


Space Details

Key:	GrouperWG
Name:	Grouper
Description:	Grouper Product & Project Wiki
Creator (Creation Date):	steveo@internet2.edu (Apr 07, 2006)
Last Modifier (Mod. Date):	jbibbee@internet2.edu (Jul 06, 2006)

Available Pages

- Home 
 - About Grouper
 - Credits
 - Grouper Product
 - API Building & Configuration
 - Getting Started
 - Grouper Binary Release
 - LDAPPC
 - LDAPPC Annotated Configuration
 - LDAPPC Usage
 - Architecture
 - Bad Membership Finder Utility
 - Contact Information
 - Custom Group Types, Fields, Attributes, Lists
 - Customising the Grouper UI
 - Grouper UIs
 - Developer's Guide to the Grouper API
 - Glossary
 - UI Terminology
 - Grouper - Loader
 - Grouper loader classlist example from Penn
 - GrouperShell (gsh)
 - Grouper UI Components
 - Grouper UI Development Environment
 - Grouper Web Services
 - Authentication for Grouper Web Services
 - Grouper Web Services FAQ
 - Import-Export
 - Initializing Administration of Privileges
 - License
 - Nav
 - Prerequisites
 - Software Download
 - Archives
 - Coding new DDL

- Group Math v1.0
 - News
 - v0.0.1_gsh
 - v0.1.0_gsh
 - v1.0_API Configuration
 - v1.0_Import-Export
 - v1.0_RC1_Release Notes
 - v1.0_UI Building and Configuration
- Database Conversion v1.0 - v1.1
- Database Conversion v1.2.0 - v1.2.1
- Database Conversion v1.2.1 - v1.3.0
- Entity Relationship Diagram For Grouper 1.3.0
- Grouper change log v1.3
- Grouper change log v1.4
- Grouper change log v1.5
- Release Notes
- v1.3-Release Notes
- v1.4.0 Getting Started
- v1.4.1 Upgrade Instructions from v1.3.*
- v1.4 Release Notes
- Specs sheet
- Technical FAQ
 - PHPClient
 - PHPClientAddMember
 - PHPClientDeleteMembers
 - PHPClientGetMembers
 - PHPClientGroupSave
 - PHPClientStemSave
- UI Building and Configuration
 - Media Properties
- UI Customization Guide
- Unresolvable Subject Deletion Utility (USDU)
- Hooks
- Include exclude and require groups
- v1.4.0 Grouper Web Services
 - Add Member
 - Add or remove grouper privileges
 - Authentication
 - Delete Member
 - Find Groups
 - Find Stems
 - Get grouper privileges

- Get Groups
- Get Members
- Get Memberships
- Group Delete
- Grouper WS Lite - REST input - output XHTML - XML - JSON
- Group Save
- Has Member
- Member change subject
- Stem Delete
- Stem Save
- Web Services FAQ
- WS - Proof of Concept 2007-10-2007
- .NET Clients for Grouper WS
- PHP Clients for Grouper Web Service
- Getting started with hooks
- Grouper Client
- Hibernate and data layer updates
- Hibernate ID's and versioning
- Hooks POC (Proof of concept)
- Useful sample Grouper scripts and queries
- Grouper UI custom authentication example

Home


This page last changed on Oct 13, 2009 by steveo@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Edit access instructions: <http://middleware.internet2.edu/docs/internet2-spaces-instructions-200703.html>

Welcome to the Grouper Groups Management Toolkit Wiki

This wiki space is a companion to the Grouper Website: <http://www.internet2.edu/grouper/>

Grouper Product	Grouper Project
<p>Grouper Product Documentation The Grouper Product space, with the latest software and documentation.</p> <ul style="list-style-type: none">• New! - Grouper v1.4.2 is available.• Intended to house the overview and more technical documents regarding the production release of Grouper.• For the manager, sysadmin, and applications developer• Find the current or previous versions of the Grouper software.• http://www.internet2.edu/grouper/: Grouper's official website•  <p>Grouper Infosheet (PDF)</p>	<p>The Grouper Working Group The Grouper Project space, with the design and development work of the MACE-led Grouper Working Group.</p> <ul style="list-style-type: none">• Intended to house items beyond the convenience of the mailing list.• Contribute your campus' software, documentation, use cases here.• Storage for Member presentations, draft documents, proposals, etc.• <i>Return to the Web:</i> Grouper Working Group Home<ul style="list-style-type: none">◦ Minutes - notes from the WG bi-weekly conference calls◦ WG Final Documents - link coming soon!

NOTE WELL: All Internet2 Activities are governed by the [Internet2 Intellectual Property Framework](#).

Working Group Chair

[Tom Barton](#), University of Chicago

Working Group Flywheel

[Steve Olshansky](#), Internet2

Mailing Lists

To subscribe to Grouper mailing lists, including Grouper-Announce, see the [Mailing Lists](#) page.

Related Internet2 Middleware projects

- [CManage](#)
- [Shibboleth](#)
- [MACE-paccman](#) (privilege and access management)

Development of this software was supported with funding from [Internet2](#), the [University of Chicago](#), the [University of Bristol](#), the [NSF Middleware Initiative](#) (NSF 02-028, Grant No. OCI-0330626), and [JISC](#). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the [National Science Foundation](#) (NSF).

 Questions or comments?  [Contact Information](#)[Contact us](#).

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

About Grouper

This page last changed on Sep 23, 2009 by steveo@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

This page has moved to <http://www.internet2.edu/grouper/about.html>. Please update your bookmarks and links.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)



Credits

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

Grouper Credits


The credits page is now linked from the [About](#) page or direct at <http://middleware.internet2.edu/dir/groups/grouper/credits.html>

 Questions or comments?  [Contact us](#).

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

Grouper Product

This page last changed on Dec 08, 2009 by tbarton@uchicago.edu.

 [Contact us](#) if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Welcome to the Grouper v1.4.2 Product Documentation Space

NEW!! [Grouper v1.4.2](#) is now available!

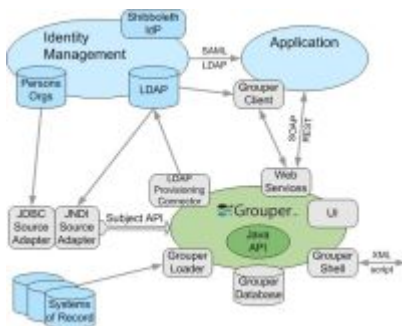
Software Overview

Download - Download the latest product version from the main Grouper Software web site.
Specsheet - Offers operational specifications for the development and deployment of Grouper.
License - Terms under which Grouper is licensed, the Apache 2.0 license.
Archives - Details feature information and downloads of previous Grouper versions.
Glossary - Terms and definitions relevant to Grouper.

[Grouper Trademark Guidelines and Styleguide](#) - Usage policy for the Grouper logo.

Grouper Integration

Graphic illustration of the the range of options for integrating Grouper with existing Identity Management components and with applications.



Systems Administration

Installation & Configuration

Prerequisites - Establish the Grouper environment.
API Installation - Configure the API and integrate with existing identity stores.
UI Installation - Configure, build, and deploy the Grouper User Interface.
WS Installation - Build, secure, and deploy Grouper Web Services.
Initializing Administration of Privileges - The first stem and group you should create.

Tools & Topics for On-Going Administration

GrouperShell - Documentation for the *gsh* command line utility.
LDAPPC - Documentation for the LDAP Provisioning Connector.
Grouper Client - Experimental client for Grouper LDAP and Web Services.
Import/Export Tool - Documentation for the XML Import/Export tool.
Unresolvable Subject Deletion Utility - Documentation for the command line *usdu* tool.

[Bad Membership Finder Utility](#) - Documentation for the *findbadmemberships* command line script.
[Custom Group Types & Fields](#) - What they are and how to create and delete them.

Applications Development

[Grouper UI Customisation Guide](#)*- click here for more information regarding the following documents:

- [Grouper UI Components](#)
- [Architecture](#)
- [Struts Actions and Tiles](#)
- [Customising the Grouper UI](#)
- [Grouper UI Development Environment](#)

[Developer's Guide to Grouper Web Services](#)

[Hooks for 3rd party extensions to the java API](#)

[API & UI v1.4.2 Javadoc](#)

[WS v1.4.2 Javadoc](#)

[Grouper Client v1.4.2 Javadoc](#)

[CVS](#) - manages the versioning of the Grouper source code.
Access the Grouper source code via the web:


- **Connection type:** pserver
- **User:** anoncvs
- **Passwd:** <your email address>
- **Host:** anoncvs.internet2.edu
- **Repository Path:** /home/cvs/i2mi
- **Use default port:** yes
- Access the complete Grouper source code via the command line:

```
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi login
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_1_4_2 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_1_4_2 grouper-ui
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_1_4_2 grouper-ws
```

Bug Reports - Bugs submitted and fixes found here. Note: You will need to create a login with Jira.

Unable to render {include} Couldn't find a space with key: i2miCommon

Archived Documentation

 Archived documentation can be viewed on the [Archives](#) page, bundled in a (.PDF) file under each release version (major- and sub-releases only.)

Community Area

Documents & Presentations - View others' and post your Grouper-related drafts and final works.

Contributions - Share your code, software, documentations, use cases, and other contributions with the Grouper community.

 **Contact us** if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

API Building & Configuration

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Building the Grouper API as of v1.4.1

As of Grouper v1.4.1, the API is now provided both as a binary and source distribution.

To build the source distribution :

```
cd grouper-api-1.4.1
ant dist
```

Testing the API is performed using GrouperShell :



Testing will destroy any pre-existing data in the Groups Registry database

```
bin/gsh.sh -test -all
```

Configuring the Grouper API as of v1.4.1

In this section we describe all of the Grouper API configuration files and important settings.

The Grouper API is distributed with example configuration files with ".example" inserted in the middle of their names. These should be renamed or copied to remove the ".example" substring, or doing a build with ant will do this, or it is already copied in the binary distribution. e.g. for grouper.properties, the example file is grouper.example.properties.

Section	Configuration File	Purpose
Database-Related Settings and Procedures	grouper.hibernate.properties	integrating the Grouper API with the database that will house your Groups Registry
Configuration of Source Adapters	sources.xml	integrating the Grouper API with chosen identity sources
Grouper properties	grouper.properties	defaults for Grouper privileges, enabling identified external users to act with elevated root-like privilege, changing the display name for internal subjects
Logging	log4j.properties, grouper.properties	logging
Loader	grouper-loader.properties	auto-load memberships from external sql sources. or run other jobs

Database-Related Settings and Procedures

Database Driver Location

Place the jar file containing the JDBC driver for your database in the lib/custom/ directory. The Grouper v1.4 package includes the JDBC driver for HSQLDB v1.7.2.11. Sample JDBC drivers are located in lib/jdbcSample (e.g. for Oracle, MySQL, and PostgreSQL).

General Property Settings

Grouper uses Hibernate to persist objects in the Groups Registry. Database-specific settings are configured in conf/grouper.hibernate.properties, which has pre-populated examples for HSQLDB, MySQL, Oracle, and PostgreSQL.

Required properties are:

Property Name	Purpose
hibernate.connection.driver_class	JDBC driver classname
hibernate.connection.url	JDBC URL for the database
hibernate.connection.username	database user
hibernate.connection.password	database user's password Note, you can also put a filename of the encrypted password
hibernate.dialect	classname of a Hibernate dialect, for setting platform specific features. Choices are listed here

You may need to refer to your database support person to determine these required properties.

Detailed Hibernate configuration documentation is available [here](#).

MySQL Transaction Support

If you want transactions to work (i.e. when doing a unit of work in grouper, it either all completes or none), which is definitely recommended, though not required, your mysql table format needs to be transactional, e.g. innodb, which is not the default (myisam is the default). One way to enable innodb in mysql is with this line in the my.cnf: default-storage-engine=innodb

Database Whitelists and Blacklists

Some database operations (such as dropping tables or recreating data during tests) require confirmation of a prompt asking whether or not to continue. It is possible to automatically allow or deny these database operations in conf/grouper.properties :

```
# whitelist (allow) and blacklist (deny) for db data or object deletes.
# if a listing is in the whitelist (allow), it will be allowed to delete db
# if a listing is in the blacklist (deny), it will be denied from deleting db
# multiple inputs can be entered with .0, .1, .2, etc. These numbers must be sequential, starting with 0
db.change.allow.user.0=grouper3
db.change.allow.url.0=jdbc:mysql://localhost:3306/grouper3
db.change.allow.user.1=grouper1
db.change.allow.url.1=jdbc:mysql://localhost:3306/grouper1

db.change.deny.user.0=grouper2
db.change.deny.url.0=jdbc:mysql://localhost:3306/grouper2
```

Database Initialization Procedure

Database initialization is performed using the GrouperShell.



Initializing the database will destroy any pre-existing data

To initialize the Groups Registry and install tables, populate default group types and fields, and create the root naming stem :

```
bin/gsh.sh -registry -check -runscript
```

To re-initialize the Groups Registry (e.g. after running junit tests) :

```
bin/gsh.sh -registry -reset
```

To see all options :

```
bin/gsh.sh -registry
```

Configuration of Source Adapters

Grouper uses [\[Subject API\]](#) compliant "source adapters" to integrate with external identity stores. "Subjects" are the objects housed there that are presented to Grouper for management vis-à-vis group membership and Grouper privileges. These may represent people, other groups, computers, applications, services, most anything for which you manage identity. With the exception of Grouper groups, Grouper treats all subjects opaquely. See the [\[Subject API\]](#) documentation for further background and details concerning subjects, source adapters, and other aspects of the Subject API.

Each source adapter connects with a single back-end store using JDBC or JNDI. Grouper makes no specific assumptions about the schema of any subject types. Instead, sections of the configuration file, `grouper/conf/sources.xml`, declare the details of how to connect with each back-end store, the identifier(s) to be used for the subjects it contains, how to select and search for subjects, and which subject attributes should be made available to Grouper.

Grouper v1.4.X relies on v0.4.2 of the Subject API. Please refer to the section on [\[Subject API v0.3.1\]](#) for detailed configuration information.

Three source adapter classes are included in the Grouper API v1.3.0 package. `JDBCSourceAdapter` and `JNDISourceAdapter` classes are included in `subject-0.3.1.jar`, and `GrouperSourceAdapter` is built along with the Grouper API. Every Grouper API deployment MUST include a `*source*` element in `grouper/conf/sources.xml` for the `GrouperSourceAdapter` so that Grouper can refer to its own groups in the same manner as other subjects.

See the `sources.example.xml` for example usages of the `sources.xml`

Choosing Identifiers for Subjects

Identifiers and their management can get complicated. They can be revoked or not, re-assigned or not, lucent or opaque, etc. Depending on such characteristics, a given identifier might be a good or bad choice to use in the context of managing the identified subject's group memberships.

For example, a username is often lucent - easily remembered by the person to whom it is associated. But it may also be revokable, meaning that it no longer refers to that person (perhaps they have a new one), or even re-assignable, meaning that it might refer to some other person at a later time. If a username is used to record membership, username changes must trigger corresponding membership changes. A username is better suited to authentication than it is to indicating membership.

On the other hand, an opaque registryID (machine, not human, readable) that never changes is great for membership, but lousy for authentication - it might not even be known by the person to whom it is associated. How would I identify myself to Grouper if I wished to opt-in to a list or manage a group?

Grouper accommodates subject identifier issues in two ways. First, it maintains UUIDs for every subject and group within the Groups Registry. These are never exposed by the API, but are associated with

externally supplied subject identifiers within the Groups Registry. This approach allows the identifier associated with a given subject to be changed without any need to change actual memberships.

Second, by relying on the Subject API, Grouper is able to lookup subjects that are presented with an identifier in one namespace and obtain identifiers in other namespaces for that subject. That means that it can translate a username into a registryID, for example. So, when a user authenticates to an application using the Grouper API, that application can use the Subject API to fetch an identifier for the person chosen by the site for use in memberships. Similarly, when a membership in the Groups Registry is to be expressed elsewhere, the identifier used for group members can be translated by a provisioning connector by use of the Subject API into one that is suitable in the provisioned context.

Grouper Properties

All configuration of Grouper properties detailed in this section occur in the grouper/conf/grouper.properties file. Look in the grouper.example.properties file for the more obscure settings. Common settings are listed below

This setting describes the env that grouper is running, e.g. used in the daily report from the loader which

```
grouper.env.name = production
```

If grouper should auto init the registry if not init'd (i.e. insert the root stem, built in fields, etc)

```
registry.autoinit = true
```

If grouper should try and detect and log configuration errors on startup, in general this should be true, unless the output is too annoying or if it is causing a problem

```
configuration.detect.errors = true
```

If groups like the wheel group should be auto-created for convenience (note: check config needs to be on)

```
configuration.autocreate.system.groups = false
```

Auto-create groups (increment the integer index), and auto-populate with users (comma separated subject ids) to bootstrap the registry on startup (note: check config needs to be on). The next group would end in 1, then 2, etc

```
configuration.autocreate.group.name.0 = etc:uiUsers  
configuration.autocreate.group.description.0 = users allowed to log in to the UI  
configuration.autocreate.group.subjects.0 = johnsmith
```

By default, anyone with admin rights on a group can edit the types or attributes. Specify types (and related attributes) which are wheel only, or restricted to a certain group

```
security.types.typeName.wheelOnly = true  
security.types.grouperLoader.wheelOnly = true  
  
#security.types.typeName.allowOnlyGroup = etc:someAdminGroup
```

If you don't want to be prompted for DDL changes in certain databases (e.g. dev), list them here: Whitelist (allow) and blacklist (deny) for db data or object deletes, without prompting the user to confirm

If a listing is in the whitelist (allow), it will be allowed to delete db

If a listing is in the blacklist (deny), it will be denied from deleting db

Multiple inputs can be entered with .0, .1, .2, etc. These numbers must be sequential, starting with 0

```
db.change.allow.user.0=grouper3  
db.change.allow.url.0=jdbc:mysql://localhost:3306/grouper3  
db.change.allow.user.1=grouper1  
db.change.allow.url.1=jdbc:mysql://localhost:3306/grouper1  
  
db.change.deny.user.0=grouper2  
db.change.deny.url.0=jdbc:mysql://localhost:3306/grouper2
```

There is a substantial section for [include/exclude and requireGroups](#). These are group types which help you create composite groups to manage include/exclude lists for groups (especially useful for grouper loader provisioned groups), or groups which require memberships in other groups (e.g. activeStaff). See the grouper.example.properties file if you want to customize things, but to enable, set these:

```
grouperIncludeExclude.use = false
grouperIncludeExclude.requireGroups.use = false
```

Here are some requireGroups (increment the 0 to add more):

```
#grouperIncludeExclude.requireGroup.name.0 = requireActiveStudent
#grouperIncludeExclude.requireGroup.attributeOrType.0 = attribute
#grouperIncludeExclude.requireGroup.group.0 = school:community:activeStudent
#grouperIncludeExclude.requireGroup.description.0 = If value is true, members of the overall group must be
an active student (in the school:community:activeStudent group). Otherwise leave this value not filled in.
```

[Hooks](#) are ways to plugin in your own java code to affect how Grouper does its logic. You can register multiple classes for one hook base class by comma separating the hooks implementations. You can also register hooks at runtime with: `GrouperHookType.addHookManual("hooks.group.class", YourSchoolGroupHooks2.class);`

See the grouper.example.properties for the full list, here are two examples:

```
#implement a group attribute hook by extending edu.internet2.middleware.grouper.hooks.AttributeHooks
#hooks.attribute.class=edu.yourSchool.it.YourSchoolGroupHooks,edu.yourSchool.it.YourSchoolGroupHooks2

#implement a group hook by extending edu.internet2.middleware.grouper.hooks.GroupHooks
#hooks.group.class=edu.yourSchool.it.YourSchoolGroupHooks,edu.yourSchool.it.YourSchoolGroupHooks2
```

You can validate group attributes via regex (see grouper.example.properties for more info) (increment the 0 to add more)

```
#group.attribute.validator.attributeName.0=extension
#group.attribute.validator.regex.0=^[a-zA-Z0-9]+$
#group.attribute.validator.vetoMessage.0=Group ID '$attributeValue$' is invalid since it must contain only
alpha-numeric
```

Database structure data definition language (DDL) settings (see grouper.example.properties for full list)

```
# if you want to not create the subject tables (grouper examples for unit testing),
# then set this to true
ddlutils.exclude.subject.tables = false

# set the path where ddl scripts are generated (they will be uniquely named in this directory).
# if blank, the directory used will be the current directory
ddlutils.directory.for.scripts = ddlScripts

# during schema export, should it install grouper data also or not. e.g. insert the root stem, default true
ddlutils.schemaexport.installGrouperData = true

# when grouper starts, should it shut down if not right version?
ddlutils.failIfNotRightVersion = true

# after you have converted id's, and are happy with the conversion of removing the uuid col,
# this will remove the backup uuid cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupUuidCols = false

# after you have converted field id foreign keys, and are happy with the conversion of removing the attribute
name,
# membership list name, and type cols,
# this will remove the backup field name/type cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupFieldNameTypeCols = false

# this is the schema ddlutils uses to query metadata with jdbc. usually this can be omitted,
# and it defaults to your database loginid, however, in postgres, it can be different, so enter here
#ddlutils.schema = public
```

```
#if you are running a DB that supports them, but you dont want them, disable views or object comments here
(defaults to false)
ddlutils.disableComments = false
ddlutils.disableViews = false
```

Mail settings (optional, e.g. for daily report form loader)

```
#smtp server is a domain name or dns name, must be simple clear text smtp with no authentication
#mail.smtp.server = whatever.school.edu

#leave blank if unauthenticated
#mail.smtp.user =

#leave blank if unauthenticated
#mail.smtp.pass =

#this is the default email address where mail from grouper will come from
#mail.from.address = noreply@school.edu

#this is the subject prefix of emails, which will help differentiate prod vs test vs dev etc
#mail.subject.prefix = TEST:
```

Default privileges

Grouper requires that all subjects must be explicitly granted access or naming privileges (cf. [Glossary](#)), with one caveat. There is a special "subject" internal to Grouper called the ALL subject, which is a stand-in for any subject. The ALL subject can be granted a privilege in lieu of assigning that privilege explicitly to each and every subject.

When a new group or naming stem is created, any of its associated privileges can be granted by default to the ALL subject. This is configured by a series of properties in `grouper.properties`, one per privilege. If a property has the value "true" then ALL is granted that privilege by default when a group or naming stem is created. Otherwise it is not, and hence no subject has that privilege by default.

Property Name	Value in Grouper v1.3.1 Distribution
groups.create.grant.all.admin	false
groups.create.grant.all.optin	false
groups.create.grant.all.optout	false
groups.create.grant.all.update	false
groups.create.grant.all.read	true
groups.create.grant.all.view	true
stems.create.grant.all.create	false
stems.create.grant.all.stem	false

Super-user Privileges

Grouper has another special "subject" called `GrouperSysAdmin` that acts as a super-user. `GrouperSysAdmin` is permitted to do everything - the privilege system is ignored for that special subject. Grouper can be configured to consider all members of a distinguished group to be able to act as super-users, much as the "wheel" group does in BSD Unix. Two properties control this behavior:

Property Name	Description
groups.wheel.use	"true" or "false" to enable or disable this capability.
groups.wheel.group	The group name of the group whose members are to be considered security-equivalent to GrouperSysAdmin. By default this is: etc:sysadmingroup

The Grouper UI enables users that belong to the wheel group to choose when to act with the privileges of GrouperSystem and when to act as their normal selves.

Changing the display name of GrouperAll and GrouperSystem

Before version 1.3.0 the Grouper UI referred to *EveryEntity* as GrouperAll and *GrouperSysAdmin* as GrouperSystem. As of version 1.3.1 the name attribute of GrouperAll and GrouperSystem can be set through the properties below.

Property Name	Description
subject.internal.grouperall.name	The name to use for GrouperAll instead of EveryEntity
subject.internal.groupersystem.name	The name to use for GrouperSystem instead of GrouperSysAdmin

If you choose not to use the defaults you will have to update the UI nav.properties file to ensure consistency e.g. subject.privileges.from-grouperall=inherits from EveryEntity

Changing default privilege caching

Grouper includes three `PrivilegeCache` implementations:

- NoCachePrivilegeCache - No caching performed
- SimplePrivilegeCache - Caches results but flushes all cached entries upon any update
- SimpleWheelPrivilegeCache - Same as 'SimplePrivilegeCache' but with better support for using a wheel group.

The privileges.access.cache.interface and privileges.naming.cache.interface properties can be set to determine the privilege caching regimen. The default is edu.internet2.middleware.grouper.NoCachePrivilegeCache.

Using a privilege management system external to Grouper

Grouper's internal security implementation relies on two java interfaces, one for Naming Privileges and another for Access Privileges. Grouper ships with classes that implement these interfaces, but 3rd parties are free to supply their own and so manage Grouper privileges using a privilege management system external to Grouper. Two properties declare the java classes that Grouper will use to implement these interfaces:

Property Name	Description
privileges.access.interface	classname of the java class that implements the Access Interface
privileges.naming.interface	classname of the java class that implements the Naming Interface

It is not clear that this has been taken advantage of... the internal privilege management is the one usually used

Logging

Logging is configured in the grouper/conf/log4j.properties configuration file. By default Grouper will write event log information to grouper/grouper_event.log, error logging to grouper/grouper-error.log, and debug logging, if enabled, to grouper/grouper-debug.log. The log4j configuration can be adjusted to control the verbosity, type and output of Grouper's logging.

In addition, there are several configuration parameters in grouper/conf/grouper.properties that may be adjusted to control the logging of effective membership modifications in the event log.

```
# Control whether the addition and deletion of effective groups memberships
# are logged in the event log. If using the _GrouperAccessAdapter_ this
# will include granted and revoked access privileges.
memberships.log.group.effective.add = true
memberships.log.group.effective.del = true

# If using _GrouperNamingAdapter_, control whether the granting and
# revoking of effective naming privileges are logged in the event log.
memberships.log.stem.effective.add = true
memberships.log.stem.effective.del = true
```

Loader

The [Grouper - Loader](#) is a daemon command line process which run jobs that load/remove memberships of groups based on results from a sql query. There is a grouper-loader.properties file to configure. The common settings are described below, see the grouper-loader.example.properties for descriptions of all settings.

If you want grouper to make sure the loader type and attributes exist if not there, set this. Otherwise you need to add the type and attributes yourself with GSH.

```
# auto-add grouper loader types and attributes when grouper starts up if they are not there
loader.autoadd.typesAttributes = false
```

If most of your loader queries come from one subject source, you can set the default subject source here, so your loader queries only need to return SUBJECT_ID and not the source id also:

```
default.subject.source.id =
```

If all of your queries are run against your grouper db credentials (e.g. if you have other schemas on the same DB to query), you dont have to configure DB connections. If you have other databases to query (e.g. an external data warehouse, etc), you can configure the db credentials here. The name here is "warehouse", use different names for different connections



```
db.warehouse.user = mylogin
#note the password can be stored encrypted in an external file
db.warehouse.pass = secret
db.warehouse.url = jdbc:mysql://localhost:3306/grouper
db.warehouse.driver = com.mysql.jdbc.Driver
```

If you want to use the Grouper daily report, configure in the grouper-loader.properties (and the mail settings described above in grouper.properties). This is a daily email that gets sent to you about your grouper health, including status about all the loader jobs in the last day.

```
#quartz cron-like schedule for daily grouper report, the default is 7am every day: 0 0 7 * * ?
#leave blank to disable this
daily.report.quartz.cron =

#comma separated email addresses to email the daily report, e.g. a@b.c, b@c.d
```

daily.report.emailTo =

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Getting Started

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Getting started with the Grouper API Binary Distribution v1.4.0

This page contains instructions for installing and configuring a new Grouper v1.4.0 installation.

- Meet required [prerequisites](#) and download the Grouper API binary release
- [API Building & Configuration](#). Config files are in the conf/ directory. Config files ending in ".example" are templates and should not be modified. Config files not ending in ".example" are customizable
- Initialize your database (by default HSQLDB) by running **gsh -registry -check -runscript**

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh -registry -check -runscript
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\conf
Using JAVA:         "c:\dev_inst\java\bin\java"
using MEMORY:       64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\logs\grouper_error.log, at min
level WARN
for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: sa@jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist\run/
grouper;create=true
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider

Based on grouper.properties: ddlutils.schemaexport.installGrouperData=true
(note, might need to type in your response multiple times (Java stdin is flaky))
(note, you can whitelist or blacklist db urls and users in the grouper.properties)
Are you sure you want to schemaexport all tables (dropThenCreate=F,writeAndRunScript=T) in db user 'sa', db
url 'jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist\run/grouper;create=true'? (y|n):
y
Continuing...
Grouper ddl object type 'Grouper' has dbVersion: 0 and java version: 12
Grouper ddl object type 'Subject' has dbVersion: 0 and java version: 1
Grouper database schema DDL requires updates
(should run script manually and carefully, in sections, verify data before drop statements, backup/export
important data
before starting, follow change log on confluence, dont run exact same script in multiple envs - generate a new
one for
each env),
script file is:
C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\ddlScripts\grouperDdl_20081114_02_00_20_771.sql
Script was executed successfully
```

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>
```

- Verify the installation was successful by examining log files in the logs/ directory.
- When using HSQLDB, a second attempt is sometimes necessary: **gsh -registry -check -runscript**
- It is now possible to run [GrouperShell \(gsh\)](#) and [Initialize Administration of Privileges](#)

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
```

Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\../conf
Using JAVA: "c:\dev_inst\java\bin\java"
using MEMORY: 64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: console, at min level WARN for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: sa@jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist/run/grouper;create=true
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider

2008-11-14 02:12:06,041: [main] WARN ApiConfig.printConfigOnce(189) - Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
Type help() for instructions
gsh 0% addRootStem("penn", "penn");
stem: name='penn' displayName='penn' uuid='b693f8f1-9763-44ae-a984-be25e0b6ea0f'
gsh 1% addGroup("penn", "faculty", "faculty");
group: name='penn:faculty' displayName='penn:faculty' uuid='fbf3646f-eb10-47c1-9cc0-d32b7049a336'
gsh 2%

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Binary Release

This page last changed on Nov 22, 2008 by [mchyer](#).

The grouper binary release is a package which does not need to be built with ant.

Contents

- Grouper jar
- All jars that grouper requires
- Configuration files (from examples, need to be customized)
- Example jdbc jars (for Oracle, MySQL, Postgres, hsql)
- Grouper javadoc
- Grouper shell (GSH) script used to interact with grouper from the command line

Using

- You can use the binary release of grouper to get started with grouper, or to upgrade a current installation. Note, if you want to run the UI, you need to build it, and grouper itself
- Unzip the binary release
- Run from command prompt: `bin/gsh -registry -runscript`
 - This will create the database in hsql
- Run from command prompt: `bin/gsh`
 - This will run [Grouper Shell](#)
- Type exit, edit the `conf/grouper.hibernate.properties` with the hibernate dialect, driver, url, user, and pass of a real database (preferably oracle, mysql, or postgres)
- Run from command prompt: `bin/gsh -registry -runscript`
 - This will create the database in hsql
- Run from command prompt: `bin/gsh`
 - This will run [Grouper Shell](#)
- Go through the `conf/grouper.properties`, `conf/sources/xml`, etc and look at options available and customize

LDAPPC

This page last changed on Oct 29, 2009 by tzeller@idp.protectnetwork.org.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

LDAPPC - LDAP Provisioning Connector

LDAPPC provisions group and membership information contained in the Groups Registry to a site's LDAP directory service.

As of Grouper v1.4.1, LDAPPC is included in the Grouper API, and is run using [GrouperShell](#).

Documentation for the current version is being updated.

[Usage](#)

[Configuration](#)

Documentation for previous versions is available at <https://wiki.internet2.edu/confluence/display/i2miCommon/Ldappc>

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

LDAPPC Annotated Configuration

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

LDAPPC Annotated Configuration as of v1.4.1

• **<ldappc> - Provisioning Configuration**

```
<ldappc>
  <grouper ... />
  <source-subject-identifiers ... />
  <ldap ... />
</ldappc>
```

The LDAPPC provisioning configuration consists of three elements : Grouper, LDAP subject, and LDAP connection parameters.

◦ **<grouper> - Grouper Configuration**

```
<grouper>
  <grouper-queries ... />
  <groups ... />
  <memberships ... />
</grouper>
```

The Grouper configuration includes the selection of groups to be provisioned, and optional group and membership provisioning.

- **<grouper-queries> - Select Groups to be Provisioned**

```
<grouper-queries>
  <subordinate-stem-queries ... />
  <attribute-matching-queries ... />
</grouper-queries>
```

The groups to be provisioned may be selected by stem, attribute value, or both. Currently, it is not possible to exclude a group that otherwise matches the selection criteria from being provisioned.

- **<subordinate-stem-queries> - Select Groups to be Provisioned by Stem**

```
<subordinate-stem-queries>
  <stem-list>
    <stem>uc:faculty:art</stem>
    <stem>uc:faculty:math</stem>
  </stem-list>
```

```
</subordinate-stem-queries>
```

All groups subordinate to any of the given stems are selected for provisioning.

- **<attribute-matching-queries> - Select Groups to be Provisioned by Attribute**

```
<attribute-matching-queries>
  <attribute-list>
    <attribute name="attr1" value="value1" />
    <attribute name="attr2" value="value2" />
  </attribute-list>
</attribute-matching-queries>
```

All groups having the given attribute value(s) are selected for provisioning.

- **<groups> - Provision Groups**

