.middleware.grouper.Field: academic-staff,teaching,list
gsh-0.0.1 8% type.addList(sess, "clerical-staff", read, update)
edu.internet2.middleware.grouper.Field: clerical-staff,teaching,list
```

Using GrouperShell to remove a group type

The Grouper API's GroupType and GroupTypeFinder methods can be used delete a group type. Here's an example of using the GrouperShell to delete the "teaching" group created above:

```
gsh-0.0.1 0% GrouperSession.startRootSession()gsh-0.0.1 2% subj=SubjectFinder.findById("GrouperSystem")
subject: id='GrouperSystem' type='application' source='g:isa' name='GrouperSystem'
gsh-0.0.1 2% sess=GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession: eb0c794b-09a1-4cc5-b45b-
4c2e4a6d3434,'GrouperSystem','application'
gsh-0.0.1 3% type=GroupTypeFinder.find("teaching")
edu.internet2.middleware.grouper.GroupType: teaching
gsh-0.0.1 4% type.delete(sess)
```