```
<groups structure="flat"
  root-dn="ou=grouper,ou=groups,dc=example,dc=edu"
  ldap-object-class="groupOfNames"
  ldap-rdn-attribute="cn"
  grouper-attribute="name"
  initial-cache-size="170003">
  <group-members-dn-list ... />
  <group-members-name-list ... />
  <group-attribute-mapping ldap-object-class="" ... />
</groups>
```

The optional <groups> element defines how entries and DN's for provisioned groups are created.

<groups>	
structure	The group DN naming structure may be either "flat" or "bushy". A flat structure provisions all groups into the same root DN using the name of the group as the RDN, e.g. cn=stem:child-stem:group-name,root-dn. A bushy structure will provision groups hierarchically, e.g. cn=group-name,ou=child-stem,ou=stem,root-dn.
root-dn	The DN of the entry used as the root of the provisioned groups
initial-cache-size	Optional attribute specifying the initial size of the group cache. Setting this larger than the likely number of groups to be provisioned should improve performance.
ldap-object-class	Defines the LDAP object class used to create each provisioned group. If this object class has required attributes not populated by this provisioning process, then an error will occur.

ldap-rdn-attribute	Defines the attribute in the ldap-object-class used as the RDN. This value may not be "ou" in order to prevent naming collisions between stems and groups when the structure is "bushy".
grouper-attribute	Required when the structure is flat. Defines whether the "id" or "name" attribute value of the group is to be used for the value of the ldap-rdn-attribute.

- **<group-members-dn-list> - Provision Member DNs**

```
<group-members-dn-list
  list-attribute="member"
  list-object-class="groupOfNames"
  list-empty-value="" />
```

If defined, provisioned groups will include member DNs.

<grouper-members-dn-list	
list-attribute	Defines the LDAP attribute in which to store member DNs.
list-object-class	Optional. Defines the LDAP object class the group entry must have to support the list-attribute. Please note that if this object class has required attributes not populated by this provisioning process, then an error may occur.
list-empty-value	Optional. Defines the value of the list-attribute if no member DNs are stored there. If list-attribute is optional (i.e., a MAY attribute), this value is most likely not needed. If list-attribute is required (i.e., a MUST attribute), then this value should be defined.

- **<group-members-name-list> - Provision Member Names**

```
<group-members-name-list
  list-attribute="hasMember"
  list-object-class="eduMember"
  list-empty-value="">
  <source-subject-name-mapping>
    <source-subject-name-map source="sourceA" subject-attribute="userid" />
    <source-subject-name-map source="sourceB" subject-attribute="userid" />
  </source-subject-name-mapping>
</group-members-name-list>
```

If defined, provisioned groups will include member names.

<grouper-members-name-list>

list-attribute	Defines the LDAP attribute in which to store member names.
list-object-class	Optional. Defines the LDAP object class the group entry must have to support the list-attribute. Please note that if this object class has required attributes not populated by this provisioning process, then an error may occur.
list-empty-value	Optional. Defines the value of the list-attribute if no member DNs are stored there. If list-attribute is optional (i.e., a MAY attribute), this value is most likely not needed. If list-attribute is required (i.e., a MUST attribute), then this value should be defined.

The <source-subject-name-mapping> element contains one or more <source-subject-name-map> elements, which defines the subject attribute containing the subject's name.

<source-subject-name-map>	
source	Source ID
subject-attribute	The Subject attribute containing the Subject's name

<group-attribute-mapping> - Provision Group Attributes

```
<group-attribute-mapping ldap-object-class="">
  <group-attribute-map
    group-attribute="aci"
    ldap-attribute="aci"
    ldap-attribute-empty-value="" />
</group-attribute-mapping>
```

Optionally, group attributes may be provisioned.

<group-attribute-mapping>	
ldap-object-class	Optional. Defines the LDAP object class the group entry must have to support the attribute mapping. Please note that if this object class has required attributes not populated by this provisioning process, then an error may occur.

The <group-attribute-mapping> element contains one or more <group-attribute-map> elements, which map Grouper attributes to LDAP.

<group-attribute-map>	
group-attribute	The Grouper attribute name.

ldap-attribute	The LDAP attribute name.
ldap-attribute-empty-value	Optional. Defines the value to be placed in the ldap-attribute if no values are stored there. If ldap-attribute is optional (i.e., a MAY attribute), this value is most likely not needed. If ldap-attribute is required (i.e., a MUST attribute), then this value should be defined.

<membership> - Provision Membership

```

<memberships>
  <member-groups-list
    list-object-class="eduMember"
    list-attribute="isMemberOf"
    naming-attribute="name"
    list-empty-value=""
    temporary-directory="" />
  </memberships>

```

In addition to provisioning groups, LDAPPC may provision memberships. The optional <memberships> element contains one <member-groups-list> element, which defines the LDAP attribute of member entries containing the groups of which they are a member.

<member-groups-list	
list-object-class	Optional. Defines the LDAP object class the Member's entry must have to support the group list. Please note that if this object class has required attributes not populated by the provisioning process, then an error may occur.
list-attribute	Defines the LDAP attribute in which to store groups.
naming-attribute	The Grouper attribute used to create the list of groups for a member.
list-empty-value	Optional. Defines the value to be placed in the list-attribute if no values are stored there. If list-attribute is optional (i.e., a MAY attribute), this value is most likely not needed. If list-attribute is required (i.e., a MUST attribute), then this value should be defined.
temporary-directory	Optional. Defines the file system directory in which temporary files will be written. Defaults to the current directory.

<source-subject-identifiers> - Finding Subjects in the Directory

```
<source-subject-identifiers>
  <source-subject-identifier
    source="jdbc"
    subject-attribute="id"
    initial-cache-size="350007">
    <ldap-search
      base="ou=people,dc=example,dc=edu"
      scope="onelevel_scope"
      filter="(&(examplePersonId=\{0\})(objectclass=examplePerson)))" />
    </source-subject-identifier>
  </source-subject-identifiers>
```

The <source-subject-identifiers> element contains one or more <source-subject-identifier> elements, which defines for a Source the Subject attribute and LDAP search parameters used to lookup Subjects in the directory.

<source-subject-identifier>	description
initial-cache-size	The initial cache size to cache subject DNs by subject ID. Specifying a larger number than the number of subjects should give better performance. Optional.
source	Subject Source ID
subject-attribute	The name of the Subject attribute. If a value other than "id" (the subject ID) is specified, performance may suffer as an extra Subject lookup will be performed. It is strongly recommended that the subject ID be in the subject's directory object and that it be indexed.

Each <source-subject-identifier> element contains exactly one <ldap-search> element.

<ldap-search>	description
base	The base DN of the context or object to search.
scope	Either " object_scope ", " onelevel_scope ", or " subtree_scope ". The JNDI scope constants are defined in javax.naming.SearchControls . For most flat people branches, "onelevel_scope" is a good choice.
filter	The string "{0}" in the search filter will be replaced by the value of the Subject's attribute defined by subject-attribute in the <source-subject-identifier> element.

◦ **<ldap> - LDAP Connection Parameters**

```
<ldap>
<context>
<parameter-list>
  <parameter name="initial_context_factory" value="com.sun.jndi.ldap.LdapCtxFactory" />
  <parameter name="provider_url" value="ldaps://directory.example.edu:636" />
  <parameter name="security_authentication" value="simple" />
  <parameter name="security_principal" value="cn=ldappc-agent,ou=agents,dc=example,dc=edu"
/>
  <parameter name="security_credentials" value="GetFromPropertiesFile" />
</parameter-list>
</context>
</ldap>
```

The <ldap> element contains the information necessary to connect to the LDAP directory being provisioned.

initial_context_factory	fully qualified class name of a context factory
provider_url	LDAP URL
security_authentication	"none", "simple", or "strong"
security_principal	authentication DN
security_credentials	authentication password, or GetFromPropertiesFile

Specific to LDAPPC, if the value of "security_credentials" is "GetFromPropertiesFile" then the property "securityCredentials" from ldappc.properties is used. This allows ldappc.xml to be maintained in version control while the password is stored separately.

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

LDAPPC Usage

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

LDAPPC Usage as of v1.4.1

Groups and memberships may be provisioned independently.

The configuration file, e.g. ldappc.xml, contains the criteria for selecting which groups and memberships should be provisioned.

To run LDAPPC from the command line :

```
bin/gsh.sh -ldappc <args>
```



For example, the following will provision groups and memberships using the Grouper subject GrouperSystem, and will poll every 60 seconds after the last run.

```
bin/gsh.sh -ldappc -subject GrouperSystem -groups -memberships -interval 60
```

Key	Value	Description
no arguments		Display the following list of available arguments to standard output.
-subject	<i>subjectId</i>	The SubjectId used to establish Grouper API sessions. If any arguments are used, then "-subject <i>subjectId</i> " is required.
-groups		(Optional) Provision groups.
-memberships		(Optional) Provision memberships.
-lastModifyTime	<i>lastModifyTime</i>	(Optional) DateTime representation to select only objects changed since then. Allowed format: yyyy-MM-dd_hh:mm:ss or yyyy-MM-dd_hh:mm or yyyy-MM-dd_hh or yyyy-MM-dd. Missing hours, minutes, or seconds are replaced with zeros and the appropriate delimiters.
-interval	<i>interval</i>	(Optional) Number of seconds between polling intervals. If omitted, only one provisioning cycle is performed.
-configManager	<i>configMgr</i>	(Optional) Location of substitute configuration file, other than default, ldappc.xml.

If the "-configManager" option is present, the configuration is read from the specified file, otherwise it will be read from the ldappc.xml and ldappc.properties files in the class path. Since the conf directory is

included in the class path, that is where the ldappc.xml and ldappc.properties files are generally placed. The ldappc.properties file is optional and contains the LDAP password.

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper UI Architecture

This document is current as of the v1.2.0 release.

1. [Introduction](#)
2. [The Architecture and How It Matches the Requirements](#)
 - 2.1. [Core Technology](#)
 - 2.2. [Framework](#)
 - 2.3. [Internationalization](#)
 - 2.4. [Extensibility](#)
 - 2.5. [Separation of Core Grouper UI Code from Site-Specific Code](#)
 - 2.6. [Accessibility](#)
3. [Implementation Details](#)
 - 3.1. [Grouper UI Out of the Box](#)
 - 3.2. [Grouper UI Tailored for the University of Bristol \(UoB\)](#)
 - 3.3. [Supporting an Additional Language](#)
 - 3.4. [Overloading in ResourceBundles](#)
 - 3.4.1. [How It Works](#)
 - 3.4.2. [Multiple UIs One Instance](#)
 - 3.4.3. [Chaining ResourceBundles](#)
 - 3.5. [Overloading Struts Config Elements](#)
 - 3.6. [Additional Hooks to Extend the Grouper UI](#)
 - 3.7. [Page Composition Using Tiles](#)
 - 3.8. [Content Composition Using Tiles](#)
 - 3.9. [Dynamic Tiles](#)
 - 3.10. [JSP Tag Libraries](#)

1. Introduction

This document describes the proposed architecture for the Grouper UI. The architecture aims to satisfy the following general requirements:

1. **Core technology**
Java-based web application to complement the Grouper API which is also Java-based
2. **Web application framework**
Use of established and well-documented framework not tied to particular platform
3. **Internationalization**
Designed to facilitate sites who wish to use a language other than english and in such a way that an institution can offer more than one language
4. **Extensibility**
Institutions will likely want to tailor the UI to reflect local business rules and available meta data
5. **Separation of core Grouper UI code from site-specific code**
Separation is important to ensure that local changes do not require modification of Grouper sources thus making upgrades much easier
6. **Accessibility**
Content must be accessible through use of DOM /CSS layouts rather than table layouts

2. The Architecture and How It Matches the Requirements

... Where relevant links are provided to some of the concepts and their justifications.

2.1. Core Technology

The Grouper UI will use the standard Servlet API (Version 2.3) which includes [Java Server Pages](#) (JSP). The servlet API is supported by many J2EE application servers, e.g., [BEA Weblogic](#), [IBM WebSphere](#) and [Sun Java System Application Server](#), as well as some open source application servers (which may not provide complete J2EE support) such as [Resin](#) and [Apache Tomcat](#).

The UI will be developed using J2SE 1.4 and Tomcat 4.1.x. Given the widespread support for the Servlet API it is expected that the UI distribution will work with any compliant application server, however, we will depend on early adopters to feedback any issues.

2.2. Framework

[Apache Struts](#) has been chosen as the web application framework due to its wide acceptance and use in this arena. It is an implementation of the [Model-View-Controller](#) (MVC) design pattern which encourages separation of business logic from the presentation layer. This document will also show how the Struts framework can help satisfy other requirements indicated in the Introduction.

Struts supports a templating engine called [Tiles](#) which allows common UI elements to be defined once and re-used as often as needed.

2.3. Internationalization

Struts supports [internationalization](#) through the use of message resources and [locale specific tiles](#) (page 48). Since Struts was developed, the [Java Standard Tag Library](#) (JSTL) has been released, and this offers an alternative way of specifying message resource. The JSTL method will be adopted for this project.

2.4. Extensibility

Struts is configured through XML descriptors. By adding configuration elements to the XML descriptor, new actions can be created, or existing ones modified. The use of Tiles offers a great deal of freedom to redefine individual templates as deemed necessary.

2.5. Separation of Core Grouper UI Code from Site-Specific Code

Struts supports multiple configuration files. If two configuration elements with the same name are loaded the one loaded last is used. This allows new configuration elements to be added and existing ones to be overridden without actually modifying the Struts configuration file supplied in the Grouper UI distribution.

The same is true of Tiles definition descriptors.

Text for each additional language is contained in a separate properties file. Java [ResourceBundles](#) automatically look in the correct file based on the currently specified locale.

The Grouper UI will include a [cascading style sheet](#) (CSS) file which governs the visual appearance of the UI. We will provide a way of importing additional site-specified CSS files which can add additional style definitions or override existing definitions.

2.6. Accessibility

The Grouper UI will be tested for [compliance](#).

3. Implementation Details

This section will discuss important features of the directory structure and configuration files used by the default Grouper UI distribution, and how institutions can tailor the UI to meet their own requirements. It focuses on the deployed directory structure. A working knowledge of Struts* will aid understanding.

*A few minor changes to some Struts source code has been required to make everything work as described below.

3.1. Grouper UI Out-of-the-Box

grouper	The web application document root.
grouper	Grouper specific stylesheets
images	Grouper specific images
i2mi	Internet 2 specific stylesheets
images	Internet 2 specific images
WEB-INF	Web application data which cannot be accessed directly by a web browser. Includes the web.xml descriptor which defines the web application. The Struts ActionServlet is configured here.
classes	Java classes and resources placed here are available to the web application through its classloader. Grouper UI code will go here
jsp	Java Server Pages placed here cannot be referenced directly from a web browser / client, however, they can be accessed by servlets within this web application. For consistency all pages are processed by the Struts ActionServlet.
lib	.jar files placed here are available to the web application through its classloader. Jar files for the Struts framework and the Grouper API would go here

The Struts ActionServlet is configured in the web.xml file thus:

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

The <servlet-mapping> element causes any url ending in .do to be served by the Struts ActionServlet.

Note the **config** parameter. The ActionServlet initialization is driven by this file.

In order to use Tiles with Struts, Struts must be configured to load the Tiles plugin. In the struts-config.xml file:

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="moduleAware" value="true"/>
  <set-property property="definitions-debug" value="0"/>
  <set-property property="definitions-parser-details" value="0"/>
  <set-property property="definitions-parser-validate" value="false"/>
  <set-property property="definitions-config" value="/WEB-INF/tiles-def.xml"/>
</plug-in>
```

The Tiles plugin initialization is driven by **/WEB-INF/tiles-def.xml**.

All display text for the Grouper UI should be defined in a Java ResourceBundle. At its simplest a ResourceBundle may map to a single properties file e.g.

ResourceBundle bundle = ResourceBundle.getBundle("/resources/grouper/nav",locale);

could be satisfied by a single file, nav.properties, in the WEB-INF/classes/resources/grouper.

The file contents would be something like:

```
groups.my=My Groups
groups.manage=Manage Groups
groups.create=Create Groups
groups.join=Join Groups

groups.edit.name=Name
groups.edit.description=Description
groups.edit.type=Type
```

and text would be rendered in JSP pages by code such as:

```
<fmt:message bundle="${nav}" key="groups.my"/>
```

or

```
<html:submit property="submit.save" value="${navMap['stems.action.save']}/>
```

This assumes that nav and navMap have been made available as JSTL variables. The Grouper UI code is responsible for setting them up in each users' session.

In this case, no matter what the locale, each user would get the same text, the default Grouper UI text, however, as discussed in the next section it is relatively straightforward to provide a new language, or even a variation on the default language e.g., British English as opposed to US English.

3.2. Grouper UI Tailored for the University of Bristol (UoB)

GroupsManager	The web application document root. Note that we can name the web application as we see fit
---------------	--

grouper	Grouper specific stylesheets / JavaScript
images	Grouper specific images
i2mi	Internet 2 specific stylesheets / JavaScript
images	Internet 2 specific images
uob	UoB specific stylesheets / JavaScript
images	UoB specific images
WEB-INF	The Struts ActionServlet configuration must be modified.
classes	Additional resources / Java classes required by UoB copied here
jsp	Additional JSP files placed here - probably in a UoB subdirectory
lib	Any additional JAR files required by UoB code placed here

The Struts ActionServlet is configured in the web.xml file thus:

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config_uob.xml,/WEB-INF/struts-config.xml,/WEB-INF/struts-
config_uob.xml</param-value>
  </init-param>
  <init-param>
    <param-name>config/i2mi</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup> </servlet>
```

Note the `config/i2mi` parameter. This is only necessary IF you want to be able to run the unmodified Grouper UI at the same time as your modified version. This may be useful during development. In this instance i2mi represents a Struts module. We always use a module, however, we used the default module previously.

The `config` element now takes advantage of Struts' ability to load more than one config file. Ignore the first `/WEB-INF/struts-config_uob.xml` and focus on `/WEB-INF/struts-config.xml`, `/WEB-INF/struts-config_uob.xml`. Here we inherit all of Grouper's Struts configuration, loading any additional / replacement configuration we need. We might add some additional Struts actions, modify a form bean etc.

The `/WEB-INF/struts-config_uob.xml` file has a modified plug-in section:

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="moduleAware" value="true"/>
  <set-property property="definitions-debug" value="0"/>
  <set-property property="definitions-parser-details" value="0"/>
  <set-property property="definitions-parser-validate" value="false"/>
  <set-property property="definitions-config" value="/WEB-INF/tiles-def.xml,/WEB-INF/tiles-def_uob.xml"/>
</plug-in>
```

</plug-in>

Again, all the Grouper Tiles configuration is inherited, whilst additional / replacement tiles can be loaded.

The reason for the additional `/WEB-INF/struts-config_uob.xml` in the `config` parameter above is that although Struts can load multiple configuration files, the first instance of a particular plugin wins. Without the additional `/WEB-INF/struts-config_uob.xml`, the Tiles plugin configured for Grouper alone would be loaded.

It is possible that although the default language for Grouper is English, that UoB would like to reword / rebrand parts of the Grouper UI. We have already seen how text is stored in `/WEB-INF/classes/resources/grouper/nav.properties`. By adding a `/WEB-INF/classes/resources/uob/nav.properties` file we can configure the web application to first look at the UoB bundle. If it fails to find a particular key it will default to the Grouper bundle.*

In the same way that readable text is rendered on a page based upon looking up a key and obtaining a value, it is also possible to define urls for images and style sheets in a ResourceBundle e.g., `/WEB-INF/classes/resources/grouper/media.properties`: `image.organisation-logo=grouper/images/organisation-logo.jpg`

`image.grouper-logo=grouper/images/grouper.jpg`

`/WEB-INF/classes/resources/uob/media.properties` might include:

`image.organisation-logo=uob/images/banner.logo.gif`

`css.additional=uob/uob.css`

*see 3.4. for a discussion of the alternative approaches to overloading properties for locales and application components.

3.3. Supporting an Additional Language

Let's say that Bristol had a requirement to have a Spanish language version of the Grouper UI. The following list explains how this might be achieved:

1. Copy `/WEB-INF/classes/resources/uob/nav.properties` to `/WEB-INF/classes/resources/uob/nav_es.properties`, and replace the English text with Spanish text - leaving the keys alone.
2. Copy `/WEB-INF/classes/resources/uob/media.properties` to `/WEB-INF/classes/resources/uob/media_es.properties`, and replace the paths to any images which include English text to paths to new images which incorporate Spanish text - leaving the keys alone.
3. Create a file `/WEB-INF/tiles-def_uob_es.xml` and add any definitions required.
4. Provide the means to select a language*.

* If a user's browser is configured with a locale it might automatically be selected, however, it may be necessary to incorporate a selection means on the initial page of the site - this might be driven from a configuration option for the Grouper UI, e.g.,:

```
known.locales=en,es
```

Java [Locale](#) objects can, in principle, return an appropriate display name for a locale.

3.4. Overloading in ResourceBundles

3.4.1. How it Works

A ResourceBundle is created based on a specific Locale. A Java Locale can be made up from 3 components:

1. Language code
2. Country code
3. Variant

A Locale's string representation would be: `language_country_variant`. All the properties files associated with a `ResourceBundle` for a given Locale can be calculated by starting with the base name and adding _ followed by the Locale string, e.g.,:

`nav_en_GB` -> british english. If a key is not found here check `nav_en`. If a key is not found here check `nav`.

In our scenario `nav` provides the default Grouper UI text. A US university could add / replace key values by creating an `nav_en_US` properties file where as Bristol would go with `nav_en_GB`.

If Bristol also wanted to provide for a local dialect - 'west country' they could create a file `nav_en_GB_WestCountry.properties`.

At this point it isn't possible to extend the naming system. The `java.util.Locale` documentation discusses multiple variants, however, the actual code does not deal with them.

Now consider a release of the Grouper UI which is provided with a 'contributed' `nav_en_GB.properties` file. Bristol might still want to make changes to the contributed text and could do so by having a bristol variant e.g. `nav_en_GB_Bristol`, however, it would then not be possible to have a 'south west' variant.

3.4.2. Multiple UIs One Instance

Consider a scenario where an institution wants to offer different versions of the UI i.e., a student interface and a staff interface, or a read-only Portal interface where XML is output rather than XHTML.

Also consider a service provider of a Grouper UI wanting to offer a central but customizable service e.g. a regional consortium who would provide UI services to Bath University, University of the West of England, Bristol University, Exeter university and Plymouth University.

Each variation could be run as a separately configured web application. An alternative would be for one instance to support multiple views. In Struts terms this would mean a separate module for each variation.

In principle, each variation can be considered a Locale variant e.g. at Bristol we could use:

- `nav_en_GB`
- `nav_en_GB_student`
- `nav_en_GB_staff`
- `nav_en_GB_portal`

By choosing the correct Locale appropriate text can be presented.

In the consortium example:

- `nav_en_GB`
- `nav_en_GB_bu`
- `nav_en_GB_uwe`
- `nav_en_GB_uob`
- `nav_en_GB_exu`
- `nav_en_GB_plym`

However, there is a limit to the hierarchy. `nav_en_GB_uob_staff` is not possible.

3.4.3. Chaining ResourceBundles

An alternative is to chain `ResourceBundles` i.e., look for a resource in an institution first. e.g. `nav_uob` If not found, look for a resource in the supplied Grouper `ResourceBundle` e.g., `nav`. Several `ResourceBundles` could be chained if necessary.

Chaining may lead to other problems e.g., if the Grouper UI was supplied with a contributed `nav_es.properties`, Bristol could still overload this by having a `nav_uob_es.properties`, however, consider a key `grouper.audit-trail`:

- `nav` -> audit trail
- `nav_es` -> sendero auditoría

- nav_uob_en_GB->log
- nav_uob_es ->
{no key/value}

If the Locale was set to es, grouper.audit-trail would return log. Therefore, any keys overridden in nav_uob_en_GB would need to be overridden in nav_uob_es as well.

An alternative would be to copy nav_es to nav_uob_es and then override anything using nav_uob_es_ES (or other South American country codes!)

3.5. Overloading Struts Config Elements

When loading multiple Struts config files, the latter of two elements identified by the same name wins, however, in some cases it may be useful to extend rather than replace a config element. One example would be [ActionForms](#). The Grouper UI would provide an HTML form and a matching Struts ActionForm which would deal with known group types, however, institutions may define their own group types with attributes not known to the base Grouper UI code. An institution wishing to alter the ActionForm definition would, ideally, only specify the additional fields necessary rather than copy the definition and modify it.

This type of inheritance may be achieved by modifying the institutional Struts config file at build time. A system will be devised to make this automatic.

Struts Actions normally define a limited set of ActionForwards- destinations, one of which is chosen based on the Action logic. An institution may want to change the normal page flow by redefining a forward, or by providing extra logic to determine which ActionForward should be chosen.

Redefinition of an existing ActionForward could be achieved automatically at build time.

Providing extra logic could be achieved by various means e.g., redefine the Action class called to be a subclass of the Grouper UI supplied Action class, and call super.execute(...)

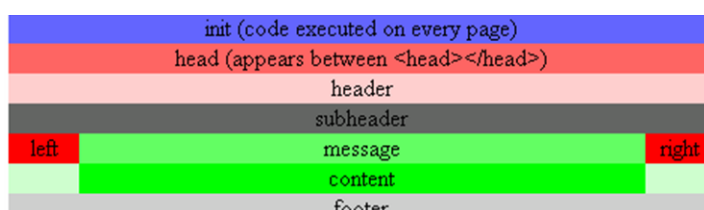
3.6. Additional Hooks to Extend the Grouper UI

Let's say that every time a group was created through the Grouper UI, Bristol wanted to immediately set up a shared mail folder for that group. This might be achieved by subclassing a Struts CreateGroupAction, however, a better way would probably be to register listeners for particular events.

We would need to decide if the UI code or the Grouper API is the appropriate place to register listeners for API calls which modify the underlying repository.

3.7. Page Composition Using Tiles

The actual page template for the Grouper UI has not yet been determined. The following discussion describes an approach to defining page layouts. Not all of the page areas defined in the following diagram may be used by the Grouper UI, however, individual institutions will be free to tailor the layout to meet their needs.



Such a page can be represented using a Tiles definition:

```
<definition name="BaseDef" path="/WEB-INF/jsp/template.jsp">
  <put name="header" type="definition" value="headerDef"/>
</definition>
```

```

<put name="footer" type="definition" value="footerDef"/>
<put name="subheader" type="definition" value="subheaderDef"/>
<put name="content" type="definition" value="/WEB-INF/jsp/empty.jsp"/>
<put name="left" type="definition" value="leftDef"/>
<put name="right" type="definition" value="rightDef"/>
<put name="head" type="definition" value="headDef"/>
<put name="message" type="definition" value="messageDef"/>
<put name="init" type="definition" value="initDef"/>
</definition>

```

Each xxxDef is a reference to another definition where, typically, the path will be defined as a specific JSP file.

Other page definitions can extend the BaseDef e.g.,:

```

<definition extends="BaseDef" name="EditGroupDef">
  <put name="content" type="page" value="/WEB-INF/jsp/EditGroup.jsp"/>
</definition>

```

This page has exactly the same layout as the BaseDef, however, the content attribute has been overridden. Any number of attributes could be overridden e.g.,:

```

<definition extends="BaseDef" name="EditGroupDef">
  <put name="content" type="page" value="/WEB-INF/jsp/EditGroup.jsp"/>
  <put name="left" type="page" value="/WEB-INF/jsp/EditGroupLeft.jsp"/>
</definition>

```

In the distributed Grouper UI it is likely that areas such as header and footer will be static, whilst subheader, left and right will be dynamically generated based on context. Each actual page displayed to a user will override content. Institutions are free to compose pages as they see fit.

BaseDef is implemented through /WEB-INF/jsp/template.jsp:

```

<%@include file="/WEB-INF/jsp/include.jsp"%>
<tiles:insert attribute="init" />
<html:html locale="true">
<head>
  <tiles:insert attribute="head"/>
</head>
<body>
  <div id="Header">
    <tiles:insert attribute="header" />
  </div>
  <div id="Navbar">
    <tiles:insert attribute='subheader'/>
  </div>
  <div id="Sidebar">
    <tiles:insert attribute="left" />
  </div>
  <div id="ContentSpace">
    <div id="TitleBox">
      <tiles:insert attribute="title" />
    </div>
    <c:if test="${!empty message}">
      <div id="Message">
        <tiles:insert attribute="message" />
      </div>
    </c:if>
  </div>

```



```
</c:if>
<\!--content-->
<div id="Content">
  <tiles:insert attribute='content' />
</div>
<\!--/content-->
</div>
<div id="Right">
  <tiles:insert attribute="right" />
</div>
<div id="Footer">
  <tiles:insert attribute="footer" />
</div>
</body>
</html:html>
```

The initial include holds common Java import statements and taglib references - needed for tiles and html tags, amongst others, to be handled correctly.

Each tiles:insert tag is replaced by the result of loading the relevant Tiles definition indicated by the attribute e.g. header is replaced by headerDef which is defined as having a path of /WEB-INF/jsp/header.jsp

It is entirely possible to have several different templates. It is also straightforward to override the definition of BaseDef such that a JSP file other than template.jsp acts as the common page template. This is a very powerful approach since changing a few files can completely alter how pages are composed.

Of course, if existing page elements are rearranged it is likely that CSS definitions will also need to be overridden.

3.8. Content Composition Using Tiles

The previous section dealt with the structural composition of a page from a coarse-grained perspective. It is also inevitable that institutions will want to customize what happens in the content area e.g. when browsing groups, rather than just display the displayExtension, other custom attributes may be shown.

The same approach as the last section can be applied, if a content area is composed of several tiles. Taking the Manage Groups page as an example:

MANAGE GROUPS - BROWSE GROUPS HIERARCHY
Groups I can manage

BROWSE GROUPS 0

GROUPS HOME- Bristol University 1

- Personal Groups 3 2
- Academic faculties
- Student Union
- Non academic departments
- Community groups
- [All staff at University of Bristol] 4
- [All students at University of Bristol]

SEARCH GROUPS 5

Search groups

MANAGE STEM 6

- Edit stem
- Delete stem
- Create stem
- Create group

The content is generated from ManageGroups.jsp. Box 0 is a tile (browseStems.jsp). The rest of the content is not yet broken down further, but boxes 1 through 6 show how further tiles could be used to assemble the content.

1. Bread crumb trail showing where you are in the hierarchy
2. The child stems and groups for the current location in the hierarchy (not actually used in the example below)
3. A tile which displays a stem
4. A tile which displays a group
5. A tile for displaying a Search groups form.
6. A tile for showing what actions are available for the current stem

All of these tiles could be re-used elsewhere. In addition they can be parameterized in exactly the same way as BaseDef in the previous section. e.g.,

```
<definition name="browseStemsDef"
  path="/WEB-INF/jsp/
browseStems.jsp"
  controllerUrl="/"
prepareStems.do">
  <put name="breadcrumb"
type="page" value="breadcrumb.jsp"/>
  <put name="childStem" type="page"
value="childStem.jsp"/>
  <put name="childGroup"
type="page" value="childGroup.jsp"/>
</definition>
```

Note the controllerUrl. This is a Struts action which is responsible for making the relevant data structures available to the tile.

The tile would be inserted into the content, thus:

```
<tiles:insert definition="browseStemsDef"/>
```

or

```
<tiles:insert definition="browseStemsDef"
  controllerUrl="/uobPrepareStems.do">
  <put name="childGroup"
type="page"
value="/uobChildGroup.jsp"
</tiles:insert>
```

In the former case all the defaults are used from the definition, whereas in the latter, controllerUrl and the childGroup definitions are overloaded.

3.9. Dynamic Tiles

Given that sites may define their own attributes for custom groups and for Subjects, as well as new Subject types, and given that depending on context it may be desirable to show a different *view* of an object, a flexible approach has been adopted for rendering objects. In essence, the Grouper UI can determine, at run time, the appropriate Tile to display based upon the type (and possibly subtype) of an object, and a named *view*. If a named view for an object has not been configured in an appropriate Java properties file, then a default view will be selected. The core Grouper UI will only configure a minimal set of default views for the object types and subtypes it knows about, however, institutions will be free to configure additional tiles to suit their needs.

A generic dynamic tile has been defined thus:

```
<definition controllerUrl="/getDynamicTileName.do" name="dynamicTileDef" path="/WEB-INF/jsp/dynamicTile.jsp"/>
```

The controllerUrl is responsible for determining the actual tile to load based on attributes assigned to the dynamic tile:


```
<tiles:insert definition="dynamicTileDef" flush="false">
  <tiles:put name="viewObject" beanName="subject"/>
  <tiles:put name="view" value="groupMember"/>
</tiles:insert>
```

In this example, a Subject is being rendered as a member of a group. The default view for all subject types is to display the *description* attribute of the Subject, however, it may be desirable to visually differentiate different Subject types and provide links appropriate to that Subject type. The dynamic tile approach provides this level of extensibility. Moreover, the implementation of the code which resolves an object and a view to a tile is pluggable i.e., can be extended or replaced completely, so that new object types / algorithms can be added to the Grouper UI.

3.10. JSP Tag Libraries

Where possible, the Java Standard Tag Library will be used in templates rather than scriptlets. Other open source tag libraries available from the [Apache Jakarta](#) site may be used. In addition, it is possible that we may write some custom tag libraries that work with Grouper objects.

It will be possible for institutions to configure and use additional tag libraries of their choice.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Bad Membership Finder Utility

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Bad Membership Finder Utility

This document is released alongside Grouper v1.4.0.

The Bad Membership Finder Utility goes through all of your list memberships, access privileges and naming privileges stored in the grouper_memberships table and finds any bad effective and composite memberships that have been incorrectly generated from immediate memberships.

This utility is divided into three phases.

Phase 1: The first phase will locate memberships that are associated with a group or stem that no longer exists.

Phase 2: The second phase will locate bad list memberships and access privileges for groups.

Phase 3: The third phase will locate bad naming privileges for stems.

Note that because this utility has to go through all of your groups and stems, it may take several hours to run if you have a large (1 million or more) number of memberships.

This utility is read-only; it does not make any membership changes in your database.

Usage

As of Grouper 1.4.0, this utility runs with GSH.

```
$GROUPER_HOME/bin/gsh.sh -findBadMemberships <command line arguments>
```

```
usage: FindBadMemberships -all | -group <arg> | -stem <arg>
-all          Find bad list memberships, access privileges, and naming
               privileges owned by all groups and stems.
-group <arg>   Find bad list memberships and access privileges for a
               specific group.
-stem <arg>    Find bad naming privileges for a specific stem.
```

This script will find bad effective and composite memberships in your Grouper database. It will not make any modifications to the Grouper database.

If bad memberships are found, this script will create a GSH script that will delete and re-add memberships.

To fix your memberships, complete these steps in the order listed:

1. Review the GSH script before applying any changes to your database.
2. Run the GSH script.
3. Re-run the bad membership finder utility for all groups and stems since additional membership errors for other groups and stems may be revealed after the current groups and stems are fixed.

For every bad membership, the utility will print one line.

If the membership does not belong to any group or stem, you will see a message like the following:

```
FOUND BAD MEMBERSHIP: Membership with uuid=5de0b45e-f1be-442d-8240-9aacc4b1054c has invalid owner
with uuid=5423131e-667c-4463-8e85-e5d16864f3ab.
```

Otherwise, you will see a message like the following for bad memberships with a group:

```
FOUND BAD MEMBERSHIP: Bad membership in group with uuid=c59c0b99-a735-4798-841a-a497d31afd0b and
name=i2:test.
```

And you will see a message like the following for bad memberships with a stem:

```
FOUND BAD MEMBERSHIP: Bad membership in stem with uuid=c59c0b99-a735-4798-841a-a497d31afd0b and name=i2.
```

Resolving bad memberships

If bad memberships are found in your Grouper database, a GSH script called `findbadmemberships.gsh` will be created to help you resolve the issues. For each group with a bad membership, the following will be added to the GSH script.

1. `delMember()` commands for each immediate member of the group.
2. `revokePriv()` commands for each access privilege.
3. A `delComposite()` command if this group is a composite group.
4. SQL statements to delete any remaining effective and composite memberships.
5. `addMember()` commands to re-add immediate members.
6. `grantPriv()` commands to re-add access privileges.
7. `addComposite()` to re-create the composite if this is a composite group.

For each stem with a bad membership, the following will be added to the GSH script.

1. `revokePriv()` commands for each naming privilege.
2. SQL statements to delete any remaining effective and composite memberships.
3. `grantPriv()` commands to re-add naming privileges.

For each membership with an invalid owner, the following will be added to the GSH script.

1. An SQL statement to simply delete the membership.

To fix your bad membership, do the following:

1. Review the GSH script before applying any changes to your database.
2. Run the GSH script.
3. Re-run the bad membership finder utility for all groups and stems since additional membership errors for other groups and stems may be revealed after the current groups and stems are fixed.

Note that you should try to avoid membership updates to the bad groups and stems between the time the GSH script is created and the time that you execute the script. If updates are made to the groups and stems between this time, you should recheck the groups and stems using `findBadMembershipsByGroup(group)` and `findBadMembershipsByStem(stem)`.

Also note that depending on the type of bad membership found, your execution of the GSH script may produce a constraint violation due to the foreign key `fk_memberships_parent`. If this happens, executing the GSH script a second time should succeed in most cases. But if it doesn't succeed, drop the foreign key, execute the GSH script, and then add the foreign key back.

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Contact Information

This page last changed on Sep 23, 2009 by steveo@internet2.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

This page has moved to <http://www.internet2.edu/grouper/about.html><http://www.internet2.edu/grouper/contact.html>.

Mailing List information is at <http://www.internet2.edu/grouper/lists.html>.

Please update your bookmarks and links.

Custom Group Types, Fields, Attributes, Lists

This page last changed on Feb 05, 2009 by [mchzyer](#).

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

This document will explain how to work with custom group types, and how you can best apply it to your environment.

Custom Group Types and Fields for Grouper v1.2.1

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). Deployers can augment the pool of available fields with their own **Custom Fields**. These are gathered into sets of custom fields to form a **Group Type**, and all defined group types are available to be assigned to individual groups by their ADMINS. Assignment of a group type to a group adds the associated set of fields to that group. These custom fields can then be managed or accessed by the API or the UI, appear in XML exports, etc.

In this section we describe 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, you'll need to know a bit more about Grouper fields.

Fields come in two flavors: **attributes** and **lists**. 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

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

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% subj=SubjectFinder.findById("GrouperSystem")
subject: id='GrouperSystem' type='application' source='g:isa' name='GrouperSystem'
gsh-0.0.1 1% sess=GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession: eb0c794b-09a1-4cc5-
b45b-4c2e4a6d3434,'GrouperSystem','application'
gsh-0.0.1 2% type=GroupTypeFinder.find("teaching")
edu.internet2.middleware.grouper.GroupType: teaching
gsh-0.0.1 3% type.delete(sess)
```

 Questions or comments?  [Contact](#) us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Customising the Grouper UI

This page last changed on May 28, 2008 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

How to Customise the Grouper User Interface (UI)

This document is current as of the Grouper v1.3.0 release.

- [Introduction](#)
- [CSS Changes](#)
- [Changing the Internet2 logo](#)
- [Changing the Default Text](#)
- [Using Custom Templates Instead of the Standard Templates](#)
- [Defining Custom Dynamic Templates](#)
- [Modifying Existing Struts Actions, Adding New Actions, and Making New Tiles Definitions Available](#)
- [Customising authentication](#)
- [Customising group authorizations](#)
- [Customising the Root Node of the Grouper Repository](#)
- [Creating an InitialStems View](#)
- [Customising Browsing and Searching](#)
- [Customising the Menu](#)
- [Personal Groups](#)
- [Displaying subjects, groups and stems](#)
- [Sort order of lists](#)
- [Enabling import / export of group memberships](#)
- [Customising the Build Process](#)
- [Customising web.xml](#)
- [Running the Standard UI at the Same Time as the Custom UI](#)
- [Determining How a Grouper UI Page Was Constructed](#)
- [Providing Feedback and Getting Help](#)

Introduction

This document is written for the web application developer. It describes how to make changes to the Grouper UI and is best understood with reference to the Grouper UI architecture (which contains links to underlying concepts and technologies). The section [Determining How a Grouper UI page Was Constructed](#) explains how to display on screen information, which can help determine what you need to change in order to modify a Grouper UI page. [Struts Actions](#) in the Grouper UI gives an overview of which Struts actions relate to which functional areas.

The document focuses on the specific customisations which can be built into the QuickStart demo (see the QuickStart README), and will refer to differences between the standard and custom screen shots in [Grouper UIs](#). You may want to open this link in a separate window / tab for reference.

The UI has been designed so that the source code for the standard UI need not be changed in order to effect customisations. This is intended to make it easier to upgrade changes to the standard UI without compromising customisations you have made.

The structure and contents of **grouper-qs/custom-grouper-ui** should be used as the starting point for your own custom Grouper UI.

CSS Changes

Grouper has its own CSS stylesheets, but provides a mechanism for site-specific stylesheets to be loaded after the Grouper stylesheets. This allows sites to override/extend existing styles and to add new styles. Such changes can alter the position of screen elements, fonts, colours, etc.

The custom stylesheet was configured in **grouper-qs/custom-grouper-ui/resources/custom/media.properties**:

css.additional=custom/custom.css

The actual stylesheet is found at **grouper-qs/custom-grouper-ui/webapp/custom/custom.css**.

Note: As of version 0.9 of Grouper `css.additional` can be a space separated list of stylesheets. It is also possible to 'turn off' the Grouper stylesheets by setting the key `grouper-css.hide=true`.

Differences in the standard and custom UIs which are due to CSS are listed below:

Feature	Standard UI	Custom UI
Menu	Positioned vertically on left.	Positioned horizontally below logos. Various colour changes.
Content area	To the right of the menu.	Aligned left and full width.
'Members'	Blue text on pale background.	White text on grey background.

Many more CSS classes are applied to elements in the HTML source than are specified in the Grouper stylesheets so a high degree of CSS customisation is possible.

It is possible to turn off the style sheets. This may be useful for testing the accessibility of the Grouper UI and any customisations you make (see the *Remove CSS stylesheet references?* form field in [Determining How a Grouper UI Page Was Constructed](#)).

Changing the Internet2 Logo

In **grouper-qs/custom-grouper-ui/resources/custom/media.properties**, the following property was set:

image.organisation-logo=custom/images/banner.logo.gif

Changing the Default Text

In the standard UI, all navigational text and instructions are derived from a Java ResourceBundle based on **grouper-qs/grouper-ui/resources/grouper/nav.properties**.

In the custom UI, values for keys set in **grouper-qs/custom-grouper-ui/resources/custom/nav.properties** will override the standard UI values. In the UI example screen shots, the custom UI includes *UoB* in menu items and changes *Help* to *Assistance*.

Through the use of Java ResourceBundles, the Grouper UI supports Internationalization. The default locale for the standard UI is set in **grouper-qs/grouper-ui/resources/init.properties**:

`default.locale=en_US`

In **grouper-qs/custom-grouper-ui/resources/init.properties**, **default.locale=en_GB**, however, there is no *nav_en_GB.properties* file so there are no differences due to locale in the standard and custom UIs.

Currently, there is nowhere in the UI to select a different locale from the default, however, if a *lang* parameter is passed as part of the URL which invokes login, the value will be used as the locale for the current session.

See [Determining How a Grouper UI Page Was Constructed](#) for details of how to display which key / value pairs were used in an actual page in the UI.

If you create your own templates you are not under any obligation to use ResourceBundles instead of directly entering text in templates, however, if you wish to contribute code back to Grouper, such a contribution would be more useful if it used ResourceBundles.

Using Custom Templates Instead of the Standard Templates

The Grouper UI uses Struts's Tiles to define core page components. In the standard UI these are defined in **grouper-qs/grouper-ui/webapp/WEB-INF/tiles-def.xml**. In the custom UI some definitions are modified and another added in the file **grouper-qs/custom-grouper-ui/webapp/WEB-INF/tiles-def-custom.xml**. New definitions for *headerDef* and *footerDef* allow site specific branding. *groupStuffDef* defines a template which is included in the Group Summary page of the UI.

EasyLoginFormDef defines a new page (see [Customising Authentication](#)).

Defining Custom Dynamic Templates

Grouper recognises some core entities such as Groups, Stems, Subjects and Collections. The Grouper UI dynamically chooses the appropriate template for an entity at runtime based on its type and the UI context. The default templates for an entity and view are defined in **grouper-qs/grouper-ui/resources/grouper/media.properties**. When a specific entity-view key is not found, a default is used. Key-value pairs can be overridden, or more specific keys added to **grouper-qs/custom-grouper-ui/resources/grouper/media.properties**. The algorithms used to choose appropriate keys are described in the Javadoc for [DefaultTemplateResolverImpl](#). This class can be extended to add support for other entity types, or a completely new implementation plugged in (see Javadoc for the [TemplateResolver](#) interface).

In the standard UI the default template for a subject is defined:

subject.view.default=/WEB-INF/jsp/subjectView.jsp

In the custom UI:

personQS.view.default=/WEB-INF/jsp/custom/customPersonSubjectView.jsp

defines a specific template for subjects whose source has an ID of personQS. In the UI examples the name Keith Benson is followed by (kebe) in the custom UI due to the use of **/WEB-INF/jsp/custom/customPersonSubjectView.jsp** as a template. Subjects and groups may have any number of site specific attributes, so dynamic templates allow sites to create templates which have access to these custom attributes.

See [Determining How a Grouper UI Page is Constructed](#) for determining which template was chosen for an entity-view on a page in the UI.

Modifying Existing Struts Actions, Adding new Actions, and Making New Tiles Definitions Available

The Grouper UI is based on Struts and the standard Struts configuration is through **grouper-qs/grouper-ui/webapp/WEB-INF/struts-config.xml**. Existing actions can be replaced and new ones added to **grouper-qs/custom-grouper-ui/webapp/WEB-INF/struts-config-custom.xml**.

See [Struts Actions in the Grouper UI](#) for an explanation of how the standard Grouper ui actions interact.

In the custom UI the action */callLogin* is redefined such that it forwards to */easyLogin* - a new action.

More information on how to change the behaviour of the Grouper Struts actions is available in the appropriate [javadoc package description](#).

Even if you don't need to change/add any actions, a Tiles plugin must be configured in order to make custom templates available (see [Using Custom Templates](#) instead of the standard templates):

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="moduleAware" value="true"/>
  <set-property property="definitions-debug" value="0"/>
  <set-property property="definitions-parser-details" value="0"/>
  <set-property property="definitions-parser-validate" value="false"/>
  <set-property property="definitions-config" value="/WEB-INF/tiles-def.xml,/WEB-INF/tiles-def-custom.xml"/>
</plug-in>
```

Note that the order of files in the definitions-config property is important as the last Tile definition with a particular name loaded is used.

Customising Authentication

The standard UI uses basic HTTP authentication configured through Tomcat and the web application web.xml file. A Filter [LoginCheckFilter](#) checks if you are logged in before allowing access to the application. It checks the `javax.servlet.http.HttpServletRequest.getRemoteUser()`. If not set the user is redirected to the splash page, otherwise, access is granted, and if necessary, the user session initialised.

In the custom UI, when a user clicks the *Login* link on the splash page, the */callLogin* action is requested. This forwards the user to the */easyLogin* action which displays the template named *EasyLoginFormDef*, which is a simple form allowing a username to be entered. The custom UI also defines a Filter *EasyLoginFilter* which is called prior to LoginCheckFilter. If it sees a request parameter called username, it attempts to load a Subject (through *SubjectFinder.findByIdentifier*). If successful, it stores the username in the HttpSession and calls the next Filter in sequence with a modified *HttpServletRequest*, which responds to *getRemoteUser()* by returning the stored username.

This section shows how new authentication schemes can be introduced. A more serious scheme that allows [Yale CAS Authentication](#) has been contributed to the Grouper UI distribution.

In order to configure new Filters, it is necessary to modify the web application web.xml file. How to do this is described in [Customising web.xml](#).

Note: The standard UI does not have a logout link, because it is not possible to safely logout of basic HTTP authentication. Other authentication schemes will generally work by setting an HttpSession attribute - which is cleared when an HttpSession is invalidated, so a logout link is provided.

Customising group authorizations

As of v1.3.0 it is possible to override how the UI decides what the current user can do with a group. The primary motivation for this feature is to allow some UI features to be turned off i.e. when the feature should be maintained by a loader. Customization is achieved by implementing the [UIGroupPrivilegeResolver](#) interface, and configuring it through media.properties e.g.

```
edu.internet2.middleware.grouper.ui.UIGroupPrivilegeResolver=uk.ac.bris.is.grouper.ui.UoBUIGroupPrivilegeResolver
```

Customising the Root Node of the Grouper Repository

*see also [Customizing Browsing and Searching](#)

The Grouper API defines a root stem. In the standard UI the **Current location** begins with *Root* whereas in the custom UI it starts with *QS University of Bristol*. Both screen shots are views of the same Grouper repository. Three scenarios are possible

1. Root is visible, and browsing starts at Root,
2. Root is visible but browsing starts at a defined stem e.g. *QS University of Bristol*.
3. Root is not visible and browsing starts at a defined stem e.g. *QS University of Bristol*.

As of Grouper 0.9 it is possible to specify a root node per browse mode (see [AbstractRepositoryBrowser](#)). If none is specified, **default.browse.stem** from media.properties is used. If this is not set the the root node is used and scenario 1 occurs. If the root node is displayed, its name is determined by **stem.root.display-name** from nav.properties. If this is not set 'Root' is used by default.

By default, the hide-pre-root-node value for each *RepositoryBrowser* defined in media.properties, is set to *true*. This leads to scenario 3.

If hide-pre-root-node is set to false then scenario 2 occurs.

Creating an *InitialStems* View

*see also [Customizing Browsing and Searching](#)

This feature is not implemented in either the standard or custom UIs; however, it provides an alternative starting point for browsing, by allowing sites to provide a customised list of stems or *quick links*. The list of stems can come from any part of the hierarchy, and so may provide a better starting point for users, i.e., for GrouperSystem the default view is:

- [Personal Groups](#)
- [Academic Faculties](#)
- [Student Union](#)
- [Non-Academic Departments](#)
- [Community Groups](#)
- [\[All Students\]](#)
- [\[All Academic Staff\]](#)
- [\[All Students and Academic Staff\]](#)
- [\[UoB Administrators\]](#)

But for another user (e.g., an art student), the following list might be more appropriate:

- [Personal Groups for <name>](#)
- [Arts Faculty](#)
- [Student Union Clubs](#)

Such a list could be generated in a site-specific way based on a username. A site might also provide a means for a user to edit their list of quick links.

See Javadoc for [InitialStems](#) interface.

As of Grouper 0.9 it is possible to define a different InitialStems implementation for each browse mode; see [AbstractRepositoryBrowser.getInitialStems\(\)](#)

Customising Browsing and Searching

As of version 0.9 of Grouper it is possible to customise existing browse modes and add new browse modes. It is also possible to specify *root nodes* and *InitialStem* implementations on a mode by mode basis.

At runtime [RepositoryBrowserFactory](#) is used to obtain a [RepositoryBrowser](#) implementation for the current browse mode, by obtaining the class name of the implentation from **resources/grouper/media.properties** using the key **repository.browse.<mode>.class**. All Grouper supplied implementations extend [AbstractRepositoryBrowser](#), which reads further properties. Thus, the behaviour of supplied browse modes can be modified by changing relevant properties. The logic can be modified by providing a new implementation class - possibly a subclass of the Grouper implementation.

Alternatively, new browse modes can be implemented and configured. In general you will also need to implement a top level Struts action and page for any new browse mode, and provide links as appropriate. See [Customising the Menu](#) for details on how to change the default menu.

Customising the Menu

As of version 0.9 of Grouper the menu is configurable. [PrepareMenuAction](#) reads **resources/grouper/menu-items.xml** to obtain a list of known menu items. A **media.properties** key, **menu.order**, determines the order in which items are rendered.

Sites can add additional menu items by creating their own menu-items.xml and adding the file name to the **media.properties** key: **menu.resource.files**.

If sites want to have different menus for different users they can subclass **PrepareMenuAction** and override the **protected boolean isValidMenuItem(Map item,GrouperSession grouperSession,HttpServletRequest request)** method. You will also need to override the Struts action **prepareMenu.do**.

As of version 1.2.1 a new [MenuFilter](#) interface has been introduced to allow a more structured approach to customizing menus. Two implementations are provided, configured as:

```
menu.filters=edu.internet2.middleware.grouper.ui.RootMenuFilter
edu.internet2.middleware.grouper.ui.GroupMembershipMenuFilter
```

[GroupMembershipMenuFilter](#) is configured using a [UiPermissions](#) object and allows menu items to be vetoed depending on whether or not a user is a member of a group.

Personal Groups

Grouper has no specific support for personal groups, however, by implementing the [PersonalStem](#) interface, the Grouper UI will create a 'personal stem' for a user (if one does not exist) at login. An implementation of *PersonalStem* is provided at **grouper-qs/custom-grouper-ui/java/src/edu/internet2/middleware/grouper/customqs/ui/CustomQSPresonalStem.java**. This implementation creates a stem (extension=subject id) as a child of **/qsuob/personal**. Currently any user who is logged in can see personal stems. Whether they can see groups in the personal stem will depend upon Access privileges. Sites could use custom implementations of RepositoryBrowsers to implement their own business rules around personal stems and groups.

Displaying subjects, groups, and stems

Prior to Grouper v1.2.0 it was necessary to use custom dynamic tiles to change how subjects, groups and stems are displayed. This still remains the most flexible approach, especially if you need to show more than one attribute.

As of Grouper v1.2.0 it is possible to configure a single arbitrary attribute to use when displaying a subject, group or stem. The default media.properties keys are:

```
#Default if an attribute is not configured for a specific subject source. 'description' is set for backwards
compatability
subject.display.default=description
```

```
#used for subjects which are groups sourced by Grouper
subject.display.g\:gsa=displayExtension
```

```
#used for internal subjects i.e. GrouperSystem and GrouperAll
subject.display.g\:isa=name
```

```
#default attribute for groups (when not viewed as a subject)
group.display=displayExtension
```

```
#flat = context i.e. flat mode in the UI. Here the hierarchy is not shown and names displayExtension need not
be unique across
#multiple stems.
group.display.flat=displayName
```

```
#default attribute for stems
```

```
stem.display=displayExtension
```

In the QuickStart the following key is also used:

```
subject.display.qsuob=name
```

When displaying search results sites can configure a default attribute to display:

```
search.group.result-field=name  
search.stem.result-field=name
```

in addition sites can configure a set of attributes from which the user may select one to display:

```
search.group.result-field-choice=name displayExtension displayName  
search.stem.result-field-choice=name displayExtension displayName
```

As of Grouper v1.2.1 it is possible, for the SubjectSummary page, to specify a subset of available attributes to display and the order in which to display them:

```
# subject.attributes.order.<SOURCE_ID>=comma separated list of case sensitive attribute names  
subject.attributes.order.g  
\:gsa=displayExtension,displayName,name,extension,createTime,createSubjectId,createSubjectType,modifySubjectId,modifySub  
  
#subject.attributes.order.qsuob=LOGINID,LFNAME,subjectType,id
```

The UI wraps API objects as specific subclasses of [ObjectAsMap](#). As of v1.3.0 it is possible to configure your own implementations. Typically these would subclass the Grouper UI concrete subclasses and add/modify behaviour. Configuration is through media.properties e.g.

```
objectasmap.StemAsMap.impl=uk.ac.bris.is.grouper.ui.util.UOBStemAsMap
```

You could use this feature to create virtual attributes as composites of other attributes.

Sort order of lists

As of Grouper v1.2.0 various lists may be sorted alphabetically.

- search results for:
 - groups
 - stems
 - subjects
- group memberships
- group privilegees
- stem privilegees
- groups / stems where a subject has a selected privilege
- saved subjects
- saved groups
- stems / groups whilst browsing
- stems / groups in *flat* mode

See Javadoc for [LowLevelGrouperCapableAction.html.sort](#) and [DefaultComparatorImpl](#) for detailed information.

In principle, the sort algorithm should use exactly what is displayed on screen to sort lists, and, by default, this is what it does. The sort algorithm, therefore, takes account of the configuration for [Displaying subjects, groups and stems](#). On the other hand, Grouper allows the use of dynamic tiles and so it is possible to override the defaults in a way that the sort algorithm cannot work out. If a site does use dynamic tiles to display subjects, groups or stems, it is possible to configure Grouper to use alternate configuration for sorting, but it is the responsibility of the administrator to ensure that the sort configuration is appropriate for what is displayed on screen. For maximum flexibility, it is also possible to configure different attribute(s) to sort on for different contexts*. The 'search' order for keys is documented for each implementation of [GrouperComparatorHelper](#).

*The different contexts recognised are:

- search

- members
- flat
- subjectSummary
- privilegees

As of Grouper v1.2.1 you can sort subjects from the same source together by defining strings which are pre-pended to the usual sort string:

```
subject.pre-sort.g\gsa=AAA
subject.pre-sort.qsuob=BBB
```

Enabling import / export of group memberships

As of Grouper v1.2.0 it is possible to configure the UI to enable import / export of group memberships. Simple implementation classes are provided for dealing with tab or comma delimited files. In general, the formats for import or export vary for different sites. By default import / export is not enabled. Import is controlled by [MembershipImportManager](#) and a [DefaultMembershipImporter](#) class is provided. Export is controlled by [MembershipExporter](#).

In the QuickStart import and export is 'activated' using:

```
membership-export.config=resources/custom/membership-export.xml
membership-import.config=resources/custom/membership-import.xml
```

You can adapt these configuration files for your own needs and even write your own import implementation if the classes provided are unsuitable.

As of Grouper v1.2.1 you can configure the UI to allow import of data from a text area:

```
membership-import.allow-textarea=true
```

If a user does not select a file to import, the user is presented with a text area where they can type or paste data.

Customising the Build Process

The Grouper UI uses the **grouper-qs/grouper-ui/build.xml** ant script to build the web application. This script is configured through **grouper-qs/grouper-ui/build.properties**, which has a key **additional.build**. In the custom UI this is set to `\${basedir}/../custom-grouper-ui/additional-build.xml`. It is the responsibility of this script, which is called by the standard script, to compile any Java source files and to copy to the build area any other necessary files. If you wish to incorporate any contributed code, calls to the relevant build scripts should be placed here. In the **custom-grouper-ui/additional-build.xml** script, the struts-patch build script is called.

Customising web.xml

A web application web.xml file is a key configuration file and any site wishing to customise the Grouper UI will need to modify it. The web.xml is a J2EE deployment descriptor which configures the Servlets (how URLs are mapped to Java classes), the filters (pre/post logic around servlets), j2ee security (if not done in apache or somewhere else), listeners (for j2ee events), custom tag libraries (how some tags in JSPs map to java classes), etc. Things you might need to customize are filters (e.g. a new way to do authentication / authorization), security (do you want the servlet container to manage authentication / authorization?), custom tag libraries (are you using a new library in JSP extensions?), etc.

The default deployment descriptor is found at **grouper-qs/grouper-ui/webapp/WEB-INF/web.core.xml**. The UI provides a mechanism for merging fragments of different web.xml files into a final deployment file. In your additional build script (see [Customising the Build Process](#) copy any web.xml fragments into `\${temp.dir}`. Typically files should be prefixed with a 2 digit number e.g. 20 (90 is used for web.core.xml). The merging process merges in name order of the files. The custom UI includes two web.xml fragments which, when copied, are prefixed with 00 and 95 so the former is merged with web.core.xml and the latter is merged with the result of the first merge.

The first web.xml fragment is **grouper-qs/custom-grouper-ui/webapp/WEB-INF/web.custom.xml** and it overrides the default action servlet definition to ensure that it loads the Struts config customisations - which in turn load the Tiles definition customisations.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config-custom.xml,/WEB-INF/struts-config.xml,/WEB-INF/struts-config-
custom.xml</param-value>
  </init-param>
  <init-param>
    <param-name>config/i2mi</param-name>
    <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
```

Note: As the order of elements in the final web.xml file is important it can be difficult to get elements in the order you want. The merging process is not extensively tested and it is quite likely it will not work properly for all elements. It may be necessary to rework the merging process, or resort to manual editing of the web.core.xml file.

The merge process is dependent on **web-xml-merge.xsl** and **web-xml-merge-tags.xml** found in **grouper-qs/grouper-ui**.

Running the standard UI at the same time as the custom UI

By applying the [struts-patch](#) contribution (see [Customising the Build Process](#)), and configuring the Struts action servlet with more than one module see (**config/i2mi** parameter in [Customising web.xml](#)), it is possible to have both the standard and custom UIs available at the same time.

After building the custom Grouper UI you appear to *lose* the standard UI, however, if instead of accessing */grouper*, you access */grouper/i2mi* in your web browser, you then get the standard UI.

Determining How a Grouper UI Page Was Constructed

Since a Grouper UI page may be constructed from many templates, including dynamic templates, and it may not be easy to determine which ResourceBundle key was used to render text, a mechanism has been created to display *debug* information. Go to the URI */grouper/populateDebugPrefs.do*. You should see a form:

The form fields are explained in the table below:

Field	Description
Enable debug display	Determines if debug information is shown at the bottom of the page. If not selected, none of the other options are active.
Webapp root for I2mi*	The complete file system path to the standard UI webapp root.

Webapp root for your site*	The complete file system path to the custom UI webapp root.
Show resource keys and values at end of page	If selected, any key-value pairs derived from the nav ResourceBundle to display screen text are listed.
Show resource keys in page rather than values	Instead of seeing the text in the page you will see <i>?key?</i>
Show dynamic tiles	If selected, a hierarchy of templates used to construct the page is shown. If a template was loaded dynamically, the <i>view</i> , <i>entity type</i> , <i>chosen key</i> and <i>template name</i> are displayed.
Executable for JSP editor	

The complete filesystem path to a JSP editor - only use if the server is your working machine!
If the webapp roots above are specified, template names are links which will open the template file in the specified JSP editor.

Remove CSS stylesheet references?	Allows you see the non stylised page - used for checking accessibility in absence of a screen reader.
-----------------------------------	---

*These fields are only required if you wish to link template names to a JSP editor.

Providing Feedback and Getting Help

The Grouper UI is intended to be extensible and not to force unnecessary constraints, however, it is only as sites try to make their own customisations that the true extensibility can be tested. If while customising the Grouper UI you find yourself forced to modify standard Grouper UI sources (of any kind), or find that you cannot easily do what you want to, please offer feedback to, or request help via the grouper-users [mailing list](#).

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper UIs

This page last changed on May 28, 2008 by tbarton@uchicago.edu.



Welcome Fiona

My tools

Explore

[Search](#)

[Group workspace](#)

[Entity workspace](#)

[Help](#)

EXPLORE

Members

Current location is:

 Root:  QS University of Bristol:  **UoB Administrators**

Membership list

- ☒ Show DIRECT members of this group
- ☐ Show INDIRECT members of this group
- ☐ Show ALL members of this group (direct and indirect)

Change display

Showing 1-1 of 1 items

Click an entity name to view entity details, or click a membership description to view details

- ☐ Keith Benson is a direct member

Remove selected members

Remove all members

[Add members](#)

[Create composite group](#)

[Back to group summary](#)

Grouper is sponsored by



[ALL UOB GROUPS](#)[Members](#) ⓘ**Current location is:** [QS University of Bristol](#):  **UoB Administrators****MEMBERSHIP LIST**

- ☒ Show [DIRECT members](#) of this group
- ☐ Show [INDIRECT members](#) of this group
- ☐ Show ALL members of this group (direct and indirect)

[Change display](#)**EXPORT MEMBERS**

Tab separated (on screen) ▼

[Export members](#)**IMPORT MEMBERS**[Browse...](#)

Tab separated ▼

[Import members](#)

Showing 1-1 of 1 items

Click an entity name to view entity details, or click a membership description to view/modify privileges.

- ☐ [Keith Benson \(kebe\)](#) is a direct member

[Remove selected members](#)[Remove all members](#)[Add members](#) [Create composite group](#) [Back to group summary](#)

Developer's Guide to the Grouper API



This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Using the Grouper API to bootstrap your Groups Registry

This document is current as of the v1.2.0 release.

- [Find *GrouperSystem* Subject](#) ([example code](#))
- [Start-and-stop sessions](#) ([example code](#))
- [Find the root stem](#) ([example code](#))
- [Find stem by name](#)([example code](#))
- [Create stem](#) ([example code](#))
- [Find group by name](#) ([example code](#))
- [Create group](#) ([example code](#))
- [Find *GrouperAll* subject](#) ([example code](#))
- [Check for membership in the wheel group](#) ([example code](#))
- [Add wheel group member](#) ([example code](#))

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Glossary

This page last changed on May 16, 2008 by bmk@isc.upenn.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Glossary

Terms with Grouper-specific meaning are defined below, along with other Grouper concepts. An understanding of these terms will enable you to take full advantage of all that Grouper has to offer.

As of v1.3.0, [terminology used in the Grouper UI](#) differs from some of the terms defined below to help the UI to present group management tasks in a manner more readily understandable by non-technical users. The terminology used in developer and system administrator oriented documentation remains unchanged.

Note: To view an HTML version of this document, open the [attachment](#) above.

TERM	DEFINITION	UI Translation (where applicable)
<i>Access Privileges</i>	<p>Privileges that determine what a Subject can do with a Group. They are:</p> <ul style="list-style-type: none">• ADMIN - can assign access privileges and manage all group information,• UPDATE - can manage membership of the group (implies READ),• READ - can see the membership of the group (implies VIEW), and• VIEW - can see the group. In addition, a group may have options for its members to:• OPTIN - can add self to the membership, and• OPTOUT - can remove self from membership.	<p>Subject is a UI "entity"</p>
<i>Attribute</i>	<p>A single-valued string associated with a Group or a Naming Stem. By default, Grouper supports six attributes (one of two kinds of Field):</p> <ul style="list-style-type: none">• id - a Grouper-assigned, globally unique identifier.• extension- the relative name of the group or naming stem within its parent naming stem; the contribution of a single element, such as a group or a naming stem, to the cumulative name.	<ul style="list-style-type: none">• id is the UI "UUID"• extension is the UI "ID"• name is the UI "ID path"• displayExtension is the UI "name"• displayName is the UI "path"

	<ul style="list-style-type: none"> • name - used to facilitate searching for groups by name, it is a read-only string representation of the logical ordered pair of (<i>parent stem</i>, <i>extension</i>). This attribute is system-maintained. The string representation of the <i>name</i> attribute is: <i><parent stem>: <extension></i>. • displayExtension - a displayed form of the extension. • displayName- used to facilitate searching for groups by the displayed name, it is a read-only string representation of the logical ordered pair of (<i>displayName of parent stem</i>, <i>displayExtension</i>). This attribute is system-maintained. The string representation of the <i>displayName</i> attribute is: <i><displayName of parent stem>: <displayExtension></i>. • description - a description of the group or naming stem. 	
Composite Group	<p>A Group whose Membership is determined by combining the membership lists of two other groups, without listing its members explicitly. These two groups are called its Factor Groups. Three methods of combining the factor groups' memberships are supported:</p> <ul style="list-style-type: none"> • union - all subjects must be a member of one OR the other factor group, e.g., Group Z = members of either Group X OR Group Y, or $Z = X \cup Y$. • intersection - all subjects that are members of the first factor group AND the second factor group, e.g., Group Z = members of both Group X AND Group Y, or $Z = X \cap Y$. • relative complement - all members of the first factor group that are NOT members of the second factor group. e.g., Group Z = members of Group X AND NOT Group Y, or $Z = X - Y$. 	

Direct Membership	A Subject that is listed in the Membership list of a Group has a direct membership in the group. Also see Indirect Membership .	Subject is a UI "entity"
Factor Group	A Group in combination (union , intersection , or relative complement) with that of another factor group, which defines the membership of a resulting Composite Group .	
Field	Either an Attribute or a List . Grouper groups are a collection of attributes and lists, i.e., a collection of fields. The set of fields attached to a given group is a function of the set of Group Types it has been assigned.	
Group	<p>A list of Subjects having Membership in the group, together with other attributes about the group. A list can have zero or more entries. In Grouper, a list contains only subject references, and an attribute is a single-valued string. A group must be created in an existing Naming Stem. If a group is made a member, i.e., a Subgroup, of another group, the members of the group will also be made members. By default, a Grouper group has:</p> <ul style="list-style-type: none"> • six naming Attributes, • a description attribute, and • a members list. <p>This information model can be extended to include additional site-defined attributes and lists.</p>	naming stem is a UI "folder"
Group Math	Any combination of groups for the purpose of creating another group based on the memberships of those groups. See Composite Group .	
Indirect Membership	A Subject that is a member of a Subgroup of a Group , or a member of a Factor Group that contributes positively to a group's membership, has an indirect membership in the group. Also see Direct Membership .	
List	A multi-valued list of Subject references, (one of two kinds	

	<p>of Field). The direct members of a group are the values of the group's members list. Lists are also used to identify which subjects have which Naming or Access Privileges. Sites can extend a group type to include custom lists; however, their semantics are external to Grouper. See Group.</p>	
Member	<p>Any Subject in the membership list of at least one group. Also, a Member of a Group is any Subject with a Direct or Indirect Membership in the Group.</p>	
Membership	<p>The direct-only, indirect-only, or direct plus indirect members of a Group. A specific variety of membership is determined by context or configuration, i.e., the default User Interface allows the user to select among these three types of membership where appropriate.</p>	
Naming Privileges	<p>These privileges determine what a Subject can do with a Naming Stem. They are:</p> <ul style="list-style-type: none"> • CREATE - can create a group(s) named with a naming stem, and • STEM - can assign who has CREATE for the naming stem, and can create naming stems subordinate to this one. 	<p>Naming privileges are now referred to as Creation privileges and the two types are Create Group (replaces CREATE) and Create Folder (replaces STEM)</p>
Naming Stem	<p>A string that forms the leading part of a Group's name. By linking the ability to create groups to a specified naming stem (via the CREATE privilege), the possibility that different groups can be given the same name is substantially reduced, and the name of each group can be made to reflect something about the authority under which it was created. ...see Examples below.</p>	<p>Stem is a UI "folder"</p>
Stem	<p>A synonym for a Naming Stem.</p>	<p>Stem is a UI "folder"</p>
Subgroup	<p>A Group that is a Direct Member of another group.</p>	
Subject	<p>An abstraction of any object whose Memberships are to be managed by Grouper. Most Grouper deployments will manage subjects that</p>	<p>Subject is a UI "Entity"</p>

	represent people and groups, but computers, accounts, services, or any other type of object maintained in a back-end identity store may be presented as subjects to Grouper by use of the [Subject API] .	
Type	<p>There are two distinct uses for this term in Grouper.</p> <ul style="list-style-type: none"> • Group Type - each Group has one or more group types associated with it. The Grouper distribution contains support for a single group type called "base", but sites may register additional types, together with the attributes and lists associated with them, within their Grouper installation. Doing so enables management of groups with a richer information model or a more diverse set of information models. • Subject Type - the [Subject API v0.2.1] that Grouper 1.0 relies on uses the notion of a subject type, such as "person", "group", or "computer", etc. 	

Examples

Step 1: Create a Root Naming Stem

In the example below, a root naming stem is first created. Note: creating a naming stem is required prior to the creation of any groups.

Naming Stem uofc

<i>attribute</i>	<i>value</i>
<i>naming stem</i>	empty
<i>extension</i>	uofc
<i>displayExtension</i>	The University Of Chicago
<i>name</i>	uofc
<i>displayName</i>	The University Of Chicago

Step 2: Create a Group

Next, a group may be created using the "uofc" naming stem.

Group uofc:exec_council

<i>attribute</i>	<i>value</i>
<i>naming stem</i>	uofc
<i>extension</i>	exec_council
<i>displayExtension</i>	Executive Council
<i>name</i>	uofc:exec_council
<i>displayName</i>	The University of Chicago:Executive Council

Step 3: Create a Subordinate Naming Stem and Group

Name and displayName values propagate down through subordinate naming stems, e.g the Biological Sciences Division within U of C:

Naming Stem uofc:bsd

<i>attribute</i>	<i>value</i>
<i>naming stem</i>	uofc
<i>extension</i>	bsd
<i>displayExtension</i>	Biological Sciences Division
<i>name</i>	uofc:bsd
<i>displayName</i>	The University Of Chicago:Biological Sciences Division

Again, a group is created, e.g., the Enterprise Information Systems staff, with the above naming stem, and is displayed as follows:

Group uofc:bsd:eis_staff

<i>attribute</i>	<i>value</i>
<i>naming stem</i>	uofc:bsd
<i>extension</i>	eis_staff
<i>displayExtension</i>	Enterprise Information Systems staff
<i>name</i>	uofc:bsd:eis_staff
<i>displayName</i>	The University Of Chicago:Biological Sciences Division:Enterprise Information Systems staff

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

UI Terminology

This page last changed on May 16, 2008 by bmk@isc.upenn.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Terminology in the Grouper User Interface as of v1.3.0

The below table breaks terminology into categories and shows the terms used prior to v 1.3, the terms in the production version v1.3.0 and a description of the term

Category	Old Term	New Term	Definition/ Description
UI Labels	Privilegees	Entities With Privileges	
	Subject	Entity	An entity is an abstract item which may be a member of a group. The two most common types of entities are 'person' or 'group'. (In the future, additional entity types may be used to describe computers or applications.)
	is a direct privilegee	has direct privileges	as a member of the group
	is a indirect privilegee	has indirect privileges	within a group that is a member of the group
	Extension	ID	An internal name describing this group that is generally not exposed to the user. This name cannot be changed after it is edited
	Name	ID Path	An internal concatenation of the hierarchy to this group that is generally not exposed to the user
	Display extension	Name	The group name that is displayed when browsing or searching
	Display name	Path	The path is the concatenation of the hierarchy (folders and groups) that lead to the unique location of this group
Hierarchy	stem [conceptual]	Folder	

			a fundamental unit (container) of the hierarchy that can have a parent (folder or 'root') or children (folders or groups)
	group	group	a type of entity made up of members
	Manage Stem	Manage Folder	This is where you can create or edit the folders within the hierarchy or add groups to the hierarchy
Hierarchy Priv	stem [privilege]	Create Folder	the ability to create children folders or branches in the hierarchy
	Create	Create Group	Add or create the name for a new group at this folder (location) in the hierarchy however the entity that creates a group is given Admin rights to the group by default. This does not provide access to manage the group (add membership or edit attributes)
	Stem privilege	Creation Privileges	a hierarchy Is made up of folders. The folder subfolder relationship define the path through the hierarchy
Navigation	saved subjects	Entity Workspace	a session specific area where you can store groups that you will need to create compound groups, etc
	Saved groups	Group Workspace	a session specific area where you can store groups that you will need to create compound groups, etc
	Search subjects	Search	
Administrative	grouperAll	EveryEntity	Default group privileges that are inherited upon group creation
	GrouperSystem	GrouperSysAdmin	the highest level administrative user of the system

	WheelGroup	SysadminGroup	all people in this group have full system admin privileges
Group Priv	Admin	Admin	Entity (typically group or person) may modify the membership of this group, delete the group or assign privileges for the group
	Member	Member	Any entity (typically group or person) that is a part of this group
	Optin	Optin	Entity (typically group or person) may choose to join this group
	Optout	Optout	Entity (typically group or person) may choose to leave this group
	Read	Read	Entity (typically group or person) may see the membership list for this group
	Update	Update	Entity (typically group or person) may modify the membership of this group
	View	View	Entity (typically group or person) may see that this group exists

Below are Grouper concepts described/translated using the UI terminology of version v1.3.0

TERM	DEFINITION
Access Privileges	<p>Privileges that determine what a Entity can do with a Group. They are:</p> <ul style="list-style-type: none"> • ADMIN - can assign access privileges and manage all group information, • UPDATE - can manage membership of the group (implies READ), • READ - can see the membership of the group (implies VIEW), and • VIEW - can see the group. <p>In addition, a group may have options for its members to:</p> <ul style="list-style-type: none"> • OPTIN - can add self to the membership, and • OPTOUT - can remove self from membership.
Attribute	<p>A single-valued string associated with a Group or a Folder. By default, Grouper supports six attributes (one of two kinds of Field):</p>

	<ul style="list-style-type: none"> • UUID - a Grouper-assigned, globally unique identifier. • ID- the relative name of the group or folder within its parent folder; the contribution of a single element, such as a group or a folder, to the cumulative name. • ID Path- used to facilitate searching for groups by name, it is a read-only string representation of the logical ordered pair of (<i>parent folder, ID</i>). This attribute is system-maintained. The string representation of the ID Path attribute is: <i><folder>: <ID></i>. • Name- a displayed form of the ID. • Path -used to facilitate searching for groups by the path, it is a read-only string representation of the logical ordered pair of (<i>Path of parent folder, Name</i>). This attribute is system-maintained. The string representation of the path attribute is: <i><Path of parent folder>: <name></i>. • description - a description of the group or folder.
Composite Group	<p>A Group whose Membership is determined by combining the membership lists of two other groups, without listing its members explicitly. These two groups are called its Factor Groups. Three methods of combining the factor groups' memberships are supported:</p> <ul style="list-style-type: none"> • union - all entities must be a member of one OR the other factor group, e.g., Group Z = members of either Group X OR Group Y, or $Z = X \cup Y$. • intersection - all entities that are members of the first factor group AND the second factor group, e.g., Group Z = members of both Group X AND Group Y, or $Z = X \cap Y$. • relative complement - all members of the first factor group that are NOT members of the second factor group. e.g., Group Z = members of Group X AND NOT Group Y, or $Z = X - Y$.
Direct Membership	<p>An entity that is listed in the Membership list of a Group has a direct membership in the group. Also see Indirect Membership.</p>
Factor Group	<p>A Group in combination (union, intersection, or relative complement) with that of another factor group, which defines the membership of a resulting Composite Group.</p>
Field	<p>Either an Attribute or a List. Grouper groups are a collection of attributes and lists, i.e., a collection of fields. The set of fields attached to a given group is a function of the set of Group Types it has been assigned.</p>
Group	<p>A list of Subjects having Membership in the group, together with other attributes about the group. A list can have zero or more entries. In Grouper, a list contains only entity references, and an</p>

	<p>attribute is a single-valued string. A group must be created in an existing Folder. If a group is made a member, i.e., a Subgroup, of another group, the members of the group will also be made members. By default, a Grouper group has:</p> <ul style="list-style-type: none"> • six naming Attributes, • a description attribute, and • a members list. <p>This information model can be extended to include additional site-defined attributes and lists.</p>
Group Math	Any combination of groups for the purpose of creating another group based on the memberships of those groups. See Composite Group .
Indirect Membership	An Entity that is a member of a Subgroup of a Group , or a member of a Factor Group that contributes positively to a group's membership, has an indirect membership in the group. Also see Direct Membership .
List	A multi-valued list of Entity references, (one of two kinds of Field). The direct members of a group are the values of the group's members list. Lists are also used to identify which entities have which Creation or Access Privileges . Sites can extend a group type to include custom lists; however, their semantics are external to Grouper. See Group .
Member	Any Entity in the membership list of at least one group. Also, a Member of a Group is any Entity with a Direct or Indirect Membership in the Group .
Membership	The direct-only, indirect-only, or direct plus indirect members of a Group . A specific variety of membership is determined by context or configuration, i.e., the default User Interface allows the user to select among these three types of membership where appropriate.
Creation Privileges	<p>These privileges determine what an Entity can do with a Folder. They are:</p> <ul style="list-style-type: none"> • CREATE GROUP - can create a group(s) named with a naming stem, and • CREATE FOLDER - can assign who can CREATE folders (and sub folders) in a branch of the folder hierarchy.
Path	<p>A string that precedes the Group's name. By linking the ability to create groups to a specified folder (via the Creation privilege), the possibility that different groups can be given the same name is substantially reduced, and the name of each group can be made to reflect something about the authority under which it was created.</p> <p>...see Examples below.</p>
Subgroup	A Group that is a Direct Member of another group.

Entity	An abstraction of any object whose Memberships are to be managed by Grouper. Most Grouper deployments will manage entities that represent people and groups, but computers, accounts, services, or any other type of object maintained in a back-end identity store may be presented as an entity to Grouper by use of the Subject API.
Type	<p>There are two distinct uses for this term in Grouper.</p> <ul style="list-style-type: none"> • Group Type - each Group has one or more group types associated with it. The Grouper distribution contains support for a single group type called "base", but sites may register additional types, together with the attributes and lists associated with them, within their Grouper installation. Doing so enables management of groups with a richer information model or a more diverse set of information models. • Entity Type - the Subject API v0.2.1 that Grouper 1.3 relies on uses the notion of a subject type, such as "person", "group", or "computer", etc.

Examples

Step 1: Create a Root Folder

In the example below, a root Folder is first created. Note: creating a folder is required prior to the creation of any groups.

Folder uofc

attribute	value
folder	empty
ID	uofc
name	The University Of Chicago
ID path	uofc
path	The University Of Chicago

Step 2: Create a Group

Next, a group may be created using the "uofc" naming stem.

Group uofc:exec_council

attribute	value
folder	uofc

<i>ID</i>	exec_council
name	Executive Council
ID path	uofc:exec_council
path	The University of Chicago:Executive Council

Step 3: Create a Subordinate Folder and Group

Folder ID and Path values propagate down through subordinate folders, e.g the Biological Sciences Division within U of C:

Folder uofc:bsd

<i>attribute</i>	<i>value</i>
<i>folder</i>	uofc
<i>ID</i>	bsd
name	Biological Sciences Division
ID path	uofc:bsd
path	The University Of Chicago:Biological Sciences Division

Again, a group is created, e.g., the Enterprise Information Systems staff, with the above folder, and is displayed as follows:

Group uofc:bsd:eis_staff

<i>attribute</i>	<i>value</i>
<i>folder</i>	uofc:bsd
ID	eis_staff
name	Enterprise Information Systems staff
ID path	uofc:bsd:eis_staff
path	The University Of Chicago:Biological Sciences Division:Enterprise Information Systems staff

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper - Loader

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Grouper loader v1.4.0

Here is a grouper loader which can be used to automatically manage grouper memberships based on a data source.

Penn is using it in production to load membership for groups, and for groups of groups (in Penn's case, org lists).

One-time setup in your grouper database

To make a dynamic (loadable) group, first you need the correct metadata in grouper. The easiest way is to set the `grouper-loader.properties` key `loader.autoadd.typesAttributes` to `true`. If you don't want to do that, then here is the setup in GSH:

```
subj=SubjectFinder.findById("GrouperSystem")
sess=GrouperSession.start(subj)
type=GroupType.createType(sess, "grouperLoader")
read=Privilege.getInstance("read")
admin=Privilege.getInstance("admin")
type.addAttribute(sess, "grouperLoaderType", read, admin, true)
type.addAttribute(sess, "grouperLoaderDbName", read, admin, true)
type.addAttribute(sess, "grouperLoaderScheduleType", read, admin, true)
type.addAttribute(sess, "grouperLoaderQuery", read, admin, true)
type.addAttribute(sess, "grouperLoaderQuartzCron", read, admin, false)
type.addAttribute(sess, "grouperLoaderIntervalSeconds", read, admin, false)
type.addAttribute(sess, "grouperLoaderPriority", read, admin, false)
type.addAttribute(sess, "grouperLoaderAndGroups", read, admin, false)
type.addAttribute(sess, "grouperLoaderGroupTypes", read, admin, false)
type.addAttribute(sess, "grouperLoaderGroupsLike", read, admin, false)
type.addAttribute(sess, "grouperLoaderGroupQuery", read, admin, false)
```

Note that a loadable group has the type "grouperLoader", and there are some attributes that you can set about the group:

- [grouperLoaderType](#): there will be various types to choose from, currently only `SQL_SIMPLE` is available, which is a group whose membership is fed from a query, and the whole query's results will be the groups results (not incremental). `SQL_GROUP_LIST` is available and requires a `group_name` column in query, so one query can control multiple group memberships. This will default to `SQL_SIMPLE` if no `group_name` before the `FROM` in the query. Else it will be `SQL_GROUP_LIST`.
- `grouperLoaderDbName`: if it is a sql based type, this is the name in the `grouper-loader.properties` of the db connection properties. If this is set to "grouper" that is a special reserved term for the grouper db (in `grouper.hibernate.properties`) Here is a snippet from `grouper-loader.properties`

```
# specify the db connection with user, pass, url, and driver class
# the string after "db." is the name of the connection, and it should not have
# spaces or other special chars in it
db.warehouse.user = mylogin
db.warehouse.pass = secret
db.warehouse.url = jdbc:mysql://localhost:3306/grouper
db.warehouse.driver = com.mysql.jdbc.Driver
```

- [grouperLoaderScheduleType](#): Grouper-loader uses the quartz open source job scheduler, and currently supports two schedule types (note, that a job will not start if a previous run has not finished. This defaults to `CRON` if there is a `CRON` filled in, else it defaults to `START_TO_START_INTERVAL`
 - `CRON`: This is a [cron-like syntax that I think is quartz specific](#)
 - `START_TO_START_INTERVAL`: This is a repeated schedule that runs based on a delay from the start of one run to the start of another run
- `grouperLoaderQuery`: This is the query to run in the DB, which must have certain columns required or optional based on the `grouperLoaderType`. e.g. for `SQL_SIMPLE`, the `SUBJECT_ID` is required, and the `SUBJECT_SOURCE_ID` is optional. If your DB supports views, might not be a bad idea to

link query up to a view so you can easily see what it will return and change it without affecting the group attribute. But will work with any select query. This is required

- grouperLoaderQuartzCron: If a CRON schedule type, this is the cron setting string from the quartz product to run a job daily, hourly, weekly, etc: <http://www.opensymphony.com/quartz/wikidocs/TutorialLesson6.html>
- grouperLoaderIntervalSeconds: If a START_TO_START_INTERVAL schedule type, this is the number of seconds between the start of one run to the start of another run. This defaults to daily if not filled in. Note, for daily jobs, it is probably better to use cron so it won't fire up each time the loader is restarted.
- grouperLoaderPriority: Quartz has a fixed threadpool (max configured in the grouper-loader.properties), and when the max is reached, then jobs are prioritized by this integer. The higher the better, and the default if not set is 5.
- grouperLoaderAndGroups: If you want to restrict membership in the dynamic group based on other group(s), put the list of group names here comma-separated
- grouperLoaderGroupTypes: whatever you put in the value should be comma separated GroupTypes which will be applied to the loaded groups. The reason this enhancement exists is so we can do a SQL_GROUP_LIST query and attach addIncludeExclude to the groups. Note, if you do this (or use some requireGroups), the group name in the loader query should end in the system of record suffix, which by default is _systemOfRecord.
- grouperLoaderGroupsLike attribute: this should be a sql like string (e.g. school:orgs:%org %_systemOfRecord), and the loader should be able to query group names to see which names are managed by this loader job. So if a group falls off the loader resultset (or is moved), this will help the loader remove the members from this group. Note, if the group is used anywhere as a member or composite member, it won't be removed. All include/exclude/requireGroups will be removed. Though the two groups, include and exclude, will not be removed if they have members. There is a grouper-loader.properties setting to note remove loader groups if empty and not used:

#if using a sql table, and specifying the name like string, then should the group (in addition to memberships)
be removed if not used anywhere else?

loader.sqlTable.likeString.removeGroupIfNotUsed = true

- grouperLoaderGroupQuery: query (optional) for SQL_GROUP_LIST which should return cols: group_name, group_display_name (optional), group_description (optional) which if there are used for the group display and extension. Note: the parent stem display names are only changed when creating them. This should return all groups in the membership list, and if not there, its ok, the extension will be used as display extension, and a generated description. Note: the display name is the display path, with the display extension of each parent stem. If there is a column named any of the following: readers, viewers, admins, updaters, optins, optouts, then the data in the column (comma separated subjectId's or subjectIdentifiers (which can include group names) will be assigned to that privilege list. Note, existing assignments to that privilege list will not be removed, so if you remove an item from the query, you will need to manually remove it from the groups. This is a way to have a loaderJob-wide list of readers or viewers.

Configure a loadable group (obviously any number of dynamic loadable groups can exist at once)

With GSH, it would look like this:

```
group=getGroups("aStem:aGroup2")
groupAddType("aStem:aGroup2", "grouperLoader")
setGroupAttr("aStem:aGroup2", "grouperLoaderDbName", "grouper")
setGroupAttr("aStem:aGroup2", "grouperLoaderType", "SQL_SIMPLE")
setGroupAttr("aStem:aGroup2", "grouperLoaderScheduleType", "START_TO_START_INTERVAL")
setGroupAttr("aStem:aGroup2", "grouperLoaderQuery", "select SUBJECT_ID, SUBJECT_SOURCE_ID from
agroup2_v")
setGroupAttr("aStem:aGroup2", "grouperLoaderIntervalSeconds", "30")
```

You can also use the UI, here are screenshots (obviously these need some work).



My enrollment

[My memberships](#)

[Join groups](#)

My responsibilities

[Manage groups](#)

[Create groups](#)

My tools

Explore

[Search](#)

[Group workspace](#)

[Entity workspace](#)

[Help](#)

EXPLORE

Edit group [?](#)

Current location is:

[Root](#): [a](#) stem: [aGroup2](#)

<u>Name</u>	<input type="text" value="aGroup2"/>
<u>ID</u>	<input type="text" value="aGroup2"/>
<u>Description</u>	<input type="text" value="some description"/>
Assign privileges to everyone	<input type="checkbox"/> admin <input type="checkbox"/> update <input checked="" type="checkbox"/> read <input checked="" type="checkbox"/> view <input type="checkbox"/>
Select group types	<input type="checkbox"/> dynamic <input checked="" type="checkbox"/> grouperLoader

Save

Back to group summary

Grouper is sponsored by



Welcome Hyzer, Chris : mchyzer, Staff, ISC Administrative Systems Tools and Technologies, PROGRAMME

EXPLORE

Edit attributes

Current location is:

Root: a stem: aGroup2

Group type	Attribute	Value
grouperLoader		
	grouperLoaderDbName	grouper
	grouperLoaderIntervalSeconds	30
	grouperLoaderPriority	
	grouperLoaderQuartzCron	
	grouperLoaderQuery	select SUBJECT_J
	grouperLoaderScheduleType	START_TO_START
	grouperLoaderType	SQL_SIMPLE

Save attributes and finish

Save attributes and add members

sponsored by

NET.

Run grouper loader

The first time you run, it will probably fail, and give you DDL in the logs to run in your database (to add a couple of tables). Run the scripts and you should be all set.

Run with:

```
GROUPER_HOME/bin/gsh.sh -loader
```

This will kick off as a command line program that you will want to run as a service. This process will be always running, the scheduler will schedule the jobs. You should monitor the process with a monitoring tool like nagios or whatever you use at your institution so that you know when it is not up.

You can also run a one-timer via gsh. This is useful to run once at the beginning, and not have to wait for the schedule. Or to troubleshoot e.g.

```
loaderGroup = GroupFinder.findByName(GrouperSession.startRootSession(), "school:orgs:orgGroup");
loaderRunOneJob(loaderGroup);
loaderRunOneJob("MAINTENANCE_cleanLogs");
```

Logging of jobs in DB

Each job (and subjob if the job manages multiple things) will have an entry in the grouploder_log table. This will show the following information. This can be used to tune performance problems, see which jobs have unresolvable subjects, verify that jobs are running, etc.

COLUMN_NAME	DATA_TYPE
ID	VARCHAR2
JOB_NAME	VARCHAR2
STATUS	VARCHAR2
STARTED_TIME	TIMESTAMP(6)
ENDED_TIME	TIMESTAMP(6)
MILLIS	NUMBER
MILLIS_GET_DATA	NUMBER
MILLIS_LOAD_DATA	NUMBER
JOB_TYPE	VARCHAR2
JOB_SCHEDULE_TYPE	VARCHAR2
JOB_DESCRIPTION	VARCHAR2
JOB_MESSAGE	VARCHAR2
HOST	VARCHAR2
GROUP_UUID	VARCHAR2
JOB_SCHEDULE_QUARTZ_CRON	VARCHAR2
JOB_SCHEDULE_INTERVAL_SECONDS	NUMBER
JOB_SCHEDULE_PRIORITY	NUMBER
LAST_UPDATED	TIMESTAMP(6)
UNRESOLVABLE_SUBJECT_COUNT	NUMBER
INSERT_COUNT	NUMBER
UPDATE_COUNT	NUMBER
DELETE_COUNT	NUMBER
TOTAL_COUNT	NUMBER
PARENT_JOB_NAME	VARCHAR2
PARENT_JOB_ID	VARCHAR2

You can also look at log4j debug log messages, and info log messages (less frequent). to see these, set log level in log4j.properties

```
## Log debug info on loader to see progress etc
log4j.logger.edu.internet2.middleware.grouper.app.loader = INFO
-or-
log4j.logger.edu.internet2.middleware.grouper.app.loader = DEBUG
```

Misc

- You can set transaction level in the grouper-loader.properties. It defaults to not use transaction since for huge groups, you might have memory or db problems
- By default only wheel group members can edit the grouperLoader type or attributes. You can edit these settings in the grouper.properties in the type security part.

Possible to do's

- add subject source to group attribute (or default at least) so it doesnt have to be a sql column if all are the same
- make specific blackout times (runtime via config file?)
- make jobs based on person trigger. If there is a query that says when people change, then update that person's memberships in the groups that are dynamic based on that person-change-query
- make full refresh jobs for the incremental jobs (e.g. weekly)
- save quartz info to DB so that stopping / starting isnt that drastic
- make more job types (jndi?)
- make an RMI server so that interactions can happen at runtime (to see status, stop/start jobs, etc). Maybe this would happen from gsh

- try the name pattern after loader is done, and if the number of groups is less than the number of groups in this round of loader, set the job status to WARNING and add a descriptive message

Grouper loader classlist example from Penn

This page last changed on Oct 19, 2009 by mchzyer@idp.protectnetwork.org.

Summary

Penn implemented class-lists with the Grouper Loader which includes lists of students, instructors, assistants (to instructors), guests, and all members. The students, instructors, and assistants are fed from our student system, so they should be include/exclude. The guests are an ad hoc group. The "all" members should include the groups: students, instructors, assistants, guests. There can be cross listings (multiple names for a course), and the cross listing groups should just contain the primary course groups (5 groups: all, students, instructors, guests, assistants). Security should be setup on all courses such that there is a global readers group, a global updaters group, and in each course the instructors and assistants should be able to read and update the appropriate groups. Note you need Grouper 1.4 built after 10/18/2009, or 1.5+ for all of this to work properly.

Naming

Our naming convention is similar to name groups like this (this is just one course of many):

penn:community:student:**course** - root of all courses
penn:community:student:**loader** - holds groups with loader rules
penn:community:student:**security** - holds security groups (e.g. readers or updaters of all courses)

penn:community:student:course:2009C:EG:CIS:120:203:**all**

- holds all members for term 2009C, engineering, computer science, course 120, section 203
- includes the students, instructors, assistants, and guests

penn:community:student:course:2009C:EG:CIS:120:203:**assistants**

- overall assistants to instructors (system of record, add includes, remove excludes)

penn:community:student:course:2009C:EG:CIS:120:203:**assistants excludes**

- assistants to instructors excludes list (remove from system of record)

penn:community:student:course:2009C:EG:CIS:120:203:**assistants includes**

- assistants to instructors includes list

penn:community:student:course:2009C:EG:CIS:120:203:**assistants_systemOfRecord**

- assistants to instructors system or record from student system

penn:community:student:course:2009C:EG:CIS:120:203:**guests**

- ad hoc guests group to course

penn:community:student:course:2009C:EG:CIS:120:203:**instructors**

- overall instructors group (system of record, add includes, subtract excludes)

penn:community:student:course:2009C:EG:CIS:120:203:**instructors excludes**

- instructors excludes, will block members from system of record

penn:community:student:course:2009C:EG:CIS:120:203:**instructors includes**

- instructors includes, to add members to the system of record

penn:community:student:course:2009C:EG:CIS:120:203:**instructors_systemOfRecord**

- instructors system of record, form student system

penn:community:student:course:2009C:EG:CIS:120:203:**students**

- overall students list, made up of system of record, add includes, subtract excludes

penn:community:student:course:2009C:EG:CIS:120:203:**students excludes**

- students excludes removes students from the system of record

penn:community:student:course:2009C:EG:CIS:120:203:**students includes**

- students includes add members to the system of record

penn:community:student:course:2009C:EG:CIS:120:203:**students_systemOfRecord**

- students system of record is fed from student system

Security design

There are two high level global security groups for system or admins:

penn:community:student:security:courseReaders (can read all course membership lists)
penn:community:student:security:courseUpdaters (can update all course membership lists)

The instructors and assistants can read all groups in the course (listed above), and all cross-listed courses to that course.

They can update:

- guests
- instructors includes
- instructors excludes
- students includes
- students excludes
- assistants includes
- assistants excludes

Note, no cross listed courses are editable since they mirror the primary course

Primary course students list

For primary courses (not cross listings), lets setup a job which manages the memberships and security: ! studentsLoader.gif!This shows:

- The data is coming from our warehouse (configured in grouper-loader.properties)
- There is a grouper group type applied to the group, which is addIncludeExclude. This automatically creates the includes, excludes, and overall groups
- The job runs once per day at 8am
- It is a SQL_GROUP_LIST which manages many groups at once
- The membership query: (penn_id and group_name one), returns data like this:
12345678 penn:community:student:course:2009C:EG:BE:099:001:students_systemOfRecord
12345679 penn:community:student:course:2009C:EG:BE:100:001:students_systemOfRecord
12345677 penn:community:student:course:2009C:EG:BE:100:001:students_systemOfRecord
12345676 penn:community:student:course:2009C:EG:BE:100:001:students_systemOfRecord
12345675 penn:community:student:course:2009C:EG:BE:101:001:students_systemOfRecord
12345678 penn:community:student:course:2009C:EG:BE:101:001:students_systemOfRecord
- The groups query, controls empty groups, names groups, and sets the security. It returns data that looks like this:

GROUP_NAME	READERS	UPDATERS
penn:community:student:course:2009C:EG:BE:099:001:students	penn:community:student:security:courseUp	penn:community:student:security:courseUp
penn:community:student:course:2009C:EG:BE:100:001:students	penn:community:student:security:courseUp	penn:community:student:security:courseUp
penn:community:student:course:2009C:EG:BE:101:001:students	penn:community:student:security:courseUp	penn:community:student:security:courseUp

- Note that the queries in the loader job are built on views, it is important that you do this so you can see when upgrades make things not compile, and so the SQL can be changed without editing the job, and since there is a character limit for attribute values
- First I create a stem name view, which includes if a course is primary or not (cross listed). Note this also restricts which terms to select from

```
CREATE OR REPLACE VIEW COURSE_GROUP_STEM_NAME_V
(GROUP_NAME_STEM, TERM, SECTION_ID, XLIST_PRIMARY, PRIMARY_COURSE)
AS select 'penn:community:student:course:' || trim(cs.term) || ':' || trim(cs.section_school)
|| ':' || trim(cs.subject_area)
|| ':' || trim(cs.course_number) || ':' || trim(cs.section_number)
as group_name_stem,
cs.term, cs.section_id, cs.XLIST_PRIMARY, decode(cs.SECTION_ID, cs.XLIST_PRIMARY, 'T', 'F') as
primary_course
from course_section cs, course_section cs_xlist_primary, course_term_v ctv ...
```

- Then I built on that to create a course group name which has all the names of the groups (for easy reuse)

```

CREATE OR REPLACE VIEW COURSE_GROUP_NAME_V
(GROUP_NAME_STEM, STUDENTS, STUDENTS_SYSTEMOFRECORD, STUDENTS_INCLUDES,
STUDENTS_EXCLUDES,
INSTRUCTORS, INSTRUCTORS_SYSTEMOFRECORD, INSTRUCTORS_INCLUDES, INSTRUCTORS_EXCLUDES,
ASSISTANTS,
ASSISTANTS_SYSTEMOFRECORD, ASSISTANTS_INCLUDES, ASSISTANTS_EXCLUDES, ALLGROUP, GUESTS,
SECTION_ID, TERM, XLIST_PRIMARY, PRIMARY_COURSE)
AS
select
cgsnv.GROUP_NAME_STEM,
cgsnv.GROUP_NAME_STEM || ':students' as students,
cgsnv.GROUP_NAME_STEM || ':students_systemOfRecord' as students_systemOfRecord,
cgsnv.GROUP_NAME_STEM || ':students_includes' as students_includes,
cgsnv.GROUP_NAME_STEM || ':students_excludes' as students_excludes,
cgsnv.GROUP_NAME_STEM || ':instructors' as instructors,
cgsnv.GROUP_NAME_STEM || ':instructors_systemOfRecord' as instructors_systemOfRecord,
cgsnv.GROUP_NAME_STEM || ':instructors_includes' as instructors_includes,
cgsnv.GROUP_NAME_STEM || ':instructors_excludes' as instructors_excludes,
cgsnv.GROUP_NAME_STEM || ':assistants' as assistants,
cgsnv.GROUP_NAME_STEM || ':assistants_systemOfRecord' as assistants_systemOfRecord,
cgsnv.GROUP_NAME_STEM || ':assistants_includes' as assistants_includes,
cgsnv.GROUP_NAME_STEM || ':assistants_excludes' as assistants_excludes,
cgsnv.GROUP_NAME_STEM || ':all' as allGroup,
cgsnv.GROUP_NAME_STEM || ':guests' as guests,
cgsnv.SECTION_ID,
cgsnv.TERM,
cgsnv.XLIST_PRIMARY,
cgsnv.PRIMARY_COURSE
from COURSE_GROUP_STEM_NAME_V cgsnv

```

- Then we can build course list view (note the members of cross listed courses are includes in the primary course)

```

CREATE OR REPLACE VIEW COURSE_PRIMARY_STUDENT_LIST_V
(GROUP_NAME, PENN_ID, TERM, SECTION_ID)
AS
SELECT
  distinct cgnv.STUDENTS_SYSTEMOFRECORD as group_name,
  eav.PENN_ID,
  eav.TERM,
  cs.XLIST_PRIMARY section_id
FROM
  DWADMIN.ENROLLMENT_ALL_V eav,
  DWADMIN.COURSE_SECTION cs,
  course_group_name_v cgnv
WHERE
  eav.SECTION_ID=cs.section_id and eav.TERM=cs.TERM
  and cgnv.SECTION_ID = cs.XLIST_PRIMARY
  and cgnv.TERM = cs.term
  and cgnv.PRIMARY_COURSE = 'T'

```

- Finally we make the primary students group view for the students list which includes the security aspects

```

CREATE OR REPLACE VIEW COURSE_PRIM_STUD_GROUP_QUERY_V
(GROUP_NAME, READERS, UPDATERS, SECTION_ID, TERM)
AS
select cgnv.STUDENTS_SYSTEMOFRECORD as group_name,

```

```
'penn:community:student:security:courseReaders, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as readers,
'penn:community:student:security:courseUpdaters, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as updaters, cgnv.SECTION_ID, cgnv.TERM
from course_group_name_v cgnv where cgnv.PRIMARY_COURSE = 'T'
```

Primary course instructors list

Note: this is called instructors since it is more of a role than a title, so it is not "Professors". Here is the loader screenshot:

EXPLORE
Group summary ⓘ

Current location is:

Root: penn: community: student: loader: instructorGroups

<u>Name</u>	instructorGroups		
<u>Path</u>	penn:community:student:loader:instructorGroups		
<u>Description</u>	loader job for instructors		
<u>ID</u>	instructorGroups		
<u>ID Path</u>	penn:community:student:loader:instructorGroups		
<u>UUID</u>	b7fb872dde484370bf903725bf0a16f5		
<u>Types</u>	grouperLoader	grouperLoaderAndGroups	
		grouperLoaderDbName	warehouse
		grouperLoaderGroupQuery	select gro from COURSE_
		grouperLoaderGroupTypes	addInclud
		grouperLoaderGroupsLike	
		grouperLoaderIntervalSeconds	
		grouperLoaderPriority	
		grouperLoaderQuartzCron	0 5 8 * * *
		grouperLoaderQuery	select gro subject_id COURSE_
		grouperLoaderScheduleType	CRON
		grouperLoaderType	SQL_GRO

- This is a query which runs from our warehouse connection (configured in grouper-loader.properties), add include/exclude type to the groups for include exclude lists, runs at 8:05 in the morning
- The membership query builds on a view which gives instructors or admins for a course (or any of its cross listings)

```
select cpiav.INSTRUCTOR_PENN_ID penn_id,
cpiav.instructors_GROUP_NAME as group_name
from COURSE_PRIMARY_INSTR_ASST_V cpiav
where cpiav.INSTRUCTOR_LOAD <> 0
```

- This returns data like this:

PENN_ID	GROUP_NAME
12345678	penn:community:student:course:2009C:EG:BE:200:204:ins
12345678	penn:community:student:course:2009C:EG:MGMT:586:501:
12345677	penn:community:student:course:2009C:EG:MSE:990:001:in
12345676	penn:community:student:course:2009C:EG:BE:497:001:ins

- The groups view gives the name of the group and the security rules

```
CREATE OR REPLACE VIEW COURSE_PRI_INSTR_GROUP_QUERY_V
(GROUP_NAME, READERS, UPDATERS, SECTION_ID, TERM)
AS
select cgnv.instructors_SYSTEMOFRECORD as group_name,
'penn:community:student:security:courseReaders, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as readers,
'penn:community:student:security:courseUpdaters, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as updaters, cgnv.SECTION_ID, cgnv.TERM
from course_group_name_v cgnv
```

- This returns data that looks like this:

GROUP_NAME	READERS	UPDATERS	SECTION_ID	TERM
penn:community:student:course:2009C:EG:BE:200:204:ins	penn:community:student:course:2009C:EG:BE:200:204:ins	penn:community:student:course:2009C:EG:BE:200:204:ins	258:401	2009
penn:community:student:course:2009C:EG:MGMT:586:501:	penn:community:student:course:2009C:EG:MGMT:586:501:	penn:community:student:course:2009C:EG:MGMT:586:501:	258:401	2009
penn:community:student:course:2009C:EG:MSE:990:001:in	penn:community:student:course:2009C:EG:MSE:990:001:in	penn:community:student:course:2009C:EG:MSE:990:001:in	727:401	2009
penn:community:student:course:2009C:EG:BE:497:001:ins	penn:community:student:course:2009C:EG:BE:497:001:ins	penn:community:student:course:2009C:EG:BE:497:001:ins	727:401	2009

Primary course assistants list

- Note, this is not called "teaching assistants" since this group is more of a role than a title

- Here is the loader screenshot

Group summary ⓘ

Current location is:

Root:

penn:

community:

student:

loader:

assistantGroups

<u>Name</u>	assistantGroups
<u>Path</u>	penn:community:student:loader:assistantGroups
<u>Description</u>	loader job for assistant groups
<u>ID</u>	assistantGroups
<u>ID Path</u>	penn:community:student:loader:assistantGroups
<u>UUID</u>	bb32a6e1b2f144bcb2bf5b6e059e7932
<u>Types</u>	<div>grouperLoader</div> <div> <div>grouperLoaderAndGroups</div> <div>grouperLoaderDbName</div> <div>grouperLoaderGroupQuery</div> <div>grouperLoaderGroupTypes</div> <div>grouperLoaderGroupsLike</div> <div>grouperLoaderIntervalSeconds</div> <div>grouperLoaderPriority</div> <div>grouperLoaderQuartzCron</div> <div>grouperLoaderQuery</div> <div>grouperLoaderScheduleType</div> <div>grouperLoaderType</div> </div>

- This runs from the data warehouse (configured in grouper-loader.properties), at 8:10 in the morning, and includeExclude is added for the include and exclude lists
- Here is a view of the members:

```
select cpiav.INSTRUCTOR_PENN_ID penn_id,
cpiav.ASSISTANTS_GROUP_NAME as group_name
from COURSE_PRIMARY_INSTR_ASST_V cpiav
where cpiav.INSTRUCTOR_LOAD = 0
```

- Here is a sample of the data returned:

PENN_ID	GROUP_NAME
38430684	penn:community:student:course:2009C:EG:MEAM:210:203:
45535464	penn:community:student:course:2009C:EG:CIS:120:203:as
24636453	penn:community:student:course:2009C:EG:ESE:112:001:as

- Here is the view of the group query:

```
CREATE OR REPLACE VIEW COURSE_PRIM_ASST_GROUP_QUERY_V
(GROUP_NAME, READERS, UPDATERS, SECTION_ID, TERM)
AS
select cgnv.assistants_SYSTEMOFRECORD as group_name,
'penn:community:student:security:courseReaders, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as readers,
'penn:community:student:security:courseUpdaters, ' || cgnv.ASSISTANTS
|| ',' || cgnv.INSTRUCTORS as updaters, cgnv.SECTION_ID, cgnv.TERM
from course_group_name_v cgnv
```

- Here is an example of data returned:

GROUP_NAME	READERS	UPDATERS	SECTION_ID	TERM
penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters
penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters
penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters

Primary guests group

The guests group is an ad hoc group, but we want it auto-created, and managed by the loader for security. The grouper loader doesn't really support this at the moment (we should add support), but one workaround is just to sync the members from the

grouper membership table (immediate memberships). Here is the loader screenshot

EXPLORE

Group summary ⓘ

Current location is:

Root:

penn:

community:

student:

loader:

guestGroups

<u>Name</u>	guestGroups
<u>Path</u>	penn:community:student:loader:guestGroups
<u>Description</u>	
<u>ID</u>	guestGroups
<u>ID Path</u>	penn:community:student:loader:guestGroups
<u>UUID</u>	fbf17c263d5f4375ba45359f778c57ba
<u>Types</u>	<div>grouperLoader</div> <div> <div>grouperLoaderAndGroups</div> <div> <div>grouperLoaderDbName</div> <div>grouper</div> </div> <div>grouperLoaderGroupQuery</div> <div>select group_name, r COURSE_GUESTS_G</div> </div> <div>grouperLoaderGroupTypes</div> <div>grouperLoaderGroupsLike</div> <div>grouperLoaderIntervalSeconds</div> <div>grouperLoaderPriority</div> <div>grouperLoaderQuartzCron</div> <div>0 20 8 * * *</div> <div>grouperLoaderQuery</div> <div>select group_NAME, S from COURSE_GUES</div> <div>grouperLoaderScheduleType</div> <div>CRON</div> <div>grouperLoaderType</div> <div>SQL_GROUP_LIST</div>

- Note, this is driven from the grouper db connection, since it needs to hit the grouper registry for membership info. The members view (needs to join with a db link to the warehouse to get the group names), note: driving site means that the names should be brought from the warehouse to the grouper registry, then calculated there. It runs at 8:15 (even though screen says 8:20, this was changed). Note this query will need to change in 1.5

```
CREATE OR REPLACE VIEW COURSE_GUESTS_MEMBERS_V
(GROUP_NAME, SUBJECT_ID, SUBJECT_SOURCE_ID)
AS
select /*+DRIVING_SITE(gga)*/
cgnv.guests as group_name, gm.subject_id, gm.SUBJECT_SOURCE as subject_source_id
from course_group_name_v@authzadm_warehouse cgnv,
grouper_attributes gga,
grouper_fields ggf, grouper_memberships gms, grouper_fields gf,
grouper_members gm
where gga.field_id = ggf.ID and ggf.NAME = 'name'
and gga.VALUE = cgnv.guests
and cgnv.primary_course = 'T'
and gms.OWNER_ID = gga.GROUP_ID
and gms.FIELD_ID = gf.ID
and gms.MSHIP_TYPE = 'immediate'
and gf.NAME = 'members'
```



```
and gms.MEMBER_ID = gm.id
```

- This gives data like this:

GROUP_NAME	SUBJECT_ID	SUBJECT_SOURCE_ID
penn:community:student:course:2009C-FG-BF-109-500	ANTH:258:401:guests	pennperson
penn:community:student:course:2009C-FG-BF-109-500	ANTH:258:402:guests	pennperson

- The groups query is run via db link as well (since the members query must be run from grouper registry, and both queries must run from same db). Here is the view (on warehouse)

```
CREATE OR REPLACE VIEW COURSE_GUESTS_GROUP_QUERY_V
(GROUP_NAME, READERS, UPDATERS)
AS
select primary_cgnv.guests as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers,
'penn:community:student:security:courseUpdaters, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as updaters
from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'
```

- Here is an example of the data:

GROUP_NAME	READERS	UPDATERS
penn:community:student:course:2009C-FG-BF-109-500	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters
penn:community:student:course:2009C-FG-BF-109-500	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters
penn:community:student:course:2009C-FG-BF-109-500	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters	penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters, penn:community:student:security:courseReaders, penn:community:student:security:courseUpdaters

The 'All' groups for primary courses

Now that we have the students, instructors, assistants and guests, we can roll up all of those into an 'all' group. Here is the loader screenshot:

Group summary ⓘ

Current location is:
Root: penn: community: student: loader: allGroups

Name	allGroups
Path	penn:community:student:loader:allGroups
Description	
ID	allGroups
ID Path	penn:community:student:loader:allGroups
UUID	d6656d0c5042405e9805f01043395246
Types	<div><div>grouperLoader</div><div><div>grouperLoaderAndGroups</div><div>grouperLoaderDbName</div><div>grouperLoaderGroupQuery</div><div>grouperLoaderGroupTypes</div><div>grouperLoaderGroupsLike</div><div>grouperLoaderIntervalSeconds</div><div>grouperLoaderPriority</div><div>grouperLoaderQuartzCron</div><div>grouperLoaderQuery</div><div>grouperLoaderScheduleType</div><div>grouperLoaderType</div></div></div>

Show entities with ADMIN privileges

- This runs at 8:20 (change from before, the screen says 8:15). Since this builds on existing groups and we need to know the group id for the subject id.
- Here is the membership view (in warehouse db):

```
CREATE OR REPLACE VIEW COURSE_ALL_GROUP_MEMBERS_V
(GROUP_NAME, MEMBER_GROUP_NAME)
AS
select primary_cgnv.allgroup as group_name,
primary_cgnv.ASSISTANTS as member_group_name
from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'
union all
select primary_cgnv.allgroup as group_name,
primary_cgnv.guests as member_group_name
from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'
union all
select primary_cgnv.allgroup as group_name,
primary_cgnv.students as member_group_name
```

```

from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'
union all
select primary_cgnv.allgroup as group_name,
primary_cgnv.instructors as member_group_name
from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'

```

- Here is the membership view (in grouper db): [note, this query will change in 1.5)

```

CREATE OR REPLACE VIEW COURSE_ALL_GROUP_V
(GROUP_NAME, SUBJECT_ID, SUBJECT_SOURCE_ID, MEMBER_GROUP_NAME)
AS
select /*+DRIVING_SITE(gga)*/
cagmv.GROUP_NAME,
gga.GROUP_ID as subject_id, 'g:gsa' as subject_source_id,
cagmv.member_group_name
from COURSE_ALL_GROUP_MEMBERS_V@authzadm_warehouse cagmv, grouper_attributes gga,
grouper_fields ggf
where gga.field_id = ggf.ID and ggf.NAME = 'name'
and gga.VALUE = cagmv.member_group_name

```

- Here is an example of the data returned:

GROUP_NAME	SUBJECT_ID	SUBJECT_SOURCE_ID	MEMBER_GROUP_NAME
penn:community:student:course:2009C:EG:BE:099:001	2009C:EG:BE:099:001	2009C:EG:BE:099:001	penn:community:student:course:2009C:EG:BE:099:001
penn:community:student:course:2009C:EG:BE:099:001	2009C:EG:BE:099:001	2009C:EG:BE:099:001	penn:community:student:course:2009C:EG:BE:099:001
penn:community:student:course:2009C:EG:BE:099:001	2009C:EG:BE:099:001	2009C:EG:BE:099:001	penn:community:student:course:2009C:EG:BE:099:001
penn:community:student:course:2009C:EG:BE:099:001	2009C:EG:BE:099:001	2009C:EG:BE:099:001	penn:community:student:course:2009C:EG:BE:099:001
penn:community:student:course:2009C:EG:BE:099:001	2009C:EG:BE:099:001	2009C:EG:BE:099:001	penn:community:student:course:2009C:EG:BE:099:001

- Here is the group view

```

CREATE OR REPLACE VIEW COURSE_ALL_GROUP_GROUP_QUERY_V
(GROUP_NAME, READERS, UPDATERS)
AS
select primary_cgnv.ALLGROUP as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers,
null as updaters
from course_group_name_v primary_cgnv
where primary_cgnv.primary_course = 'T'

```


- Here is an example of the data

GROUP_NAME	READERS	UPDATERS
penn:community:student:course:2009C:EG:BE:099:001	penn:community:student:security:courseReaders, penn:community:student:course:2009C:EG:BE:099:001:assistants,penn:community:student:course:2009C:EG:BE:099:001:instructors	







penn:community:student:course:2009C:EG:BE:100:001:all	security:courseReaders, penn:community:student:course:2009C:EG:BE:100:001:assistants,penn:com
---	--

Crosslisted course lists

This is the screenshot for the crosslist loader job


Group summary

Current location is:

 Root:
  penn:
  community:
  student:
  loader:
  crosslistGroups

<u>Name</u>	crosslistGroups
<u>Path</u>	penn:community:student:loader:crosslistGroups
<u>Description</u>	runs the groups for a cross listed course including student, professor, teach
<u>ID</u>	crosslistGroups
<u>ID Path</u>	penn:community:student:loader:crosslistGroups
<u>UUID</u>	2e207cde818b4106aaaa901d2d49a432
<u>Types</u>	<div> <div>grouperLoader</div> <div>grouperLoaderAndGroups</div> <div>grouperLoaderDbName</div> <div>grouperLoaderGroupQuery</div> <div>grouperLoaderGroupTypes</div> <div>grouperLoaderGroupsLike</div> <div>grouperLoaderIntervalSeconds</div> <div>grouperLoaderPriority</div> <div>grouperLoaderQuartzCron</div> <div>grouperLoaderQuery</div> <div>grouperLoaderScheduleType</div> <div>grouperLoaderType</div> </div> <div> <div>grouper</div> <div>select group_name</div> <div>course_xLIST_GRO</div> <div></div> <div></div> <div></div> <div></div> <div></div> <div>0 15 6 * * ?</div> <div>select GROUP_NAM</div> <div>SUBJECT_SOURCE</div> <div>CRON</div> <div>SQL_GROUP_LIST</div> </div>

Show entities with

ADMIN

privilege

- Again, since this is loading groups in other groups, we need to run from the grouper DB connection. It says 6:15, but this is really run at 8:30am.
- Here is the cross list members view in the warehouse:

```
CREATE OR REPLACE VIEW COURSE_XLIST_GROUP_MEMBERS_V
(GROUP_NAME, MEMBER_GROUP_NAME)
AS
select xlist_cgnv.ASSISTANTS as group_name,
primary_cgnv.ASSISTANTS as member_group_name
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
```

```

union all
select xlist_cgnv.instructors as group_name,
primary_cgnv.instructors as member_group_name
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.students as group_name,
primary_cgnv.students as member_group_name
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.ALLGROUP as group_name,
primary_cgnv.allgroup as member_group_name
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.GUESTS as group_name,
primary_cgnv.guests as member_group_name
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID

```

- Here is the members query in the grouper database

```

CREATE OR REPLACE VIEW COURSE_XLIST_V
(GROUP_NAME, SUBJECT_ID, SUBJECT_SOURCE_ID, XLIST_PRIMARY_GROUP_NAME)
AS
select cgm.GROUP_NAME,
ga.GROUP_ID as subject_id, 'g:gsa' as subject_source_id,
cgm.member_group_name as xlist_primary_group_name
from course_xLIST_GROUP_MEMBERS_V@authzadm_warehouse cgm, grouper_attributes ga,
grouper_fields gf
where ga.field_id = gf.ID and gf.NAME = 'name'
and ga.VALUE = cgm.member_group_name

```

- Here is a sample of the data returned

GROUP_NAME	SUBJECT_ID	SUBJECT_SOURCE_ID	XLIST_PRIMARY_GROUP_NAME
penn:community:student:course:200634200934534543457258231	200634200934534543457258231	g:gsa	penn:community:student:course:200634200934534543457258231
penn:community:student:course:200634200934534543457258231	200634200934534543457258231	g:gsa	penn:community:student:course:200634200934534543457258231
penn:community:student:course:200634200934534543457258231	200634200934534543457258231	g:gsa	penn:community:student:course:200634200934534543457258231

- Here is the all group query

```

CREATE OR REPLACE VIEW COURSE_XLIST_GROUP_GROUP_V

```

```

(GROUP_NAME, READERS)
AS
select xlist_cgnv.ASSISTANTS as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.INSTRUCTORS as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.STUDENTS as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.allgroup as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID
union all
select xlist_cgnv.guests as group_name,
'penn:community:student:security:courseReaders, ' || primary_cgnv.ASSISTANTS
|| ',' || primary_cgnv.INSTRUCTORS as readers
from course_group_name_v xlist_cgnv,
course_group_name_v primary_cgnv
where xlist_cgnv.PRIMARY_COURSE = 'F'
and primary_cgnv.primary_course = 'T'
and xlist_cgnv.XLIST_PRIMARY = primary_cgnv.SECTION_ID

```

- Here is sample data:

GROUP_NAME	READERS
penn:community:student:course:2009C:AS:ANTH:258:401	assistant:student:security:courseReaders, penn:community:student:course:2009C:EG:CIS:106:401:as
penn:community:student:course:2009C:FA:ARCH:727:401	instructors:student:security:courseReaders, penn:community:student:course:2009C:EG:IPD:527:401:as

- sdf

GrouperShell (gsh)

This page last changed on May 20, 2009 by [mchyzer](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

GrouperShell (gsh) as of v1.4.0

gsh is a command line shell for administering and interacting with the Grouper API. It can be used in both a batch and interactive manner. It is built on [Java BeanShell](#)

API Compability

gsh is now a core part of the Grouper API and so is always compatible with the current release.

Installation

When using the Grouper API source distribution, grouper.jar needs to be built before using gsh.sh for the first time:

```
cd $GROUPER_HOME
ant dist
```

Usage

For Windows use \$GROUPER_HOME\bin\gsh.bat

Run gsh as an interactive shell:

```
$GROUPER_HOME/bin/gsh.sh
```

Read gsh commands from STDIN:

```
$GROUPER_HOME/bin/gsh.sh -
```

Read gsh commands from a script file:

```
$GROUPER_HOME/bin/gsh.sh /path/to/your/script.gsh
```

Run Grouper utilities:

```
$GROUPER_HOME/bin/gsh.sh <option>
args: -h,           Prints this message
args: -check,       Performs startup check and enters an
                    interactive shell
args: -runarg <command> Run command (use \\n to separate commands)
args: -main <class> [args...]
    class,          Full class name (must have main method)
    args,           args as required by main method of class
args: -initEnv [<configDir>]
    On Windows sets GROUPER_HOME and adds GROUPER_HOME/bin to path
    For *nix 'source gsh.sh' for the same result
    configDir optionally adds an alternative conf directory than
    GROUPER_HOME/conf to the classpath
args: (-xmlimport | -xmlexport | -loader | -test | -registry | -usdu |
    findbadmemberships)
    Enter option to get additional usage for that
    option
    -xmlimport,      Invokes XmlExporter
    -xmlexport,      Invokes XmlImporter
    -loader,         Invokes GrouperLoader
    -registry,       Manipulate the Grouper schema and install
                    bootstrap data
```

- test, Run JUnit tests
- usdu, Invoke USDU - Unresolvable Subject Deletion Utility
- findbadmemberships, Check for membership data inconsistencies

Supported Commands

Grouper API methods

Any Grouper API method can be directly invoked just by referencing it, inclusive of the class in which it is defined. Methods return a java object which can be stored in a variable. For example, the following gsh session determines all of the groups to which a given subject belongs:

```
gsh 0% subj = findSubject("SD00125")
subject: id='SD00125' type='person' source='kitn-person' name='Barton, Tom'
gsh 1% sess = GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession: 29c40f97-9fb0-4e45-88bc-
a14877a6c9b5,'SD00125','person'
gsh 2% member = MemberFinder.findBySubject(sess, subj)
member: id='SD00125' type='person' source='kitn-person' uuid='d0fa765e-1439-4701-89b1-9b08b4ce9daa'
gsh 3% member.getGroups()
group: name='etc:sysadmingroup' displayName='Grouper Administration:SysAdmin Group' uuid='6f77fb36-
b466-481a-84a7-7af609f1ad09'
```

Groups

Command	Description
addGroup(parent stem name, extension, displayExtension)	Add group to registry
delGroup(name)	Delete group from registry
getGroupAttr(group name, attr)	Get value of group attribute
getGroups(name)	Find all groups with a matching naming attribute value, returns a Set of groups
setGroupAttr(group name, attr, value)	Set value of group attribute
GroupFinder.findByName(grouperSession, name)	Find one group by name
GroupFinder.findByUuid(grouperSession, name)	Find one group by uuid

Group Types

Command	Description
groupAddType(group name, type name)	Add type to group
groupDelType(group name, type name)	Delete type from group
groupGetTypes(group name)	Get group's types
groupHasType(group name, type name)	Check whether group had type
typeAdd(type name)	Create custom group type

typeAddAttr(type name, attr name, read, write, required)	Create custom group attribute. <i>read</i> and <i>write</i> must be an AccessPrivilege (e.g. AccessPrivilege.ADMIN)
typeAddList(type name, attr name, read, write)	Create a custom list. <i>read</i> and <i>write</i> must be an AccessPrivilege (e.g. AccessPrivilege.ADMIN).
typeDel(type name)	Delete group type
typeDelField(type name, field name)	Delete custom field from group type
typeFind(type name)	Find the group
typeGetFields(type name)	Get fields associated with the group type

Member change subject

[Change subject of a Member object](#), e.g.:

```
grouperSession = GrouperSession.startRootSession();
oldSubject = findSubject("10021368");
member = MemberFinder.findBySubject(grouperSession, oldSubject);
newSubject = findSubject("10021366");
member.changeSubject(newSubject);
```

Command	Description
member.changeSubject(newSubject);	Change the subject of the member object. If the subject is the same, its a no-op. If the new subject does not have a Member object, then the existing member object simply gets new subject information. If the new subject does have a member object, then all objects in the grouper registry which uses the old member, will be updated to the new member. Then the old member object is deleted from the registry
member.changeSubject(newSubject,! Member.DELETE_OLD_MEMBER);	Change the subject, but dont delete the old member. Do this if the way which deletes the old member doesnt work due to foreign keys. This will do all the work it can, and the rest can be manual
member.changeSubjectReport(newSubject,Member.DELETE_OLD_MEMBER);	Don't do any work, just print a report to the screen of what will be done. Dry-run.

Memberships

Command	Description
addComposite(group name, composite type, left group name, right group name)	Add composite membership. e.g. CompositeType.UNION
addMember(group name, subject id)	Add member to the members list for the group.
addMember(group name, subject id, field)	Add member to the specified list for the group.
delComposite(group name)	Delete composite membership from group

delMember(group name, subject id)	Delete member from the members list for the group
delMember(group name, subject id, field)	Delete member from the specified list for the group
getMembers(group name)	Get members of group
hasMember(group name, subject id)	Check whether subject is member of the members list
hasMember(group name, subject id, field)	Check whether subject is member of the specified list

Privileges

Command	Description
grantPriv(group name, subject id, privilege)	Grant privilege on group. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>)
grantPriv(stem name, subject id, privilege)	Grant privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>)
hasPriv(group name, subject id, privilege)	Check whether subject has privilege on group. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>)
hasPriv(stem name, subject id, privilege)	Check whether subject has privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>)
revokePriv(group name, subject id, privilege)	Revoke privilege on group. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>)
revokePriv(stem name, subject id, privilege)	Revoke privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>)

Registry

Command	Description
registryInitializeSchema()	Will generate schema DDL for the DB, and wont drop before creating, will not run script
registryInitializeSchema(registryInitializeSchema.DROP_BEFORE_CREATE)	Generate DDL for the DB, dropping existing tables, will not run script
registryInitializeSchema.WRITE_AND_RUN_SCRIPT)	generate DDL for the DB, not dropping, but will run the script after writing it to file
registryInitializeSchema(registryInitializeSchema.DROP_BEFORE_CREATE registryInitializeSchema.WRITE_AND_RUN_SCRIPT)	Generate DDL for the DB, drop existing grouper tables, and run the script after writing it to file
resetRegistry()	Restore registry to default state(delete data from all tables, install defaults)

registryInstall()	If the default Grouper data is not there, it will be added (e.g. root stem, default fields, etc)
-------------------	--

Stems

Command	Description
addRootStem(extension, displayExtension)	Add top-level stem to the registry
addStem(parent stem name, extension, displayExtension)	Add stem to registry
delStem(stem name)	Delete stem from registry
getStemAttr(stem name, attr)	Get value of stem attribute
getStems(name)	Find all stems with a matching naming attribute value, returns a Set of stems
setStemAttr(stem name, attr, value)	Set value of stem attribute
StemFinder.findByName(grouperSession, name)	Find one stem by name
StemFinder.findById(grouperSession, uuid)	Find one stem by uuid
	Delete stem and subcontents
<pre> grouperSession = GrouperSession.startRootSession(); stem = StemFinder.findByName(grouperSession, "a"); for(child : stem.getChildGroups(Stem.Scope.SUB)) { System.out.println("deleting: " + child.getName()); child.delete(); } stemList = new ArrayList(stem.getChildStems(Stem.Scope.SUB)); Collections.sort(stemList); Collections.reverse(stemList); for(childStem : stemList) { System.out.println("deleting: " + childStem.getName()); childStem.delete(); } stem.delete(); </pre>	

Subjects

Command	Description
addSubject(id, type, name)	Add local subject to registry
findSubject(id)	Find a subject
findSubject(id, type)	Find a subject
findSubject(id, type, source)	Find a subject

getSources()	Find all Subject sources
--------------	--------------------------

System

Command	Description
sqlRun(file)	Execute each line of a sql file, just like ant would. This can run the files generated by registryInitializeSchema()
sqlRun(string)	Executes a single sql statement
exit	Terminate shell
help()	Display usage information
history()	Print commands that have been run
history(N)	Print the last N commands that have been run
last()	Run the last command executed
last(N)	Execute command number N
p(command)	Pretty print results. This command is more useful when GSH_DEVEL is enabled
quit	Terminate shell
version()	Return version information

Unresolvable subject deletion utility (USDU)

usdu finds which memberships are with subjects which cannot be found in a subject source, and prints them on the screen

- if the usdu.DELETE option is passed in, then the memberships will be deleted
- a grouper session must be open when this command is run.

For more information, see [Unresolvable Subject Deletion Utility \(USDU\)](#)

Command	Description
subject=SubjectFinder.findById("GrouperSystem") session=GrouperSession.start(subject) usdu()	Sample call to find all unresolvable subjects in the registry and print details to the screen
usdu(usdu.DELETE)	Pass in that you want to delete memberships in the usdu call
usduBySource("schoolperson")	Work only in a specific subject source, pass in the sourceId from sources.xml
usduBySource("schoolperson", usdu.DELETE)	Work in a specific source and delete memberships
subject=SubjectFinder.findById("GrouperSystem") session=GrouperSession.start(subject) memberSubject=SubjectFinder.findById("1234567") member=MemberFinder.findBySubject(session,memberSubject) usduByMember(member)	Work only with a specific member

usduByMember(member, usdu.DELETE)	usdu by member, and delete memberships
-----------------------------------	--

Find bad memberships

This command will find membership records in the database which are invalid, and prints them on the screen, along with a GSH script that will fix the memberships.

For more information, see [Bad Membership Finder Utility](#)

Command	Description
findBadMemberships()	complete findBadMemberships run
subject=SubjectFinder.findById("GrouperSystem") session=GrouperSession.start(subject) stem = StemFinder.findByName(session, "test") findBadMembershipsByStem(stem)	find bad naming privileges for a specific stem
subject=SubjectFinder.findById("GrouperSystem") session=GrouperSession.start(subject) group = GroupFinder.findByName(session, "test:testGroup") findBadMembershipsByGroup(group)	find bad memberships and access privileges for a specific group

XML legacy

Command	Description
xmlFromFile(filename)	Load registry from XML in file
xmlFromString(xml)	Load registry from XML in string
xmlFromURL(url)	Load registry from XML at URL
xmlToFile(filename)	Exports registry to file
xmlToString()	Exports registry to string.
xmlUpdateFromFile(filename)	Update registry from XML in file
xmlUpdateFromString(xml)	Update registry from XML in string
xmlUpdateFromURL(url)	Update registry from XML at URL

XML export

There is an object: XmlExport which has various chaining methods, which should be ended with an exportTo() method. You can export to file or string.

For more information, see [Import-Export](#)

Command	Description
XmlExport xmlExport.stem(stem)	The stem to export. Defaults to the ROOT stem.
XmlExport xmlExport.group(group)	The group to export

XmlExport xmlExport.relative(boolean)	If group or stem specified do not export parent Stems.
XmlExport xmlExport.includeParent(boolean)	If group specified, export from the parent stem
XmlExport xmlExport.childrenOnly(boolean)	If stem specified, export child stems and groups only - not the specified stem
XmlExport xmlExport.userProperties(file)	Properties file for extra settings for import
XmlExport xmlExport.grouperSession(grouperSession)	Operate within a certain grouper session (defaults to root session)
void xmlExport.exportToFile(file)	Export to an XML file
void xmlExport.exportToString(string)	Export to an XML string

Examples:

```
gsh 1% new XmlExport().exportToFile(new File("c:/temp/export.xml"))

gsh 1% grouperSession = GrouperSession.start(SubjectFinder.findById("mchyzer"));
gsh 2% stem = StemFinder.findByName(grouperSession, "aStem");
gsh 3% new XmlExport().stem(stem).relative(true).userProperties(new File("C:/temp/
some.props")).grouperSession(grouperSession).exportToFile(new File("c:/temp/export.xml"));
```

-or- (without chaining)

```
gsh 3% xmlExport = new XmlExport();
gsh 4% xmlExport.stem(stem);
gsh 5% xmlExport.grouperSession(grouperSession);
gsh 6% xmlExport.exportToFile(new File("c:/temp/export.xml"))
```

XML import

There is an object: XmlImport which has various chaining methods, which should be ended with an importFrom() method. You can import from file, string, or url.

For more information, see [Import-Export](#)

Command	Description
XmlImport xmlImport.stem(stem)	The Stem into which data will be imported. Defaults to the ROOT stem.
XmlImport xmlImport.updateList(boolean)	XML contains a flat list of Stems or Groups which may be updated. Missing Stems and Groups are not created.
XmlImport xmlImport.userProperties(file)	Properties file for extra settings for import
XmlImport xmlImport.grouperSession(grouperSession)	Operate within a certain grouper session (defaults to root session)
XmlImport xmlImport.ignoreInternal(boolean)	Ignore internal attributes, including group and stem uuids.
void xmlImport.importFromFile(file)	Import from an XML file
void xmlImport.importFromString(string)	Import from an XML string
void xmlImport.importFromUrl(url)	Import XML from a URL

Examples:

```
gsh 1% new XmlImport().importFromFile(new File("c:/temp/export.xml"))

gsh 1% grouperSession = GrouperSession.start(SubjectFinder.findById("mchyzer"));
gsh 2% stem = StemFinder.findByName(grouperSession, "aStem");
gsh 3% new XmlImport().stem(stem).updateList(true).userProperties(new File("C:/temp/
some.props")).grouperSession(grouperSession).importFromUrl(new URL("http://whatever.xml"));
```

-or- (without chaining)

```
gsh 3% xmlImport = new XmlImport();
gsh 4% xmlImport.stem(stem);
gsh 5% xmlImport.grouperSession(grouperSession);
gsh 6% xmlImport.importFromFile(new File("c:/temp/export.xml"))
```

Transactions

Transactions facilitate all commands succeeding or failing together, and perhaps some level of repeatable reads of the DB (depending on the DB). If there is an open transaction and an exception is thrown in a command, GSH will shut down so that subsequent commands will not execute outside of a transaction.

Command	Description
help("transaction")	print help information
transactionStatus()	print the list of nested transactions
transactionStart("<GrouperTransactionType>")	start a transaction, or make sure one is already started Can use: "READONLY_OR_USE_EXISTING", "NONE", "READONLY_NEW", "READ_WRITE_OR_USE_EXISTING", "READ_WRITE_NEW"
transactionCommit("<GrouperCommitType>")	commit a transaction Can use: "COMMIT_NOW", "COMMIT_IF_NEW_TRANSACTION"
transactionRollback("<GrouperRollbackType>")	rollback a transaction Can use: "ROLLBACK_NOW", "ROLLBACK_IF_NEW_TRANSACTION"
transactionEnd()	end a transaction Note if it was read/write, and not committed or rolled back, this will commit and end

Loader

Above, it describes how you can kick off the loader in daemon mode. You can also execute one job with:

Command	Description
grouperSession = GrouperSession.startRootSession(); loaderGroup = GroupFinder.findByName(grouperSession, "stem:group"); loaderRunOneJob(loaderGroup);	Kick off the loader for one group (configured by group attributes)
loaderRunOneJob("MAINTENANCE_cleanLogs");	Kick off the loader by job name

GrouperShell Variables

gsh has several variables that can be set to modify runtime behavior

Variable	Description
GSH_DEBUG	Stack traces will be printed upon failure if true
GSH_DEVEL	Summaries of returned objects are not automatically printed if true
GSH_TIMER	Prints time spent evaluating each command if true

Example:

```
gsh 4% GSH_DEVEL = true
gsh 5% subj = findSubject("SD00125")
gsh 6% sess = GrouperSession.start(subj)
gsh 7% member = MemberFinder.findBySubject(sess, subj)
gsh 8% p(member.getGroups())
group: name='etc:sysadmingroup' displayName='Grouper Administration:SysAdmin Group' uuid='6f77fb36-
b466-481a-84a7-7af609f1ad09'
```

Note: you cannot encrypt passwords with GSH since the passwords end up in the GSH history. To encrypt passwords, issue the command:

```
C:\mchzyer\isc\dev\grouper-qs-1.2.0\grouper>java -jar lib\morphString.jar
Enter the location of morphString.properties: conf/morphString.properties
Type the string to encrypt (note: pasting might echo it back):
The encrypted string is: ca8a15be4ad0fb45c6f1b3ca0cfd9c9e
```

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

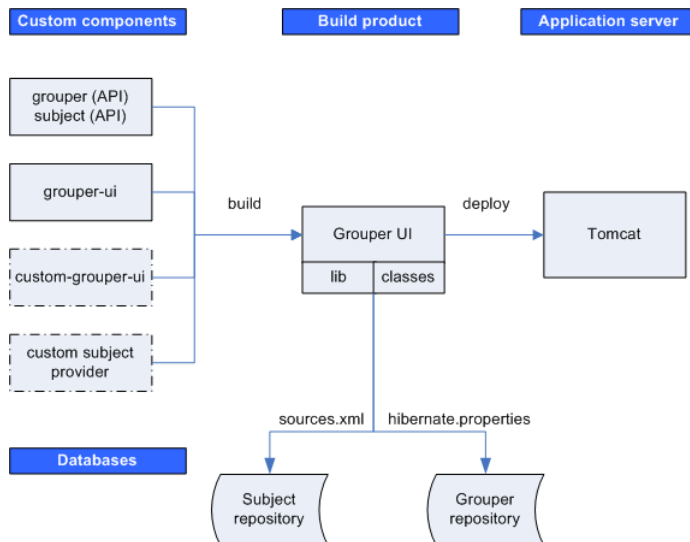
Grouper UI Components

This page last changed on Dec 06, 2007 by jbibbee@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Overview

This document is current as of the v1.2.0 release.



Custom Components

grouper	The Grouper API distribution includes the binary files for the Subject API implementation, which includes a JDBC provider.
grouper-ui	Source code for the Grouper UI is maintained as a separate module in the Internet2 Middleware CVS repository.
custom-grouper-ui <i>(optional)</i>	Sites implementing Grouper will generally want to re-brand (and perhaps heavily customise) the native Grouper UI (see Customising the Grouper UI).
custom subject provider <i>(optional)</i>	Depending on the identity management / person repository(s) in use, a site may also need to implement a custom Subject provider

Build Product

The Grouper UI build script compiles and combines the custom components in order to create a single web application build product. This step is responsible for ensuring that all required libraries (JAR files) and configuration files are available on the web application class path, i.e., in the *lib* or *classes* directories.

Application Server

Once built, the web application is then deployed to an application server. Currently, only Tomcat has been tested.

The Grouper UI does not require any container specific configuration to work.

Databases

Grouper requires a relational database. The default is HSQLDB, however, in principle, any database for which there is a JDBC driver and for which is supported by Hibernate can be used.

The Subject API can be configured to work with multiple sources (through sources.xml). A JDBC provider is provided with the Subject API distribution (an LDAP provider will be made available in the future), however, sites can implement their own providers.

Each time the Grouper UI is built, the sources.xml and hibernate.properties file are copied from grouper/conf to the web application classes directory. Database components can, through these files, be configured to be on the same machine or separate machines.

The Grouper QuickStart Distribution

The default Grouper [QuickStart](#) configuration uses the same HSQLDB database as a Grouper repository and as a source for the JDBC provider - the only source configured. In addition, Tomcat and the HSQLDB database run on the same machine.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper UI Development Environment

This page last changed on Dec 06, 2007 by jbibbee@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper UI Development Environment

Introduction

This document is current as of release v1.2.0.

There are many ways to set up a development environment using a variety of open source and commercial tools and application servers. The environment described here is the one used to develop the Grouper UI - however, you are free to use whatever setup works best for you.

I am an active developer, so the directory layout and build scripts I use are designed to facilitate development as well as final deployment. We normally use Eclipse as a Java IDE, and so some choices I made are biased to *cope* with the way Eclipse works. - Gary Brown, UoB

Directory structure

In order to verify the extensibility of the UI, I have developed the Grouper UI and a custom (University of Bristol) version in parallel, using the same environment. A minimal implementation only requires a Grouper API installation in addition to a Grouper UI installation, however, any real world implementation will have site specific components as well:

Grouper UI Development at the University of Bristol, UK

Component	Description
grouper	The Grouper API
grouper-ui	The Grouper UI
uob-grouper-ui	Bristol customisations to the Grouper UI
i2mi-subject	Bristol implementation of the Subject interface. Sites may be able to use a generic source adapter provided with The Subject interface distribution e.g. an LDAP adapter

grouper and grouper-ui are separate modules in the I2MI CVS repository.

uob-grouper-ui and i2mi-subject are separate modules in the CVS repository at Bristol.

I have all the CVS checkout directories as subdirectories at the same level (to help Eclipse), though this is not an absolute requirement, i.e., :

```
GrouperComplete
  grouper
  grouper-ui*
  uob-grouper-ui*
  i2mi-subject
```

*Both directories contain a subdirectory *webapp* which itself has a directory structure that is consistent with a web application (see Architecture document)..

During development I may need to debug source code from any of the projects. I may also want to make code changes in the appropriate CVS checkout areas*. In my ideal development environment I would be able to edit any source files and instantly see changes in the web application. A typical build script for a web application might create the web application directory structure in a new *build* directory and then either copy or make into a WAR (web application archive file), and then deploy to a Servlet container e.g. Tomcat. Using this approach every change requires a build and potential restart of the web application. Admittedly Eclipse will allow an ant script to be called when source files are modified, however, this can be overkill for a simple change to a JSP.

*I could edit JSP and other files in the *build* or *deploy* directory, however, I would then need to copy the changes back to CVS - something I may well forget to do.

Setting Up Eclipse

In Eclipse I create one *project* and pull in the *java/src* and *lib* directories from each of the 4 projects listed above. I can then set a single output directory where compiled Java classes are placed whenever I save a Java source file. JSP and other *content* files are trickier since they are saved *in situ* and not compiled to a separate destination. Normally I will be working on *either* the core Grouper UI *or* on Bristol customisations. In the Grouper UI *build.properties* file I can elect to have the *webapp* directory of grouper-ui *or* uob-grouper-ui be the web application root (configured in Tomcat)*. I manually configure Eclipse to compile Java classes to the appropriate *webapp/WEB-INF/classes* directory.

*Actually, any directory can be configured to be the web application root - I always choose either of the ones indicated when developing.

Any changes I make to the *local* JSPs are immediately picked up by Tomcat, however, I would need to run an ant script to obtain changes from the other project i.e. *uob-grouper-ui* to *grouper-ui*. Most sites which are not involved in the development of the Grouper UI should set *<institution>-grouper-ui/webapp* to be their web application root. If working with *Tomcat* and the *build.properties deploy* properties are set, the build script will automatically install your webapp on Tomcat such that Tomcat reads files from your *work area*.

A disadvantage of this approach is that it *pollutes* the CVS checkout area for one module with those from another, and I may be tempted to edit a file in the wrong location (though hopefully they are in different subdirectories). Assuming that site-specific changes are always in distinct subdirectories then on Unix it may well be possible to set up symbolic links from grouper-ui to, say, uob-grouper-ui.

Some changes e.g. adding new JAR files, modifying resources, changing Struts / tiles configuration files will always require a build and a web application restart.

The Ant Script

The following targets are available:

```
>ant help
```

```
Buildfile: build.xml help:
```

```
[echo] Please ensure you have read the documentation -
[echo] and created a build.properties file based on the template provided
[echo] The following targets are available - type the appropriate name:
[echo] 1) default
[echo]    Simply builds, without cleaning, to the default.webapp.folder
[echo] 2) nice
[echo]    Attempts to stop the Tomcat webapp before building.
[echo]    Attempts to start the webapp afterwards
[echo] 3) clean
[echo]    Always removes the webapp.class.folder. May remove the
[echo]    webapp.folder if webapp.folder.cleanable=true
[echo]    On Windows this may fail as Windows tends to lock files
[echo] 4) niceclean
[echo]    Combination of nice and clean
```

```
[echo] 5) dist
[echo]    Cleans and then builds to subfolder of dist.home
[echo] 6) war
[echo]    Does dist and then makes a WAR file
[echo] 7) resources
[echo]    Does not compile Java classes but 'refreshes' resources in
[echo]    webapp.class.folder
[echo] 8) niceres
[echo]    Does not compile Java classes but 'refreshes' resources in
[echo]    webapp.class.folder and restarts webapp
[echo] 9) help
[echo]    Displays this menu
[echo] 10) endhelp
[echo]    Subsequent invocation of ant with no target will run
[echo]    'default' rather than help
[echo] 11) starthelp
[echo]    Subsequent invocation of ant with no target will run 'help'
[echo] 12) html
[echo]    Generate Javadoc - you must have done a 'default' build previously
[echo] 13) exit
[echo]    Exit this menu without executing another target
[input] Make your choice (default)>
```

The *nice* targets will only work if you are using Tomcat and have configured the deploy properties in build.properties, and have installed catalina-ant.jar with Ant.

See Customising the Grouper UI: [Customising the Build Process](#) for details on how to customise the build process.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Web Services

This page last changed on Mar 25, 2009 by [mchyzer](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Web Services as of v1.4.0

Introduction

Grouper web services (grouper-ws) is a J2EE web application which exposes common Grouper business logic through SOAP and REST. See [FAQ](#).

To deploy the services, download the warfile and configure the property files (e.g. subject sources, databases, logging, etc). Configure [authentication](#).

Note: there is a command line and java API web service client called [Grouper Client](#)

To implement a web service client:

1. Understand the object model. All grouper-ws services are operations based on simple data structures. The structures support Strings, ints, arrays, and structure references.
 - a. [Core web service API](#)
 - b. [Example structure](#) (only "getters" and "setters" are applicable properties)
 - c. Each operation has many samples (authmated captures, versioned, and up to date). [Here is an example](#)
 - d. Most options has a sensible default (e.g. MemberFilter defaults to All members)
 - e. Lookup objects in various (consistent) ways. e.g. to delete a group, you can pass the name or uuid of the group.
2. Decide if you are using SOAP or REST (this is real REST, not Axis HTTP/XML)
 - a. Both SOAP and REST support the same API
3. Inside SOAP and REST, each operation has two levels of complexity, the normal one, and the Lite one.
 - a. Normal operation: can usually be batched (support a list of inputs, e.g. add multiple groups at once), supports complex inputs (arrays or structures)
 - b. Lite operation: supports only inputs of scalars (no structures, no arrays... only Strings, ints, etc). In REST this also means that the request can be sent via query string only
4. If SOAP:
 - a. Implement based on the [WSDL](#)
 - b. There is a [sample Java client](#) with [sample calls](#)
5. If REST:
 - a. Decide what format you want to send and receive data. grouper-ws supports [XHTML](#), [XML](#), and [JSON](#), as well as query strings for input (in URL or message body)
 - b. There are many [samples](#)

[Presentation about Grouper web services](#)

[.NET client development guide](#)

[PHP client development guide](#)

Guidelines For Working With Grouper Web Services

1. There is a bug we are tracking with Axis, where if you skip String params, it will mix up the params. So, if you are passing a param to a web service, make sure you pass empty strings for all null params before the param
2. Code clients with a mindset that the service might change in subtle ways. e.g. a result code might be added (check for success flag element, not success result code), an element might be added in a result object, another input element might be added to end of list, etc. Expect elements to be added in data
3. Make sure there is a property in the client of the URL and version for the service. The version of the service might change the URL (up to service deployer)...

Operations

- [addMember](#): assign a member to a group
 - If already a member, that is ok
 - Accepts batches of members (non-Lite)
 - Accepts flag to say that any members not in batch should be removed (e.g. replace list)
- [deleteMember](#): unassign a member from a group
 - If not a member, that is ok
 - Accepts batches of members (non-Lite)
- [getMembers](#): return the members (including subject data) in a group (from direct or indirect membership)
 - Will accept member filter (All, Effective, Immediate, Composite)
 - Accepts batches of groups (non-Lite)
- [getMemberships](#): under construction
 - Will accept member filter (All, Effective, Immediate, Composite)
 - Accepts batches of subjects and groups (non-Lite)
- [hasMember](#): see if a subject is a member of a group
 - Will return true or false
 - Accepts batches of subject ids or identifiers (returns batches of true's / false's) (non-Lite)
 - Will accept member filter (All, Effective, etc)
 - Can query on field (permission)
- [getGroups](#): list groups for a subject
 - Will accept member filter (All, Effective, etc)
 - Accepts batches of subjects (non-Lite)
- [groupSave](#)
 - Create / update a group
 - Accepts batches of groups (non-Lite)
- [groupDelete](#)
 - Delete a group
 - Accepts batches of groups (non-Lite)
- [getGrouperPrivileges](#)
 - View privileges (many combinations of input are acceptable)
 - Can view all privileges for the subject, group, stem, specific privilege and combinations thereof
- [assignGrouperPrivileges](#)
 - Add or remove a privilege for a subject and (group or stem)
 - Will not fail if the privilege is already assigned or revoked
- [findGroups](#)
 - Can query for groups based on name, uuid, parent stem, or a substring query
 - Can create complex queries with group match (AND, OR, MINUS) (non-Lite)
- [findStems](#)
 - Can query for stems based on name, uuid, parent stem, or a substring query
 - Can create complex queries with group match (AND, OR, MINUS) (non-Lite)
- [stemSave](#)
 - Create / update a stem
 - Accepts batches of stems (non-Lite)
- [stemDelete](#)
 - Delete a stem
 - Accepts batches (non-Lite)
- [memberChangeSubject](#)
 - Change the subject of a current member
 - Accepts batches (non-Lite)

Features

- **API**
 - Batched operations (e.g. add 100 subjects to a group at once). There is a separate server-side max-in-batch param in the grouper-ws.properties.
 - Transaction support (if any fails in one batch request, rollback all in that single batch request)
- **Authentication**
 - Let container or web server handle
 - PKI
 - http-simple-auth
 - Source IP address filtering (TODO)
 - Custom authenticator

- WS-Security
 - PKI
 - Kerberos
- Proxying. The web service can execute operations based on an underlying user, not the authenticating user. Note the authenticating user must have appropriate permissions
- **Error Handling**
 - Error codes and error messages are sent in responses, as well as warnings. In batched mode, batches of response codes are returned. In REST, the http status code is used as well.
- **Clients**
 - Grouper will provide a quick start with Java, and it is up to users to create their own clients. The SOAP and REST are based on the HTTP documents, so any programming language will work
- **Web Service Implementation**
 - Apache Axis for SOAP, and home-grown for REST

Quick start

Checkout the appropriate projects under [grouper-ws](#), read the [README.txt](#) in the grouper-ws/grouper-ws cvs directory

Build Script

The build script for grouper-ws is pretty basic. Generally just do the default (dist). There is also an "ant grouper" target to build a new grouper jar, and "ant quick" to do everything but generate the Axis files (takes 3 minutes).

```
C:\mchyzer\isc\dev\grouper\grouper-ws>ant
Buildfile: build.xml
```

```
dist:
```

```
clean:
```

```
[delete] Deleting directory C:\mchyzer\isc\dev\grouper\grouper-ws\build
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\grouper-ws
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws
```

```
compile:
```

```
[javac] Compiling 10 source files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\grouper-ws
[javac] C:\mchyzer\isc\dev\grouper\grouper-ws\src\grouper-ws\edu\internet2\middleware\grouper
\websservices\GrouperSer
viceServlet.java:33: warning: [deprecation] getEPRForService(java.lang.String,java.lang.String) in
org.apache.axis2.trans
sport.TransportListener has been deprecated
[javac] public class GrouperServiceServlet extends AxisServlet {
[javac]      ^
[javac] 1 warning
```

```
generate-aar:
```

```
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[delete] Deleting directory C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[copy] Copying 13 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\webapp\WEB-INF\services\GrouperService.aar
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws.jar
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\classes
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\lib
[copy] Copying 12 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\classes
[copy] Copying 30 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\lib
[copy] Copying 11 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws.war
```

```
BUILD SUCCESSFUL
```

```
Total time: 22 seconds
```

```
The system cannot find the batch label specified - end
```



```
C:\mchyzer\isc\dev\grouper\grouper-ws>
```

Notice the generate-aar target. This is what makes the axis archive, which is all the classes needed for axis to determine the wsdl, along with the services.xml config file.

Axis is ~40 jars, though most of them are pretty axis specific. There is an ant target which will compress most of these into one jar (axisBundle.jar). Here is the ant help:

```
C:\mchyzer\isc\dev\grouper\grouper-ws>ant help
Buildfile: build.xml
```

```
help:
```

```
[echo] Please ensure you have read the documentation -
[echo] and created a build.properties file based on the template provided
[echo]
```

```
[echo] The following targets are available - type the appropriate name:
```

```
[echo]
```

```
[echo] 1) default (dist)
```

```
[echo]     Simply builds, without cleaning, to the webapp.folder
```

```
[echo] 2) clean
```

```
[echo]     Clean the webapp folder, and classfiles, and build
```

```
[echo] 3) generate-aar
```

```
[echo]     Make the axis archive, which is the classfiles and services.xml that axis needs. You need to do
this i
```

```
f you ever change anything that changes the wsdl. You can do this automatically in dist by setting a property
in the bu
```

```
ild.properties
```

```
[echo] 4) generate-axis-bundle-jar
```

```
[echo]     Take all the bundlable axis jars (in lib/axis-bundle), unjar, and jar back up into one jar
```

```
[echo]
```

```
BUILD SUCCESSFUL
```

```
Total time: 0 seconds
```

```
The system cannot find the batch label specified - end
```

```
C:\mchyzer\isc\dev\grouper\grouper-ws>
```

Subject attributes

1. The default attribute names (comma separated) sent back for each request are specified in grouper-ws.properties under the key:

```
ws.subject.result.attribute.names
```

2. If the caller sets T to retrieve subject detail, then the attributes will be appended to that list in grouper-ws.properties key:

```
ws.subject.result.detail.attribute.names
```

3. If the caller specifies subjectAttributeNames in the request (comma separated), then those will be appended to the list (independent of the detail attributes).

So there are central settings, and caller settings that you need to design for and specify...

Fields and permissions

If you want to check to see if a subject as a group permission, or to get a list of people with a certain permissions on a group, use hasMember or getMembers, and pass the name of the field (note this list depends on your configuration):

```
select name from grouper_fields where type != 'naming';
```

```
admins
```


```
description
```

```
displayExtension
```

displayName
extension
members
name
optins
optouts
readers
requireActiveEmployee
requireAlsoInGroups
updaters
viewers

To do's (post 1.3.0)


1. add in get members / has members for stems. look into returning multiple lists at once
2. investigate backwards compatibility with Axis... discuss options
3. add find subject service
4. test more
5. unit test
6. build a client jar back into web services to unit test
7. make some params to test stuff... (junit to throw exceptions in the middle of tx?)
8. come up with formatter and code style and remove all warnings
9. add logging filter
10. fix javadoc warnings
11. look into axis 1.4, see if error fixed, see if samples/wsdl changes, see about enums
12. add move subject service
13. add metadata service
14. add getGroups with batched groupLookup input
15. add batched privilege service, and add more url options to REST
16. add back in memberships service
17. filter getMember by privileges (find member?)
18. in rest add GET starting points with links to resources
19. improve auto-toString methods in resultMessage
20. look at acegi
21. add ip source filtering to grouper

 [Contact us](#) if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Authentication for Grouper Web Services

This page last changed on Sep 02, 2008 by sanjay.vivek@ncl.ac.uk.

 [Contact us](#) if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Authentication for Grouper Web Services as of v1.3.0

Default authentication

Out of the box, grouper-ws uses container authentication (non-rampart). The web.xml protects all services and expects the users to be in the role "grouper_user". For tomcat, in the tomcat-users.xml, just have entries like this, and you are all set:

```
<role rolename="grouper_user"/>
<user username="jota" password="whatever" roles="grouper_user"/>
<user username="jobr" password="whatever" roles="grouper_user"/>
<user username="eldo" password="whatever" roles="grouper_user"/>
```

Note that users to the web service need to be Subjects, and you can configure the default source in the grouper-ws.properties especially if you have subjectId overlap in various sources.

Note the default authentication in grouper-ws is http-basic, so for this and other reasons make sure your deployments of grouper-ws are protected with SSL.

Note that, for some container technologies, container authentication can be externalized in various ways. A common deployment configuration is to externalize tomcat authentication to Apache 2.2+ using the AJP protocol. This permits several popular authentication technologies to be used in conjunction with grouper-ws.

Custom authentication plugin

If you want custom authentication (e.g. pass in a token, and decode it), then implement the interface `edu.internet2.middleware.grouper.ws.security.WsCustomAuthentication` and configure your fully qualified classname in the grouper-ws.properties. The default is an implementation of this interface as an example: `edu.internet2.middleware.grouper.ws.security.WsGrouperDefaultAuthentication`, which just gets the user from the container: `HttpServletRequest.getUserPrincipal().getName()`

Rampart

Rampart is Jakarta's WS-Security implementation. We have vanilla [Rampart authentication](#) working with grouper-ws (thanks to Sanjay Vivek). Unfortunately it doesn't work out of the box since it seems Rampart and basic auth cannot work together in the web app. If you want to run basic auth and rampart at the same time, you should deploy two separate web apps.

Note the URL for rampart in grouper-ws is the same, it will look like this: `/grouper-ws/services/GrouperService`

Also, for Rampart, you need custom logic to authenticate users. To use rampart, configure the grouper-ws.properties entry: `ws.security.rampart.authentication.class`. An example is: `edu.internet2.middleware.grouper.ws.security.GrouperWssecSample`. Until you configure that, clients will get a 404 http status code. This assumes you are using `WSPasswordCallback`, if not, just provide your own class directly to the services.xml file (and grouper-ws requires you have an implementation of the interface anyway which won't be executed).

You need to tell grouper that wssec is enabled in the web.xml servlet param (uncomment):

```
<servlet>
  <servlet-name>AxisServlet</servlet-name>
  <display-name>Apache-Axis Servlet</display-name>
```

```

        <servlet-class>edu.internet2.middleware.grouper.ws.GrouperServiceAxisServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
<!-- hint that this is the wssec servlet -->
<init-param>
    <param-name>wssec</param-name>
    <param-value>true</param-value>
</init-param>
</servlet>

```

Also you need to comment out the container auth in web.xml:

```

<!-- security-constraint>
    <web-resource-collection>
        <web-resource-name>Web services</web-resource-name>
        <url-pattern>/services/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>grouper_user</role-name>
    </auth-constraint>
</security-constraint -->

```

Then you need to enable the correct .aar file.

- If you are using a binary grouper-ws.war, just rename the following two files
 - /WEB-INF/services/GrouperService.aar to /WEB-INF/services/GrouperService.aar.ondeck
 - /WEB-INF/services/GrouperServiceWssec.aar.ondeck to /WEB-INF/services/GrouperServiceWssec.aar
- If you are building, just set the param in the build.properties: webapp.authentication.use.rampart
Here is a sample client

HTTP basic authentication (use)

In the web.xml for the grouper-ws project, protect all services:

```

<security-constraint>
    <web-resource-collection>
        <web-resource-name>Web services</web-resource-name>
        <url-pattern>/services/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <!-- NOTE: This role is not present in the default users file -->
        <role-name>grouper_user</role-name>
    </auth-constraint>
</security-constraint>

<!-- Define the Login Configuration for this Application -->
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>Grouper Application</realm-name>
</login-config>

<!-- Security roles referenced by this web application -->
<security-role>
    <description>
        The role that is required to log in to the Manager
        Application
    </description>
    <role-name>grouper_user</role-name>
</security-role>
</web-app>

```

Now send the user and pass in the web service client.
Here is an example with Axis generated clients:

```

GrouperServiceStub stub = new GrouperServiceStub(
    "http://localhost:8090/grouper-ws/services/GrouperService");

```

```
Options options = stub._getServiceClient().getOptions();
HttpTransportProperties.Authenticator auth = new HttpTransportProperties.Authenticator();
auth.setUsername("user");
auth.setPassword("pass");

options.setProperty(HTTPConstants.AUTHENTICATE, auth);
```

Here is an example with a manual HttpClient:

```
HttpClient httpClient = new HttpClient();
GetMethod getMethod = new GetMethod(
    "http://localhost:8091/grouper-ws/services/GrouperService/addMemberSimple?
groupName=aStem:aGroup&subjectId=10021368&actAsSubjectId=GrouperSystem");

httpClient.getParams().setAuthenticationPreemptive(true);
Credentials defaultcreds = new UsernamePasswordCredentials("user", "pass");
httpClient.getState().setCredentials(new AuthScope("localhost", 8091), defaultcreds);

httpClient.executeMethod(getMethod);
```

ActAs configuration

To enable web service users to act as another user (proxy), enable the setting in the grouper-ws grouper.properties

```
# Web service users who are in the following group can use the actAs field to act as someone else
ws.act.as.group = aStem:aGroup
```

If you specify a group name in there, you can pass in the actAs field if you connect to the web service as a user who is in the ws.act.as.group group. Here is an example with the axis generated client.

```
//set the act as id
WsSubjectLookup actAsSubject = WsSubjectLookup.class.newInstance();
actAsSubject.setSubjectId("GrouperSystem");
addMember.setActAsSubjectLookup(actAsSubject);
```

There are advanced settings, you can specify multiple groups in the grouper-ws.properties, and you can even limit who the users can act as (in a specific group).

Additional Information

The G-FIV-O Project has produced a report that delves further into options for securing Grouper Web Services.

[GFIVO: Multiple Security Mechanisms Web Services Deployment](#)

Grouper Web Services FAQ

This page last changed on Apr 17, 2008 by tbarton@uchicago.edu.

- Can I run grouper web services against any version of grouper?

No, you should use the version of grouper bundled in the web services. This makes it a little complicated if you have a UI / GSH / etc running against your grouper DB. You will need to keep them in sync...

- Why are there three flavors of web services (SOAP, XML-HTTP, REST-Lite)?

SOAP is supported since many schools required it. REST-Lite is supported because many schools required it. XML-HTTP is supported because Axis2 gives it for free when building a SOAP web service. This way the same SOAP interface can be accessed by clients who only want to talk HTTP and XML and not SOAP.

- Why is the rampart .aar file named GrouperServiceWssec.aar, but the URL is still /services/GrouperService?

The URL for rampart or not is the same. Inside the services.xml in the .aar files, it configures the app name. grouper-ws will not work if you change this (unless to do other build activities also)

- Why is there a "simple" operation for every non simple operation in SOAP and XML-HTTP?

The non-simple operations are batchable if the client wants to do one operation multiple times with one request. The simple operations are for if the client only needs to do one thing (e.g. assign a member to a group and not assign multiple members to a group). Also, there is a valuable side effect that for the XML-HTTP, if the operation only has scalar input params (and not complex types or arrays), that the input does not need XML, it can be in the query string for GET or in the http form param pairs in the body for POST.

It is confusing that there are so many different ways to call grouper via web service. The documentation will be improved to make it easier to find the best strategy.

- Why element named "return" (by axis)

This is an unfortunate "feature" by axis, the default element is named "return" which is a keyword in many programming languages, so if the language automatically converts the xml to an object graph, then it will be broken. Chris will followup with Axis to see if there is a fix for this

- Why returning error codes and messages and not just use SOAP faults?

For two of three of the flavors of web services SOAP faults are not an option since they are not SOAP. Also, for SOAP batched, the status of each line item needs to be returned for the client to process, and a SOAP fault would preclude that. Also, the fewer SOAP specific features that are used, the more widespread the compatibility will be. All three flavors of web service use the same underlying logic, so the more consistent the better.

- Can we add a service for subject search?

Yes, this will be added

Import-Export

This page last changed on Jan 08, 2009 by tzeller@memphis.edu.

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

XML Import / Export for Grouper v1.4.0

As of v1.4.0 the invocation of these tools has moved from Ant to gsh (GrouperShell):

Grouper v1.2.0+ includes XML import / export tools. Exported XML may be used for:

- provisioning to other systems
- reporting
- backups
- switching database backends - including to upgraded schemas (required by new Grouper API versions) in the same database

Imported XML may be used for:

- loading - adding to or updating existing Stems, Groups and Group Types. Whole or partial Grouper registries can be exported, and subsequently imported at a specified Stem (or the Root Stem if not specified) in the *new* instance.*
- initializing a new, *empty* registry to a known state - useful for demos, testing and system recovery

In general, exported data can be imported into the same Grouper instance it was exported from**, or a different instance. Stems and Groups and Group Types will be created, if not already present, or updated if they already exist (depending on import options provided).

The XML formats for import and export are very similar, however, there are some differences.

The export format:

- defines what is actually exported,
- includes some meta data about the export,

while the import format:

- allows import options to be embedded in the XML,
- defines additional attributes for Stems and Groups which may affect the importing of Stems and Groups,
- does not require all of the information that is exported.

Any tool which can create XML, in the correct format, can be used as a loader.

*To successfully load Subject data, the new Grouper instance must be configured with the same Subject Sources. The export tool does not export Subject registries. Subjects which cannot be resolved will be logged, but otherwise ignored.

**The initial version of the import tool did not maintain system attributes i.e. uuid, date created etc. Since v1.3.0 system attributes are maintained by default, which is the desired behavior if migrating a registry, however, this can cause a problem if you want to copy part of the registry by exporting it and importing it into a new stem because the uuids of imported groups and stems already exist. v1.4.0 introduces a new command line argument *-ignoreInternal* (see below) which ensures that uuids and other internal attributes are ignored.

Export Tool in More Detail

A Java class, XmlExporter, provides the export functionality. It can be run from the command line, from within Java code, or using gsh:

```
bin/gsh.sh -xmlexport <command line arguments>
```

The command line usage is:

Command	Summary of args.
args: -h	Prints this message
args:	subjectIdentifier [(-id) [-name]] [-relative] [-childrenOnly] [-includeParent] fileName [properties]

The above export args. can be explained as follows:

Command	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Group or Stem to export. Defaults to the ROOT stem.
-name	The name of a Group or Stem to export. Defaults to the ROOT stem.
-relative	If id or name specified do not export parent Stems.
-includeParent	If id or name identifies a Group and -relative is selected, export the Group and its parent Stem.
-childrenOnly	If id or name identifies a Stem and -relative is selected export child Stems and Groups, but not the stem itself.
filename	The file where exported data will be written. Will overwrite existing files.
properties	The name of an optional Java properties file. Values specified in this properties file will override the default export behavior documented in the XmlExporter javadoc.

The JavaDoc describes the export methods, including a method which can be used to export an arbitrary Collection of Stems, Groups, Subjects or Memberships returned by various Grouper API methods. This means that the results of any *list* or *search* methods can be exported.

An XML Schema which describes the exported XML is available [here](#).

If a relative export is performed, the export tool treats group members, list members or privilegees which are groups, and which are *descendants* of the export stem in a special manner. The Subject Identifier, which, for groups, is usually the group name, is modified so that the export stem name is replaced by an asterix, thus, if performing a relative export of uob:artf, a reference to the staff group would become *staff rather than uob:artf:staff. The import tool will replace the asterix with the import stem name. In this way the relationship between groups can be maintained.

Examples of exported data are available [here](#).

Import Tool in More Detail

A Java class, XmlImporter, provides the import functionality. It can be run from the command line, from within Java code, or using gsh:

```
bin/gsh.sh -xmlimport <command line arguments>
```


The command line usage is:

Command	Summary of args.
args: -h	Prints this message
args:	subjectIdentifier [(-id -name -list)] [-ignoreInternal] [-noprompt] filename [properties]

The above import args. can be explained as follows:

Commands	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Stem into which data will be imported. Defaults to the ROOT stem.
-name	The name of a Stem into which data will be imported. Defaults to the ROOT stem.
-list	File contains a flat list of Stems or Groups which may be updated. Missing Stems and Groups are not created.
-ignoreInternal	Do not attempt to import internal attributes including Group/Stem uuids. Overrides property: <i>import.data.ignore-internal-attributes-and-uuids</i>
-noprompt	Do not prompt user to confirm the database that will be updated
filename	the file to import
properties	The name of an optional Java properties file. Values specified in this properties file will override the default import behavior documented in the XmlImporter javadoc.

The JavaDoc describes the load methods.

An XML Schema which describes the format of XML which can be loaded is available [here](#).

It is possible to generate an XML file which validates against the schema, but which does not load properly. The annotations in the schema describe appropriate usage of attributes and elements.

The Grouper QuickStart includes a demo registry. [quickstart.xml](#) is a minimal XML import file which creates the demo registry*.

When generating XML in the import format, it is likely that relationships between stems and groups will need to be specified. This is problematic because the uuids of groups and stems are unknown prior to creation. In addition, it is not always possible to know the full name of a new Stem or Group, as this will depend on which stem it is imported into. When importing Subjects that are groups, the import tool examines the identifier attribute and makes any necessary changes before further processing. The following notations are recognised:

Notation	Description
----------	-------------

SELF	Refers to the <i>context group</i> for which the Subject is being processed as a member*, list member or privilege. *Actually the API will prevent a Group becoming a member of itself
*	As described above, * is replaced with the name of the Stem where the XML is to be imported
..:	Replace with the name of the Stem which contains the context group.
...:	Replace with the parent Stem of the Stem which contains the context group. May occur multiple times.

Notes from the Field

Some of the example xml and the xsd referenced above are inconsistent with the v1.2.0+ implementation of the xml import/export tool. Here are some details you need to know to successfully load members into groups using the xml import method.

1. The <subject> element requires the 'immediate' attribute. Best practice is to fully reference each subject, giving its source, type, and declaring it to be an immediate membership. So, instead of

```
<list field="members"> <subject id="someId"/> </list>
```

use

```
<list field="members"> <subject id="someId" source='someSource' type='person' immediate='true' /> </list>
```

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Initializing Administration of Privileges

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Initializing Administration of Grouper Privileges as of v1.4.0

GrouperSystem is the root-like principal used to manage assignment of privileges in Grouper. In addition to GrouperSystem, externally authenticated members of the [wheel group](#) can choose when to act with root-like privileges.

If you've enabled the wheel group, you must create it and add members. GrouperShell acts as GrouperSystem and can bootstrap the necessary naming stem(s), group, and memberships.

The wheel group is available as of Grouper v1.0.

Enabling the Wheel Group

The wheel group is enabled and named in `conf/grouper.properties` :

`conf/grouper.properties`

```
# A wheel group allows you to enable non-GrouperSystem subjects to act
# like a root user when interacting with the registry.
groups.wheel.use                = true

# Set to the name of the group you want to treat as the wheel group.
# The members of this group will be treated as root-like users.
groups.wheel.group              = etc:sysadmingroup
```

Automatically Creating the Wheel Group

To automatically create the wheel group :

`conf/grouper.properties`

```
configuration.autocreate.system.groups = true
```

Using GrouperShell to Create the Wheel Group

To create the wheel group using GrouperShell :

GrouperShell

```
gsh 0% addRootStem("etc", "Grouper Administration")
stem: name='etc' displayName='Grouper Administration' uuid='f7687876-2c94-4635-997c-f2793fb8152d'
gsh 1% addGroup("etc", "sysadmingroup", "SysAdmin Group")
group: name='etc:sysadmingroup' displayName='Grouper Administration:SysAdmin Group' uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

Adding Members to the Wheel Group


Whether you've set the wheel group to be automatically created, or you've used GrouperShell to create it, you'll need to add members to the wheel group using GrouperShell :

GrouperShell

```
gsh 0% addMember("etc:sysadmingroup", "SD00125")
true
```

In this example "SD00125" is the subjectId of a person, as determined outside of gsh by, in this case, an LDAP query to a directory that acts as a subject source to Grouper:

```
% ldapsearch \-b dc=kitn,dc=edu uid=tbarton
dn: kitnEduPersonRegId=SD00125,ou=people,dc=kitn,dc=edu
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: kitnEduPerson
kitnEduPersonRegId: SD00125
cn: Barton, Tom
sn: Barton
description: Professor, Mathematics
uid: tbarton
```

 Questions or comments?  [Contact us.](#)

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

License

This page last changed on Sep 23, 2009 by steveo@internet2.edu.

This page has moved to <http://www.internet2.edu/grouper/license.html>. Please update your bookmarks and links.

Nav

This page last changed on Sep 23, 2009 by steveo@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Prerequisites

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Prerequisites as of v1.4.0

The Grouper API requires:

- [Java](#)
- database (e.g. MySQL, Oracle, PostgreSQL, etc.)

The Grouper UI and WS packages also require:

- [Ant](#)
- servlet container (e.g. [Tomcat](#))
- web server (e.g. [Apache](#))

The [Specsheet](#) provides additional details regarding requirement specifications.

The [Project Layout](#) describes the directory structure of Grouper packages.

Relational Database

Grouper uses Hibernate to persist objects in a relational database, called the **Groups Registry**. Hibernate in turn uses JDBC for database connectivity. The .jar file containing the JDBC driver for the RDBMS of your choice must be available during installation.

All of Grouper's access to the underlying database is by means of a single account. The username and password or other authentication token for this account must also be available during installation.

The Grouper distribution includes the free and open source HSQLDB relational database, which is used in conjunction with testing the compiled code.

Web Server

Although it is possible to run Grouper without a web server, it is likely needed for a production deployment. The web server will restrict access to the Grouper application, authenticate your users, optionally authenticate the special GrouperSystem account, and implement SSL.

If you will use [Apache v2.2+](#), configure apache to load mod_proxy and mod_proxy_ajp and include configuration directives similar to the following:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
ProxyRequests Off
```

Then, in tomcat's server.xml, ensure that the Connector element for port 8009 is uncommented and contains directives similar to these:

```
<Connector port="8009"
  enableLookups="false" redirectPort="8443" protocol="AJP/1.3"
  tomcatAuthentication="false"/>
```

If you will use [Apache v1.3 or v2.0.X](#), configure apache to load the [mod_jk connector](#). The following configuration directs Apache to use mod_jk to redirect queries for Grouper to Tomcat. This may be done by including the following text directly in httpd.conf, or making a separate file and including it in httpd.conf.

```
<IfModule !mod_jk.c>
  LoadModule jk_module libexec/mod_jk.so
</IfModule>
JkWorkersFile "/usr/local/tomcat/conf/jk/workers.properties"
JkLogFile "/usr/local/apache/logs/mod_jk.log"
JkLogLevel emerg JkMount /grouper/* ajp13
```

- Add address="127.0.0.1" to Tomcat's server.xml inside the <Ajp13Connector> configuration element to prevent off-host access.
 - For Tomcat 5.5 or newer, add request.tomcatAuthentication="false" to the <Ajp13Connector> configuration element in server.xml to ensure that the user's identity is passed from Apache to the servlet environment.
 - For Tomcat 5.0.x or older, add tomcatAuthentication="false" to the <Ajp13Connector> configuration element in server.xml to ensure that the user's identity is passed from Apache to the servlet environment.
 - Tomcat 4.1.x defaults to having the Coyote connector enabled in /conf/server.xml. This fails with mod_jk and must be commented out. Then, uncomment and modify the traditional AJP 1.3 connector as indicated above.
- The AJP13Connector for tomcat is not compatible with the new JMX support. To remove some warnings that will appear in the Tomcat log every time Tomcat is restarted, comment out all of the JMX stuff (anything that says "mbeans") from server.xml.

Apache-based User Authentication

The interaction between the Grouper UI and an Apache-based local authentication system is implemented by providing the UI with the identity of the browser user through REMOTE_USER. Any authentication system that is capable of protecting a block of webspace using httpd.conf and populating the REMOTE_USER header variable is compatible with Grouper. This associates the appropriate authentication mechanism with the URL of the Grouper servlet, ensuring users authenticate and that their login name is passed to Grouper. The following example demonstrates use of a very basic authentication method with the Grouper UI:

```
<Location /grouper>
  SSLRequireSSL
  AuthType Basic
  AuthName "ExampleU Login Service"
  require valid-user
  ProxyPass ajp://localhost:8009/grouper/
</Location>
```

Note that the ProxyPass declaration is included only when using mod_proxy_ajp.

Grouper UI-integrated User Authentication

The Grouper UI optionally supports direct integration of external authentication systems with the UI servlet. If this style of providing external authentication services to Grouper is chosen, REMOTE_USER and use of Apache-based user authentication is not needed. See [How to Customize Authentication in the Grouper UI](#).

Grouper Web Service Authentication

Grouper Web Services can be protected by

- REMOTE_USER provided by the servlet container it runs in,
- Apache Rampart for WS-Security style of access protection, or
- a custom authentication plug-in.

See [Authentication for Grouper Web Services](#) for details.

Project Layout



The Grouper API, UI, and WS packages must be unpacked under the same parent directory in order to install successfully. The parent directory must be writable because the Grouper UI build process will attempt to create a build directory under the parent directory.

The directory structure of the unpacked distributions is:

grouper/	top level of API package
conf/	Configuration files for Grouper and third party components
bin/	Grouper command line utilities
dist/	Compiled code, javadoc, and other generated artifacts
lib/	Third party .jar files required by the Grouper API
logs/	Default location for log files
misc/	Currently holds stub instructions for the binary distribution builder
src/	Java source for Grouper API and API test suite
LICENSE	License under which Grouper may be used
build.xml	Ant configuration file
grouper-ui/	top level of UI package
contrib/	Contributed software
doc/	Grouper UI documentation
java/	Java source for Grouper UI
resources/	UI properties files and other resources
webapp/	Directory containing the as-built and customized UI servlet
webapp/grouper	Images and styles for the UI
webapp/i2mi	Generic styles
webapp/WEB-INF	Taglibs and other structural definitions
grouper-ws/	top level of Web Services package
grouper-ws/grouper-ws/	Grouper WS servlet
grouper-ws/grouper-ws/build/	Compiled code
grouper-ws/grouper-ws/doc/	Grouper WS documentation
grouper-ws/grouper-ws/lib/	Third party .jar files required by Grouper WS
grouper-ws/grouper-ws/resources/	Configuration files

grouper-ws/grouper-ws/webapp/	Servlet files
grouper-ws/grouper-ws/webservices/	Apache .aar files
grouper-ws/grouper-ws-java-generated-client/	Sample Axis-generated client
grouper-ws/grouper-ws-java-manual-client/	Sample "Lite" client
grouper-ws/grouper-ws-test/	used for testing

Note: The file grouper/lib/README lists the third party software used by Grouper and identifies the version, source, and license for each.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Software Download

This page last changed on Oct 13, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

This page has moved to <http://www.internet2.edu/grouper/software.html>. Please update your bookmarks and links.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Archives

This page last changed on Jun 04, 2009 by [mchyer](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)



Questions about Grouper's development or a particular feature? [Contact us](#).

Previous versions of the software may also be downloaded from this location.

Archives & Release Details

Click [here](#) for Grouper News, a lower-level, highly-detailed version of the below feature sets through v1.2.0. Similar detail for releases beginning with v1.2.1 can be found in [JIRA](#). Archived Documentation for past releases may also be found below.

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
Grouper v1.4.1	<p><u>Release v1.4.1 includes the following fixes and improvements:</u></p> <ul style="list-style-type: none">• Improved: LDAPPC is included in the Grouper API and is launched via Grouper Shell• Improved: WS: replacing existing members and actAs• Improved: Prevent duplicate stem names• Fixed: Replace hsql file mode with server mode in default grouper configs <p>Grouper v1.4.1 now includes Ldappc, and also has some minor fixes and improvements. See the full list here.</p> <ul style="list-style-type: none">• Grouper QuickStart v1.4.1: Designed to get a working demo up and running quickly. Contains the Grouper components, a demo database, a database engine, and setup instructions. Just add Tomcat and stir...• Grouper API Binary v1.4.1: Binary release of the Grouper Application Program Interface and associated utilities. Cf. Getting started with the Grouper API Binary Distribution• Grouper API Source v1.4.1: Contains the full source for the Grouper	2-February-2009

	<p>Application Program Interface.</p> <ul style="list-style-type: none"> • Grouper UI v1.4.1: Contains the full source for the Grouper User Interface. • Grouper WS v1.4.1: Contains the full source for the Grouper Web Services Interface. • Grouper Client v1.4.1: Experimental client for Grouper LDAP and Web Services, in binary form. Grouper Client v1.4.1 source is also available. 	
Grouper v1.4.0	<p>Release v1.4.0 includes the following fixes and improvements:</p> <ul style="list-style-type: none"> • New: Programmatic hooks for local extensions to Grouper. • New: Automatically maintain groups and memberships from external sources with Grouper Loader. • Added: A binary form of the Grouper Toolkit. • Improved: Grouper Shell, a comprehensive command line interface to Grouper and its allied utilities, is now a core part of the Grouper distribution. And there are correspondingly fewer ant targets in the source distribution. • Improved: Transaction support to ensure relational consistency of logical group operations. • Improved: XML import and export capabilities are incorporated into Grouper Shell. • New: Group type to automatically create include & exclude groups. • New: Configuration directives that provide more granular system administration and security. • New: Ability to merge two Members formerly thought to be distinct. • New: Regex-style validation of group attributes. • Improved: Faster, more efficient Grouper relational database schema. 	4-January-2009

	<ul style="list-style-type: none"> • New: Configuration checking, daily health report, views, and other diagnostic aids. • Fixed: Circumstances in which composites and circular membership loops could produce incorrect membership information. • Improved: API internal design and packaging of classes. • Improved: UI tooltips for group types and custom attributes. • Improved: Support for group details and privileges via Grouper Web Services. # Grouper-QuickStart v1.4.0 - tarball with documentation & instructions for cvs access. # Grouper API Binary v1.4.0 tarball... # Grouper API Source v1.4.0 tarball... # Grouper UI v1.4.0 tarball... # Grouper WS v1.4.0 tarball... # Grouper Client v1.4.0 tarball... # Grouper Client Source v1.4.0 tarball... # Archived Documentation, v1.4.0 (PDF) # API & UI 1.4.0 Javadoc # WS 1.4.0 Javadoc # Grouper Client 1.4.0 Javadoc 	
Grouper v1.3.1	<p><u>Release v1.3.1 includes the following fixes and improvements to the API and UI:</u></p> <ul style="list-style-type: none"> • Some circumstances in which memberships were "orphaned" • Several UI bugs • Memberships of unresolvable Subjects could not be deleted # Grouper-QuickStart v1.3.1 - tarball with documentation & instructions for cvs access. # Grouper API v1.3.1 tarball... # Grouper UI v1.3.1 tarball... # Grouper WS v1.3.1 tarball... 	22-Sep-2008

	# Archived Documentation, v1.3.1 (PDF) # API & UI 1.3.1 Javadoc # WS 1.3.1 Javadoc	
Grouper v1.3.0	<p>Release v1.3.0 includes the following fixes and improvements to the API and UI:</p> <ul style="list-style-type: none"> • New: experimental web services interface. • New: extension for removing memberships for unresolvable subjects. • Improved: user interface. • Improved: Hibernate support. Upgraded to Hibernate 3.2.6 and added transaction support. Hibernate 2.x support is no longer included. • New: applied transactions to the UI so that a whole action succeeds and is committed, or fails and is rolled back. • Added: exception handling and logging to the UI. • Added: ability to disable editing of group attributes / member lists for site configured groups i.e. groups which should be loader maintained. • Added: foreign keys to database. • Improved: performance. • Added: compiled Java in all components includes debug information, which means that stack traces in log files will have line numbers which makes it easier to debug problems. • Improved: new settings in grouper.properties make it possible to define user / db connection urls which can / cannot have their schemas rebuilt. If not configured the test script will prompt the user to confirm that it is OK to drop a schema. This makes it more difficult to accidentally lose data. • Improved: transaction handling code will attempt to inject additional information into a caught Exception. If successful the Exception is not logged - assumes that your code will handle logging. 	22-May-2008

	<ul style="list-style-type: none"> • Added: jsr107cache-1.0.jar - required by default on Solaris. • Fixed: some CSS issues with IE6. • Improved: changed the XHTML declaration to use transitional DTD rather than strict - reduces number of errors until we can try to tidy up issues. • Fixed: Advanced Search link for groups in the UI. • Added: additional error checking in the UI - whether key properties are set and added option to build the generated client when building the web service. • Added: example kerberos authentication configuration for the UI. • Added: gsh source to the Internet2 CVS repository and added a page to the Wiki. • Fixed: gsh.bat - was not correctly building the classpath. • Improved: gsh error reporting. Exceptions from Grouper should now be summarised in gsh output and full stack traces written to grouper_error.log. <p># Grouper-QuickStart v1.3.0</p> <p># Grouper API v1.3.0</p> <p># Grouper UI v1.3.0</p> <p># Grouper WS v1.3.0</p> <p># Archived Documentation, v1.3.0 (PDF)</p> <p># API & UI 1.3.0 Javadoc</p> <p># WS 1.3.0 Javadoc</p>	
Grouper v1.2.1	<p><u>Release v1.2.1 includes the following fixes and improvements to the API and UI:</u></p> <ul style="list-style-type: none"> • Improved: performance • New: API and UI use new strategies to check privileges • Improved: API caching strategy <p># Grouper-QuickStart v1.2.1 - tarball with documentation & instructions for cvs access.</p> <p># Grouper API v1.2.1 tarball...</p>	6-Dec-2007

	# Grouper UI v1.2.1 tarball... # Archived Documentation, v1.2.1 (PDF) # API 1.2.1 Javadoc # UI 1.2.1 Javadoc	
Grouper v1.2.0	<p><u>Release v1.2.0 includes the following functionality improvements and miscellaneous changes:</u></p> <ul style="list-style-type: none"> • Fixed: several critical membership and XML export/import bugs. • Added: sorting of groups, stems, and subjects when browsing or searching in the UI. • Improved: control over attributes used to display groups, stems, and subjects in the UI. • Added: search for groups by their Type and other more advanced group and stem searching capabilities. • Added: stems can be renamed. • Improved: updated the Subject API and GrouperShell. • Added: new, experimental extension framework to ease integration of code external to the core API by defining a common build process and means of referring to the Grouper installation. • Includes updated versions of the Subject API and gsh. # Grouper-QuickStart v1.2 - tarball with documentation & instructions for cvs access. # Grouper API v1.2 tarball... # Grouper UI v1.2 tarball... # API 1.2 Javadoc # UI 1.2 Javadoc 	18-Jul-2007
Grouper v1.1	<p><u>Release v1.1 is mostly a maintenance release; however, there are a few functional and operational enhancements:</u></p> <ul style="list-style-type: none"> • Added: Additional GrouperQuery filters, especially to support provisioning applications. 	30-Nov-06


	<ul style="list-style-type: none"> • Improved: Configurable Privilege and Subject caching regimens. • Improved: UI support for removing many members from a group. • Added: Support for XML Import/Export & custom group types & fields to the GrouperShell command line shell. • Improved: Query performance through caching. # Grouper-QuickStart v1.1 - tarball with documentation & instructions for cvs access. # Grouper API v1.1 tarball... # Grouper UI v1.1 tarball... # Archived Documentation, v1.1 (PDF) 	
Grouper v1.0	<p>Release v1.0 is the first production release of the Grouper product:</p> <ul style="list-style-type: none"> • Added: Group Math. • Added: Custom Group Types, attributes and lists. • Added: Basic XML export-and-import of the Groups Registry. • Added: GrouperShell command line shell - gsh. • Improved: Query performance through caching. <p># Grouper-QuickStart v1.0 - tarball with documentation & instructions for cvs access. # Grouper API v1.0 tarball... # Grouper UI v1.0 tarball... # Archived Documentation, v1.0 (PDF) # API 1.0 Javadoc # UI 1.0 Javadoc</p> <p>The Grouper v1.0 UI embodies our first attempt to enable two tasks that are complicated in a UI context: managing composite groups, and managing custom group types and attributes. We've prepared a brief tutorial to help smooth</p>	20-Jul-06

	your exploration of these new UI capabilities.	
Grouper v0.9	<p><u>Release v0.9 offers the following new and/or different items from the v0.6 pre-release:</u></p> <ul style="list-style-type: none"> • Added: New query API that enables sites to add their own custom queries. • Improved: Enhancements to the Access and Naming interfaces. • Added: New "all" subject that can be assigned memberships and granted privileges that map to all subjects identifiable through the Subject API. • Added: "Wheel" group whose members have root-like privileges within the API. • Improved: Enhanced UI support for searching. • Improved: Enhanced UI support for management of effective memberships and privileges. • Improved: Logging capabilities. <p>The v0.9 toolkit includes a full UI designed to be deployed to a java application server, java source implementing the Grouper 0.9 API, documentation for developers and implementers, and sample utilities.</p> <p># Grouper-QuickStart v0.9 tarball with documentation & instructions for cvs access.</p> <p># Grouper API v0.9 tarball...</p> <p># Grouper UI v0.9 tarball...</p> <p># archived docs Archived Documentation, v0.9:</p> <p>Grouper API</p> <p>Grouper</p> <p>Changes</p> <p>Grouper ERD</p> <p>Grouper</p> <p>Javadoc</p> <p>Grouper</p> <p>Known Issues</p> <p>Grouper News</p>	19-Dec-05

	Grouper Roadmap Grouper User Interface UI - Demo UI - Components UI - Architecture UI - Struts Actions & Tiles UI - Customizing the UI UI - Contributed Code UI - Development Environment UI components	
Grouper v0.6	<p><u>Release v0.6 is the initial UI pre-release, supporting Phase 1 functionality:</u></p> <ul style="list-style-type: none"> • Improved: Subject interface. • Added: Full implementations of the privilege interfaces. • Fixed: Proper schema validation. (#267) • Added: GrouperAccess.has(). • Improved: search and browse capabilities. • Added: Search by the name, extension, or displayName attributes of groups. [S&B#1] • Added: Search by the name, extension, or displayName attributes of namespaces. [S&B#2] • Added: Search by the name, extension, or displayName attributes of groups scoped to groups within a namespace subordinate to a given stem. [S&B#7] • Added: Search by the name, extension, or displayName attributes of namespaces scoped to namespaces subordinate to a given stem. [S&B#8] • Added: More verbose and precise error reporting via exceptions. • Added: Ability to delete groups containing members. 	16-Sep-05

	<p># Grouper-QuickStart v0.6 tarball with documentation & instructions for cvs access.</p> <p># Grouper API v.6 tarball...</p> <p># Grouper UI v0.6 tarball...</p>	
Grouper v0.5.6	<p>The API release v0.5.6 +contains fixes to several bugs found in the v0.5.5 pre-release. From its Newsfile:</p> <ul style="list-style-type: none"> • Fixed: Effective memberships for non-"members" lists (e.g. access and naming privileges) were being calculated incorrectly. (#350) • Fixed: Non-root subjects could not create stems or groups. (#353) • Fixed: STEM, not ADMIN, needed to modify namespace attributes. (#352) • Fixed: Added NullGrouperAttribute class (which extends GrouperAttribute) to handle group attributes that either do not have values or have had their values deleted. (#356) • Fixed: GrouperMember.load(session subject) replaces GrouperMember.load(subject). (#348) • Fixed: GrouperGroup.loadByID() now returns properly casted GrouperGroup objects. (#349) • Fixed: GrouperStem.loadByID() now returns properly casted GrouperStem objects. (#349) <p># Grouper API v0.5.6 tarball with documentation & instructions for cvs access.</p>	29-Apr-05
Grouper v0.5.1	<p><u>Release v0.5.1 is a maintenance release, and includes the following additions and revisions:</u></p>	

	<ul style="list-style-type: none"> • Revised: Hibernate session and transaction handling code. • Revised: effective membership algorithm. • Added: GrouperStem class for managing and representing namespaces. • Improved: compatibility with Oracle. • Added: public methods to list all groups and stems within a given stem. <p># Grouper v0.5.1 tarball with documentation & instructions for cvs access.</p>	
Grouper v0.5	<p>The API release v0.5 is the first release of Grouper:</p> <ul style="list-style-type: none"> • Added: Creation, update, and removal of groups from the Groups Registry. • Added: Subgroups. • Added: Export capabilities. • Added: Limited querying. • Added: Graphical user interface for manual groups management. • Added: Logging. <p># Grouper API v0.5 tarball with documentation & instructions for cvs access.</p>	Dec-04

 Questions or comments?  [Contact us.](#)

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Coding new DDL

This page last changed on Jun 04, 2009 by [mchyzer](#).

This document helps developers work with Grouper DDL. If you work with Grouper DDL, please keep this document up to date.

The first thing to know about is the grouper_ddl table. This has one entry for each ddl type. A ddl type means database objects with a certain prefix (e.g. the grouper one starts with grouper_, the default subject one starts with subject, and the organization management one starts with grouperorgs_). On startup, grouper will see if the version in the DB matches the version in the jar (an enum). If not, an error will be logged, and optionally grouper will not startup.

Lets do some use cases

1. Add a column to a grouper_ table

- Look in the class GrouperDdl. See which version the table is created where the column is.

Group Math v1.0

This page last changed on Jul 18, 2007 by jbibbee@internet2.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

This document will explain how Group Math works, and how you can best apply it to your environment.

... more to come soon!



Questions or comments?



Contact us.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

News

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

Grouper News

This document contains every change, addition, etc. that is born with each release, and will continually be updated.

Version	Details	Release Date
---------	---------	--------------

	<u>Changes:</u>	<i>-Jul-06</i>
--	------------------------	-----------------------

Grouper v1.2.0

- UI NEW: Sorted lists of groups, stems and subjects when browsing or searching.
- UI NEW: Advanced stem search allows finer control of searching.
- UI NEW: Import and export of membership lists from / to simple delimited text files, i.e., comma or tab separated.
- UI CHANGE: Possible to configure default attributes to use for displaying groups, stems and subjects (configured by source.)
- UI CHANGE: Advanced groups search now includes ability to search by group type.
- UI CHANGE: Support display of multi-values in subject summary view. This ought to work - but so far we have not had any to play with...
- UI CHANGE: HTML forms now use the POST method.
- API NEW: added GrouperQuery support for selecting groups by their Type.
- API NEW: Operational attributes (createTime, creator) are now maintained for Memberships as they have been for Groups and Stems, in preparation for future support of aging of Memberships.
- API NEW: Stems can be renamed.
- API NEW: A prototype extension framework to ease integration of code external to the core API by defining a common build process and means of referring to the Grouper installation.
- API NEW: Updated Subject API from v0.2.1 to v0.3.0rc1-cvs to take advantage of security and performance improvements. Note that existing JDBCSourceAdapter configuration must change somewhat.
- API NEW: An experimental DAO layer has been introduced to separate business logic from persistence logic.
- API NEW: The default privilege caching TTL has been changed.
- API NEW: GrouperShell

Grouper v1.1	<u>Changes:</u> <ul style="list-style-type: none"> • ADDED: Additional GrouperQuery filters, especially to support provisioning applications. • IMPROVED: Configurable Privilege and Subject caching regimens. • IMPROVED: UI support for removing many members from a group. • ADDED: Support for XML Import/Export & custom group types & fields to the GrouperShell command line shell. • IMPROVED: Query performance through caching. 	30-Nov-06
	<u>Changes:</u>	20-Jul-06

Grouper v1.0

- BUGFIX: Naming privileges not properly exported to XML.
- BUGFIX: Improper calculation of forward effective memberships to delete.
- BUGFIX: Improper setting of parent membership in select effective memberships.
- BUGFIX: Improper calculation of select child memberships to delete when deleting an immediate memberships.
- NEW: Added **GrouperShell** (located in `contrib/gsh`). **gsh** is a command-line shell that may be used in a batch or interactive manner to interact with and manipulate the Groups Registry.
- UPDATE: Updated Subject API to version 0.2.1.
- NEW: Basic integration of Gary Brown's XML export and import code.
- BUGFIX: Fixed another Hibernate lazy collection initialization error that was brought to light by work on **cdg**.
- BUGFIX: Improper calculation of **Membership** UUIDs when deleting memberships.
- BUGFIX: `Group.delete()` did not remove composites.
- BUGFIX: `Membership.getViaGroup()` does not handle composites
- NEW: Added **CompositeFinder**
- NEW: Added `CompositeFinder.findAsFacto
- NEW: Added `CompositeFinder.findAsOwne
- NEW: Added `Composite.getLeftGroup()`
- NEW: Added `Composite.getOwnerGroup()
- NEW: Added `Composite.getRightGroup()`
- NEW: Added `Composite.getType()`
- NEW: Added `MembershipFinder.findComp Group, Subject)`
- NEW: Added `Group.getCompositeMember
- NEW: Added `Group.getCompositeMember
- NEW: Added `Group.getRemovableTypes()` that will return all group types that can be removed

Grouper v0.9

Changes: * **NEW:** New public API. * **NEW:** New query API. This includes the ability for sites to create their own custom query filters for use within the query API.

- NEW: New access and naming adapter APIs and interfaces.
- NEW: "all" subject that can be assigned memberships and granted privileges that maps to all subjects that are identifiable by Grouper's Subject API configuration.
- NEW: Event logging
- NEW: Configuration options for controlling what privileges, if any, are granted to the "all" subject whenever a new group or stem is created. By default, *READ* and *VIEW* are granted to "all" upon group creation and no privileges are granted to "all" upon stem creation.
- NEW: Subject IDs associated with *Member* objects within the Groups Registry can now be changed with the *Member.setSubjectId()* method.
- NEW: Internal source adapter for resolving the "root" (*GrouperSystem*) and "all" (*GrouperAll*) subjects. No configuration is necessary to use this adapter within Grouper.
- NEW: Experimental wheel-group support. This is disabled by default. If enabled, all members of the group are treated as root-like subjects with all access and naming privileges.
- UPDATE: New internals.
- UPDATE: ehcache-based caches for access and naming privilege resolution.
- UPDATE: Grouper source adapter no longer needs to be configured within *conf/sources.xml*.
- UPDATE: License changed to Apache License, Version 2.0

Changes:

16-Sep-2005

--	--	--

Grouper v0.6

- NEW: VIEW Access privilege (#339)
- NEW: READ Access privilege (#338)
- NEW: OPTIN Access privilege (#343)
- NEW: OPTOUT Access privilege (#344)
- NEW:
GrouperGroup.getDisplayExtension() method
- NEW:
GrouperGroup.getDisplayName() method
- NEW:
GrouperGroup.getExtension() method
- NEW:
GrouperGroup.getMembers() method
- NEW:
GrouperGroup.getName() method
- NEW:
GrouperGroup.getStem() method
- NEW:
GrouperMember.getMember() method
- NEW:
GrouperMember.source() method
- NEW: *GrouperQuery.base()* filter method
- NEW:
GrouperQuery.group() filter method
- NEW:
GrouperQuery.group() scoped filter method (#403)
- NEW:
GrouperQuery.groupAttr() filter method
- NEW:
GrouperQuery.groupAttr() scoped filter method (#403)
- NEW:
GrouperQuery.stem() filter method
- NEW:
GrouperQuery.stem() scoped filter method (#403)
- NEW:
GrouperQuery.stemAttr() filter method
- NEW:
GrouperQuery.stemAttr() scoped filter method (#403)
- NEW:
GrouperQuery.getGroups() method
- NEW:
GrouperQuery.getListValues() method
- NEW:
GrouperQuery.getMembers() method

Grouper v0.5.6	<ul style="list-style-type: none"> • FIXED: Effective memberships for non-"members" lists (e.g. access and naming privileges) were being calculated incorrectly (#350) • FIXED: Non-root subjects could not create stems or groups (#353) • FIXED: STEM, not ADMIN, needed to modify namespace attributes (#352) • FIXED: Added NullGrouperAttribute class (which extends GrouperAttribute) to handle group attributes that either do not have values or have had their values deleted. (#356) • FIXED: GrouperMember.load(session subject) replaces GrouperMember.load(subject) (#348) • FIXED: GrouperGroup.loadByID() now returns properly casted GrouperGroup objects (#349) • FIXED: GrouperStem.loadByID() now returns properly casted GrouperStem objects (#349) 	
	Changes:	15-Apr-2005



Grouper v0.5.5

- UPDATED: Hibernate internals completely rewritten to improve session and transaction handling
- UPDATED: New and more thoroughly tested implementation of the `memberOf` algorithm. In addition, we are now tracking the entire via chain for effective memberships.
- NEW: Improved Oracle compatibility and support
- NEW: Created logical distinction between groups (`GrouperGroup`) and namespaces (`GrouperStem`)
- NEW: `GrouperGroup.hasMember()` method for verifying whether a member belongs to a group (#328)
- NEW: `GrouperMember.isMember()` method for verifying whether a member belongs to a group (#328)
- NEW: `GrouperStem.stems()` method to retrieve immediate child namespaces within a namespace
- NEW: `GrouperStem.groups()` method to retrieve immediate child groups within a namespace
- NEW: `GrouperSession` objects now serializable (#296)
- NEW: More detailed reporting (via exceptions) of errors caused by various runtime error conditions
- NEW: Now using Commons DBCP by default for connection pooling
- NEW: Now using Hibernate named queries defined as defined in *conf/Grouper.hbm.xml*
- NEW: Use Ant's SQL task to to create, initialize and reset HSQLDB database (#287)
- FIXED: Removed calls to `system.exit()` (#179) (David Langenberg, The University Of Chicago)
- FIXED: `sessionID` generation bug (#278)
- FIXED: Effective membership bug with circular group memberships (#286)
- NEW: `GrouperField` objects now have public instance

--	--	--

Grouper v0.5

- "Base" and "Naming" group types* Immediate Memberships
 - Effective Memberships
 - Search: By immediate and effective membership
 - Search: By group type
 - Search: By group create time
 - Search: By group modify time
 - Access privilege interface
 - Naming privilege interface
 - An implementation of the access privilege interface that uses groups to manage privileges
 - An implementation of the naming privilege interface that uses groups to manage privileges
 - ADMIN and UPDATE access privileges
 - CREATE and STEM naming privileges
 - A partial implementation of the I2MI Subject interface for locally-defined people subjects
 - A partial implementation of the I2MI Subject interface for groups as subjects
 - Basic Event Logging
 - Contributed: I2MI Subject Loader
 - Contributed: Group Loader
 - Contributed: Member Loader
 - Contributed: Query Program
- Known Bugs:**
- Grouper does not fail gracefully or even necessarily informatively
 - Insufficient data validation within code
 - Not all database updates are atomic transactions (#233) (#248)
 - Loading groups by *groupID* does not always fail cleanly.
 - *GrouperQuery* objects do not properly reset their state (#255)
 - *Modify* attributes *may* be incorrect (#247)
 - Session Handling is dubious at best (#173)
 - Insufficient documentation
 - Insufficient test coverage
 - Slow

 Questions or comments?  [Contact us.](#)

[GROUPER:](#) [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

v0.0.1_gsh

This page last changed on Jan 04, 2008 by jbibbee@internet2.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)



This is an archived page. To view the current page, please see [GrouperShell](#). Thanks!

GrouperShell v0.0.1

GrouperShell (gsh) is a shell for administering and interacting with the Grouper API. It can be used in both a batch and interactive manner.

Build GrouperShell

```
% cd grouper/contrib/gsh % ant
```

Test GrouperShell's scripting capabilities

```
% ant test
```

Grouper must have a JDBCSourceAdapter for subjects for the test suite to complete successfully. This need is adequately met by testing GrouperShell using the same grouper/conf directory contents used to test the Grouper API.

Note: In some environments the tests may fail due to a bug exposed by the testing procedure which might not indicate any actual error with the gsh utility itself. This is known to occur, for example, under cygwin on Windows. An alternative testing process is to run each of the test suites, which are gsh scripts, individually as follows:

```
% bin/gsh.sh src/test/test.gsh % bin/gsh.sh src/test/groups.gsh % bin/gsh.sh src/test/stems.gsh % bin/gsh.sh  
src/test/composites.gsh % bin/gsh.sh src/test/privs.gsh
```

Build `gsh.jar` file in the `dist` subdirectory

```
% ant jar
```

Build javadoc in `doc/html`

```
% ant html
```

Use GrouperShell

- Run **GrouperShell** in an interactive manner from a Unix-like environment:

```
% ./bin/gsh.sh
```

- Run **GrouperShell** (crudely) from Ant:

```
% ant shell
```

- Read **GrouperShell** commands from STDIN:

```
% ./bin/gsh.sh -
```

- Read **GrouperShell** commands from a script file:

```
% ./bin/gsh.sh /path/to/your/script.gsh
```

GrouperShell Commands

Command	Description
addComposite(group, type, left group, right group.)	Add composite membership.
addGroup(parent, extension, displayExtension)	Add group beneath parent stem with the specified extension and displayExtension.
addMember(group, subject id)	Add subject as a member to the group.
addRootStem(extension, displayExtension)	Add root stem with the specified extension and displayExtension.
addStem(parent, extension, displayExtension)	Add stem beneath parent stem with the specified extension and displayExtension.
addSubject(id, type, name)	Add a HibernateSubject to the Groups Registry.
delComposite(group)	Delete composite membership from the specified group.
delGroup(name)	Delete group with the specified name.
delMember(group, subject id)	Remove subject as a member of the group.
delStem(name)	Delete stem with the specified name.
exit	Terminate shell.
findSubject(id)	Find a subject.
findSubject(id, type)	Find a subject.

findSubject(id, type, source)	Find a subject.
getGroupAttr(stem, attr)	Get value of group's attr attribute.
getGroups(name)	Find all groups with name in any naming attribute value.
getMembers(group)	Get members of the group.
getSources()	Find all Subject sources.
getStemAttr(stem, attr)	Get value of stem's attr attribute.
getStems(name)	Find all stems with name in any naming attribute value.
grantPriv(name, subject id, privilege)	Grant privilege to subject id on name. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
hasMember(group, subject id)	Is subject a member of this group.
hasPriv(name, subject id, privilege)	Does subject id have privilege on name? <i>privilege</i> must be a <i>n_AccessPrivilege_</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
help()	Display usage information.
history()	Print commands that have been run.
history(N)	Print the last N commands that have been run.
last()	Run the last command executed.
last(N)	Execute command number N.
p(command)	Pretty print results. This command is more useful when GSH_DEVEL is enabled.
quit	Terminate shell.
resetRegistry()	Restore the Groups Registry to a default state.
revokePriv(name, subject id, privilege)	Revoke privilege from subject id on name. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
setGroupAttr(group, attr, value)	Set value of <i>group's attr</i> attribute.
setStemAttr(stem, attr, value)	Set value of <i>stem's attr</i> attribute.
version()	Return version information.

In addition, any Grouper API method can be directly invoked just by referencing it, inclusive of the class in which it is defined. And methods return a java object which can be stored in a variable. For example, the following gsh session determines all of the groups to which a given subject belongs:

```
gsh-0.0.1 0% subj = findSubject("SD00125") subject: id='SD00125' type='person'
source='kitn-person' name='Barton, Tom' gsh-0.0.1 1% sess = GrouperSession.start(subj)
```

```
edu.internet2.middleware.grouper.GrouperSession: 29c40f97-9fb0-4e45-88bc-
a14877a6c9b5,'SD00125','person' gsh-0.0.1 2% member = MemberFinder.findBySubject(sess, subj) member:
id='SD00125' type='person' source='kitn-person' uuid='d0fa765e-1439-4701-89b1-9b08b4ce9daa' gsh-0.0.1
3% member.getGroups() group: name='etc:wheel' displayName='Grouper Administration:Wheel Group'
uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

GrouperShell Variables

Variable	Description
GSH_DEBUG	If set to true, stack traces will be printed upon failure.
GSH_DEVEL	If set to true, summaries of returned objects are not automatically printed.
GSH_TIMER	If set to true, the time taken to evaluate each command will be displayed.

Example:

```
gsh-0.0.1 4% GSH_DEVEL = true gsh-0.0.1 5% subj = findSubject("SD00125") gsh-0.0.1 6% sess =
GrouperSession.start(subj) gsh-0.0.1 7% member = MemberFinder.findBySubject(sess, subj) gsh-0.0.1
8% p(member.getGroups()) group: name='etc:wheel' displayName='Grouper Administration:Wheel Group'
uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

 Questions or comments?  [Contact us.](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

v0.1.0_gsh

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)



This is an archived document. For the current version, please see [GrouperShell \(gsh\)](#)

GrouperShell v0.1.0

GrouperShell (gsh) is a shell for administering and interacting with the Grouper API. It can be used in both a batch and interactive manner.

API Compability

This version of of **GrouperShell** is compatible with Grouper v1.2.0.

Build GrouperShell

```
% cd grouper/contrib/gsh % ant
```

Test GrouperShell's scripting capabilities

```
% ant test
```

Grouper must have a JDBCSourceAdapter for subjects for the test suite to complete successfully. This need is adequately met by testing GrouperShell using the same grouper/conf directory contents used to test the Grouper API.

Note: In some environments the tests may fail due to a bug exposed by the testing procedure which might not indicate any actual error with the gsh utility itself. This is known to occur, for example, under cygwin on Windows. An alternative testing process is to run each of the test suites, which are gsh scripts, individually as follows:

```
% bin/gsh.sh src/test/test.gsh % bin/gsh.sh src/test/groups.gsh % bin/gsh.sh src/test/stems.gsh % bin/gsh.sh src/test/composites.gsh % bin/gsh.sh src/test/privs.gsh
```

Build `gsh.jar` file in the `dist` subdirectory

```
% ant jar
```

Build javadoc in `doc/html`

```
% ant html
```

Use GrouperShell

- Run **GrouperShell** in an interactive manner from a Unix-like environment:

```
% ./bin/gsh.sh
```

- Run **GrouperShell** (crudely) from Ant:

```
% ant shell
```

- Read **GrouperShell** commands from STDIN:

```
% ./bin/gsh.sh -
```

- Read **GrouperShell** commands from a script file:

```
% ./bin/gsh.sh /path/to/your/script.gsh
```

GrouperShell Commands

Command	Description
addComposite(group, type, left group, right group.)	Add composite membership.
addGroup(parent, extension, displayExtension)	Add group beneath parent stem with the specified extension and displayExtension.
addMember(group, subject id)	Add subject as a member to the group.
addRootStem(extension, displayExtension)	Add root stem with the specified extension and displayExtension.
addStem(parent, extension, displayExtension)	Add stem beneath parent stem with the specified extension and displayExtension.
addSubject(id, type, name)	Add a HibernateSubject to the Groups Registry.
delComposite(group)	Delete composite membership from the specified group.
delGroup(name)	Delete group with the specified name.
delMember(group, subject id)	Remove subject as a member of the group.
delStem(name)	Delete stem with the specified name.
exit	Terminate shell.

findSubject(id)	Find a subject.
findSubject(id, type)	Find a subject.
findSubject(id, type, source)	Find a subject.
getGroupAttr(stem, attr)	Get value of group's attr attribute.
getGroups(name)	Find all groups with name in any naming attribute value.
getMembers(group)	Get members of the group.
getSources()	Find all Subject sources.
getStemAttr(stem, attr)	Get value of stem's attr attribute.
getStems(name)	Find all stems with name in any naming attribute value.
grantPriv(name, subject id, privilege)	Grant privilege to subject id on name. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
groupAddType(group, type)	Add group type <i>type</i> to <i>group</i> .
groupDelType(group, type)	Delete group type <i>type</i> from <i>group</i> .
groupGetTypes(group)	Get <i>group</i> 's group types.
groupHasType(group, type)	Check whether <i>group</i> has group type <i>type</i> .
hasMember(group, subject id)	Is subject a member of this group.
hasPriv(name, subject id, privilege)	Does subject id have privilege on name? <i>privilege</i> must be a <i>n_AccessPrivilege_</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
help()	Display usage information.
history()	Print commands that have been run.
history(N)	Print the last N commands that have been run.
last()	Run the last command executed.
last(N)	Execute command number N.
p(command)	Pretty print results. This command is more useful when GSH_DEVEL is enabled.
quit	Terminate shell.
resetRegistry()	Restore the Groups Registry to a default state.
revokePriv(name, subject id, privilege)	Revoke privilege from subject id on name. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. <i>AccessPrivilege.ADMIN</i>) or <i>NamingPrivilege</i> (e.g. <i>NamingPrivilege.STEM</i>) constant.
setGroupAttr(group, attr, value)	Set value of <i>group</i> 's <i>attr</i> attribute.

setStemAttr(stem, attr, value)	Set value of <i>stem</i> 's <i>attr</i> attribute.
typeAdd(name)	Create custom group named <i>name</i> .
typeAddAttr(type, name, read, write, required)	Create a custom group attribute named <i>name</i> on group type <i>type</i> . <i>read</i> and <i>write</i> must be an AccessPrivilege (e.g. AccessPrivilege.ADMIN).
typeAddList(type, name, read, write)	Create a custom membership list named <i>name</i> on group type <i>type</i> . <i>read</i> and <i>write</i> must be an AccessPrivilege (e.g. AccessPrivilege.ADMIN).
typeDel(name)	Delete the group type named <i>name</i> .
typeDelField(type, name)	Delete the custom field named <i>name</i> from group type <i>type</i> .
typeFind(name)	Find the group type named <i>name</i>
typeGetFields(name)	Get the fields associated with the group type named <i>name</i> .
version()	Return version information.
xmlFromFile(filename)	Load Groups Registry with XML contained in <i>filename</i> .
xmlFromString(xml)	Load Groups Registry with XML in the <i>xml</i> string.
xmlFromURL(url)	Load Groups Registry with XML located at <i>url</i> .
xmlToFile(filename)	Exports Groups Registry to <i>filename</i> .
xmlToString()	Exports Groups Registry to string.
xmlUpdateFromFile(filename)	Update Groups Registry with XML contained in <i>filename</i> .
xmlUpdateFromString(xml)	Update Groups Registry with XML in the <i>xml</i> string.
xmlUpdateFromURL(url)	Update Groups Registry with XML located at <i>url</i> .

In addition, any Grouper API method can be directly invoked just by referencing it, inclusive of the class in which it is defined. And methods return a java object which can be stored in a variable. For example, the following gsh session determines all of the groups to which a given subject belongs:

```
gsh-0.0.1 0% subj = findSubject("SD00125") subject: id='SD00125' type='person'
source='kitn-person' name='Barton, Tom' gsh-0.0.1 1% sess = GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession: 29c40f97-9fb0-4e45-88bc-
a14877a6c9b5,'SD00125','person' gsh-0.0.1 2% member = MemberFinder.findBySubject(sess, subj) member:
id='SD00125' type='person' source='kitn-person' uuid='d0fa765e-1439-4701-89b1-9b08b4ce9daa' gsh-0.0.1
3% member.getGroups() group: name='etc:wheel' displayName='Grouper Administration:Wheel Group'
uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```



GrouperShell Variables

Variable	Description
----------	-------------

GSH_DEBUG	If set to true, stack traces will be printed upon failure.
GSH_DEVEL	If set to true, summaries of returned objects are not automatically printed.
GSH_TIMER	If set to true, the time taken to evaluate each command will be displayed.

Example:

```
gsh-0.0.1 4% GSH_DEVEL = true gsh-0.0.1 5% subj = findSubject("SD00125") gsh-0.0.1 6% sess =
GrouperSession.start(subj) gsh-0.0.1 7% member = MemberFinder.findBySubject(sess, subj) gsh-0.0.1
8% p(member.getGroups()) group: name='etc:wheel' displayName='Grouper Administration:Wheel Group'
uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

 Questions or comments?  [Contact us.](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

v1.0_API Configuration

This page last changed on Jan 04, 2008 by jbibbee@internet2.edu.

GROUPEE: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)



This is an archived document. For the current version, please see [Configuring the Grouper API](#). Thanks!

Configuring the Grouper API v1.0

In this section we describe all of the Grouper API configuration files and important settings.

Section	Configuration File	Purpose
Database-Related Settings and Procedures	grouper.hibernate.properties	integrating the Grouper API with the database that will house your Groups Registry
Configuration of Source Adapters	sources.xml	integrating the Grouper API with chosen identity sources
Grouper Privileges	grouper.properties	defaults for Grouper privileges, enabling identified external users to act with elevated root-like privilege
Logging	log4j.properties, grouper.properties	logging

Database-Related Settings and Procedures

Grouper uses Hibernate to persist objects in the Groups Registry, so all of the database-specific settings are located in `grouper/conf/grouper.hibernate.properties`. After modifying the default properties as needed, Grouper API ant tasks detailed below are used to create and install the Groups Registry schema and initialize the database.

General Property Settings

The `grouper/conf/grouper.hibernate.properties` file included in the Grouper API distribution contains sections pre-populated for HSQLDB, Postgresql, and Oracle. If you're using one of these, some of your configuration effort is just adding and removing comment characters. For others, it may be necessary to refer to more detailed [Hibernate Configuration Information](#).

The basic properties that must be set are:

Property Name	Purpose
hibernate.connection.driver_class	JDBC driver classname
hibernate.connection.url	JDBC URL for the database
hibernate.connection.username	database user
hibernate.connection.password	database user's password

and one that probably **ought** to be set is:

hibernate.dialect	classname of a Hibernate dialect, for setting platform specific features. Choices are listed here .
--------------------------	---

You may need to get a database support person to tell you what the values of these parameters must be for the instance of the database being configured.

Database-Specific Property Settings

In this section, we collect database-specific settings that we've become aware of. If your database technology is listed here, you may wish to follow the specific instructions for that technology.

Oracle 9i - Grouper uses Apache DBCP for JDBC connection pooling and enables prepared statement pooling by default. Prepared statement pooling must be disabled for Oracle 9i with the setting:

```
hibernate.dbcp.ps.maxIdle = 0
```

Database Initialization Procedure

After setting Hibernate properties for your database, change your command shell to the grouper directory and execute the following two ant tasks to install the appropriate Groups Registry DDL and perform necessary initialization:

ant schemaexport - Generates DDL appropriate for your configured RDBMS and installs the tables.

ant db-init - Populates various tables with required logical schema information (the default set of group types and fields) and creates the root naming stem of the Groups Registry in the configured database.

If you've performed the junit testing using your production database, or for any other reason need to return the Groups Registry to its initial pristine state, do

ant db-reset - Cleans up the database, returning it to its just-initialized state.

Configuration of Source Adapters

Grouper uses [\[i2miCommon:Subject API\]](#) compliant "source adapters" to integrate with external identity stores. "Subjects" are the objects housed there that are presented to Grouper for management vis-à-vis group membership and Grouper privileges. These may represent people, other groups, computers, applications, services, most anything for which you manage identity. With the exception of Grouper groups, Grouper treats all subjects opaquely. See the [\[i2miCommon:Subject API\]](#) documentation for further background and details concerning subjects, source adapters, and other aspects of the Subject API.

Each source adapter connects with a single back-end store using JDBC or JNDI. Grouper makes no specific assumptions about the schema of any subject types. Instead, sections of the configuration file, grouper/conf/sources.xml, declare the details of how to connect with each back-end store, the identifier(s) to be used for the subjects it contains, how to select and search for subjects, and which subject attributes should be made available to Grouper.

Grouper 1.0 relies on v0.2.1 of the Subject API. Please refer to the section on [\[Subject API v0.2.1\]](#) for detailed configuration information.

Three source adapter classes are included in the Grouper API 1.0 package. JDBCSourceAdapter and JNDISourceAdapter classes are included in subject-0.2.1.jar, and GrouperSourceAdapter is built along with the Grouper API. Every Grouper API deployment MUST include a *source* element in grouper/conf/sources.xml for the GrouperSourceAdapter so that Grouper can refer to its own groups in the same manner as other subjects.

Choosing Identifiers for Subjects

Identifiers and their management can get complicated. They can be revoked or not, re-assigned or not, lucent or opaque, etc. Depending on such characteristics, a given identifier might be a good or bad choice to use in the context of managing the identified subject's group memberships.

For example, a username is often lucent - easily remembered by the person to whom it is associated. But it may also be revokable, meaning that it no longer refers to that person (perhaps they have a new one), or even re-assignable, meaning that it might refer to some other person at a later time. If a username is used to record membership, username changes must trigger corresponding membership changes. A username is better suited to authentication than it is to indicating membership.

On the other hand, an opaque registryID (machine, not human, readable) that never changes is great for membership, but lousy for authentication - it might not even be known by the person to whom it is associated. How would I identify myself to Grouper if I wished to opt-in to a list or manage a group?

Grouper accommodates subject identifier issues in two ways. First, it maintains UUIDs for every subject and group within the Groups Registry. These are never exposed by the API, but are associated with externally supplied subject identifiers within the Groups Registry. This approach allows the identifier associated with a given subject to be changed without any need to change actual memberships.

Second, by relying on the Subject API, Grouper is able to lookup subjects that are presented with an identifier in one namespace and obtain identifiers in other namespaces for that subject. That means that it can translate a username into a registryID, for example. So, when a user authenticates to an application using the Grouper API, that application can use the Subject API to fetch an identifier for the person chosen by the site for use in memberships. Similarly, when a membership in the Groups Registry is to be expressed elsewhere, the identifier used for group members can be translated by a provisioning connector by use of the Subject API into one that is suitable in the provisioned context.

Grouper Privileges

All configuration of Grouper privileges detailed in this section occur in the grouper/conf/grouper.properties file.

Default privileges

Grouper requires that all subjects must be explicitly granted access or naming privileges (cf [Glossary](#)), with one caveat. There is a special "subject" internal to Grouper called the ALL subject, which is a stand-in for any subject. The ALL subject can be granted a privilege in lieu of assigning that privilege explicitly to each and every subject.

When a new group or naming stem is created, any of its associated privileges can be granted by default to the ALL subject. This is configured by a series of properties in grouper.properties, one per privilege. If a property has the value "true" then ALL is granted that privilege by default when a group or naming stem is created. Otherwise it is not, and hence no subject has that privilege by default.

Property Name	Value in Grouper 1.0 Distribution
groups.create.grant.all.admin	false
groups.create.grant.all.optin	false
groups.create.grant.all.optout	false
groups.create.grant.all.update	false
groups.create.grant.all.read	true
groups.create.grant.all.view	true
stems.create.grant.all.create	false

stems.create.grant.all.stem	false
-----------------------------	-------

Super-user Privileges

Grouper has another special "subject" called GrouperSystem that acts as a super-user. GrouperSystem is permitted to do everything - the privilege system is ignored for that special subject. Grouper can be configured to consider all members of a distinguished group to be able to act as super-users, much as the "wheel" group does in BSD Unix. Two properties control this behavior:

Property Name	Description
groups.wheel.use	"true" or "false" to enable or disable this capability.
groups.wheel.group	The group name of the group whose members are to be considered security-equivalent to GrouperSystem.

Note that, as of v1.0, the Grouper UI enables users that belong to the wheel group to choose when to act with the privileges of GrouperSystem and when to act as their normal selves.

Using a privilege management system external to Grouper

Grouper's internal security implementation relies on two java interfaces, one for Naming Privileges and another for Access Privileges. Grouper ships with classes that implement these interfaces, but 3rd parties are free to supply their own and so manage Grouper privileges using a privilege management system external to Grouper. Two properties declare the java classes that Grouper will use to implement these interfaces:

Property Name	Description
privileges.access.interface	classname of the java class that implements the Access Interface
privileges.naming.interface	classname of the java class that implements the Naming Interface

Note: although we've provided the can and the dish, we haven't as yet eaten our own dogfood!

Logging



Logging is configured in the grouper/conf/log4j.properties configuration file. By default Grouper will write event log information to grouper/grouper_event.log, error logging to grouper/grouper-error.log, and debug logging, if enabled, to grouper/grouper-debug.log. The log4j configuration can be adjusted to control the verbosity, type and output of Grouper's logging.

In addition, there are several configuration parameters in grouper/conf/grouper.properties that may be adjusted to control the logging of effective membership modifications in the event log.

```
# Control whether the addition and deletion of effective groups memberships
# are logged in the event log. If using the _GrouperAccessAdapter_ this
# will include granted and revoked access privileges.
memberships.log.group.effective.add = true
memberships.log.group.effective.del = true

# If using _GrouperNamingAdapter_, control whether the granting and
# revoking of effective naming privileges are logged in the event log.
```

memberships.log.stem.effective.add = true
memberships.log.stem.effective.del = true

 Questions or comments?  [Contact us.](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)



This is an archived page. For the current version, please see [Grouper XML Import/Export](#). Thanks!

Grouper XML Import / Export for Grouper v1.0

For version 1.0, Grouper includes XML import / export tools. Exported XML may be used for:

- provisioning to other systems
- reporting
- backups
- switching database backends - including to upgraded schemas (required by new Grouper API versions) in the same database

Imported XML may be used for:

- loading - adding to or updating existing Stems, Groups and Group Types. Whole or partial Grouper registries can be exported, and subsequently imported at a specified Stem (or the Root Stem if not specified) in the *new* instance.*
- initialising a new, *empty* registry to a known state** - useful for demos, testing and system recovery

In general, exported data can be imported into the same Grouper instance it was exported from, or a different instance. Stems and Groups and Group Types will be created, if not already present, or updated if they already exist (depending on import options provided).

The XML formats for import and export are very similar, however, there are some differences.

The export format:

- defines what is actually exported,
- includes some meta data about the export,

while the import format:

- allows import options to be embedded in the XML,
- defines additional attributes for Stems and Groups which may affect the importing of Stems and Groups,
- does not require all of the information that is exported.

Any tool which can create XML, in the correct format, can be used as a loader.

*To successfully load Subject data, the new Grouper instance must be configured with the same Subject Sources. The export tool does not export Subject registries. Subjects which cannot be resolved will be logged, but otherwise ignored.

**Although data can be exported from one Grouper instance and imported into another, system attributes are not maintained. A group with the same name will have a different uuid and create times, etc, will reflect the time of import rather than the creation time in the original Grouper instance. A future version of the import tool may have options to maintain system attributes.

Export Tool in More Detail

A Java class, `XmlExporter`, provides the export functionality. It can be run from the command line, from within Java code, or as an ant task in `grouper/build.xml`:

```
ant xml-export \-Dcmd="command line arguments"
```

The command line usage is:

Command	Summary of args.
args: -h	Prints this message
args:	subjectIdentifier [(-id) [-name)] [-relative] [-includeParent] fileName [properties]

The above export args. can be explained as follows:

Command	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Group or Stem to export.
-name	The name of a Group or Stem to export.
-relative	If id or name specified do not export parent Stems.
-includeParent	If id or name identifies a Stem export this stem and child Stems or Groups.
filename	The file where exported data will be written. Will overwrite existing files.
properties	The name of a standard Java properties file which configures the export. Check Javadoc for a list of properties. If 'properties' is not specified, XmlExporter will look for 'export.properties' in the working directory. If this file does not exist XmlExporter will look on the classpath. If 'properties' is not specified and 'export.properties' cannot be found, the export will fail.

The JavaDoc describes the export methods. Including a method which can be used to export an arbitrary Collection of Stems, Groups, Subjects or Memberships returned by various Grouper API methods. This means that the results of any *list* or *search* methods can be exported.

An XML Schema which describes the exported XML is available [here](#).

If a relative export is performed, the export tool treats group members, list members or privilegees which are groups, and which are *descendants* of the export stem in a special manner. The Subject Identifier, which, for groups, is usually the group name, is modified so that the export stem name is replaced by an asterix, thus, if performing a relative export of uob:artf, a reference to the staff group would become *staff rather than uob:artf:staff. The import tool will replace the asterix with the import stem name. In this way the relationship between groups can be maintained.

Examples of exported data are available [here](#).

Import Tool in More Detail

A Java class, XmlImporter, provides the import functionality. It can be run from the command line, from within Java code, or as an ant task in grouper/build.xml:

```
ant xml-import \-Dcmd="command line arguments"
```

The command line usage is:

Command	Summary of args.
args: -h	Prints this message
args:	subjectIdentifier [(-id -name -list)] filename [properties]

The above import args. can be explained as follows:

Commands	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Stem, into which, data will be imported. If no -id is specified, use=ROOT stem.
-name	The name of a Stem, into which, data will be imported. If no -name is specified, use=ROOT stem.
-list	File contains a flat list of Stems or Groups which may be updated. Missing Stems and Groups are not created.
filename	the file to import
properties	The name of a standard Java properties file which configures the import. Check Javadoc for a list of properties. If 'properties' is not specified, XmlImporter will look for 'import.properties' in the working directory. If this file does not exist XmlImporter will look on the classpath. If 'properties' is not specified and 'import.properties' cannot be found and import options are not included in the XML, the import will fail.

The JavaDoc describes the load methods.

An XML Schema which describes the format of XML which can be loaded is available [here](#).

It is possible to generate an XML file which validates against the schema, but which does not load properly. The annotations in the schema describe appropriate usage of attributes and elements.

The Grouper QuickStart includes a demo registry. [quickstart.xml](#) is a minimal XML import file which creates the demo registry*.

*For Grouper 0.9 and earlier, the demo registry was created by loading an XML file using a contributed XmlLoader class. The format of XML recognised by XmlLoader is significantly different to the *official* format understood by XmlImporter, however, if created files in the old format, these can still be loaded at the Root stem, simply add <data></data> inside the <registry> tags but outside your actual Stem and Group data. You can then export the registry to obtain an XML file in the new format. The XmlLoader format will not be supported in the next Grouper release.

When generating XML in the import format, it is likely that relationships between stems and groups will need to be specified. This is problematic because the uuids of groups and stems are unknown prior to creation. In addition, it is not always possible to know the full name of a new Stem or Group, as this will depend on which stem it is imported into. When importing Subjects that are groups, the import tool examines the identifier attribute and makes any necessary changes before further processing. The following notations are recognised:

Notation	Description
SELF	Refers to the <i>context group</i> for which the Subject is being processed as a member*, list member or privilege. *Actually the API will prevent a Group becoming a member of itself
*	As described above, * is replaced with the name of the Stem where the XML is to be imported
..:	Replace with the name of the Stem which contains the context group.
...:	Replace with the parent Stem of the Stem which contains the context group. May occur multiple times.

 Questions or comments?  [Contact us.](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

v1.0_RC1_Release Notes

This page last changed on Jul 18, 2007 by jbibbee@internet2.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

Grouper v1.0 Release Candidate 1 Software

Grouper v1.0 RC1 is now available. We intend to release the official v1.0 to the Grouper community on Thursday, July 20, provided that the RC1 testing conducted by this community doesn't uncover any serious bugs that can't be fixed by then. Please post bugs and other feedback to the [grouper-users](#) mailing list.

Tarballs

- [QuickStart](#) - a self-contained package, including a demo database, to get a demo up quickly.
- [Grouper API](#) - the java API.
- [Grouper UI](#) - the java UI.



Highlights

- Group math
- Custom group types and attributes
- XML import/export tool
- GrouperShell command line shell - gsh

Caveats

1. The RC1 tarballs contain some older documentation. We haven't finished producing extracts of wiki docs for inclusion in the v1.0 tarballs.
2. The Grouper v1.0 UI embodies our first attempt to enable two tasks that are complicated in a UI context: managing composite groups, and managing custom group types and attributes. We've prepared a brief [tutorial](#) to help smooth your exploration of these new UI capabilities.
3. The QuickStart tarball should work fine on JDK1.5. However, for JDK 1.4 the build.properties for the Grouper UI should be modified so the following line is uncommented:

```
\#grouper.compile.api=true
```

 Questions?  [Contact us](#).

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

v1.0_UI Building and Configuration

This page last changed on Jan 04, 2008 by jbibbee@internet2.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)



This is an archived page. To view the current page, please see [UI Building and Configuration](#). Thanks!

Configuring and Deploying the Grouper UI v1.0

In this section we describe how to configure, build, and deploy the Grouper UI.

Section	Description
Lightweight UI Configuration	Just a few easy bits, leaving in-depth coverage of the UI's extensive customization capabilities to the Grouper UI Guide .
Building & Deploying	All about building and deploying the UI.

Lightweight UI Configuration

In this subsection we'll describe how to replace the logo image included in the Grouper UI tarball and highlight a couple of settings in the `grouper-ui/resources/grouper/media.properties` file that control how the UI uses subject attributes.

Using Your Own Logo Image

1. Place the image file in `grouper-ui/webapp/grouper/images/`.
2. Replace the "image.organisation-logo" property in `grouper-ui/resources/grouper/media.properties` with the name of the file emplaced in the previous step.

Controlling the Use of Subject Attributes in the UI

Subjects are presented in the UI in various contexts. The Grouper UI supports a limited capability to control which subject attributes are displayed in which contexts. Here's a list of associated properties in the `grouper-ui/resources/grouper/media.properties` file and how to use them.

Property Name	Description	Possible Values
<code>subject.default.attribute</code>	The default attribute used to identify any subject. Might be superseded by other configuration declarations.	Any subject attribute common to all subjects presented by source adapters. The minimum set available by default is determined by the [Subject API] . Under Subject API v0.2.1, those values are: name, description, and subjectId.
<code>group.default.attribute</code>	The default attribute used to identify any group. Might	Any group naming attribute: name, displayName, extension, displayExtension, id.

	be superseded by other configuration declarations.	
stem.default.attribute	The default attribute used to identify any stem. Might be superseded by other configuration declarations.	Any stem naming attribute: name, displayName, extension, displayExtension.
search.group.result-field	The name of the group naming attribute displayed in search results and on the "saved groups" page.	Any group naming attribute: name, displayName, extension, displayExtension, id.
search.stem.result-field	The name of the stem naming attribute displayed in search results.	Any stem naming attribute: name, displayName, extension, displayExtension.



In the case of groups or stems displayed in search results, the media properties above only determine defaults. UI users are enabled to change the default for a UI session. The range of options they are presented are given in the following table.

Property Name	Description	Possible Values
search.group.result-field-choice	The set of choices of naming attributes that a UI user is presented with for display of groups in search results. If the set is empty, the user cannot change the default.	Space-separated list of zero or more of any of the group naming attributes: name, displayName, extension, displayExtension, id.
search.stem.result-field-choice	The set of choices of naming attributes that a UI user is presented with for display of stems in search results. If the set is empty, the user cannot change the default.	Space-separated list of zero or more of any of the stem naming attributes: name, displayName, extension, displayExtension.

Building & Deploying

1. **Copy grouper-ui/build.properties.template to grouper-ui/build.properties.**
2. **Review grouper-ui/build.properties.**
 - If you want the build script to automatically install the UI in your Tomcat instance, uncomment and set the appropriate value for deploy.home. If you do not set this you will need to copy the UI to your Tomcat installation's webapps directory. You will probably want to define the default.webapp.folder to suit how you intend to develop or customise the UI. See the [Grouper UI Development Environment](#) for options.
 - Make sure you set the grouper.folder property to the location of your Grouper installation.
3. **Copy grouper-ui/template-tomcat-context.xml to grouper-ui/tomcat-context.xml** (or the value of the property deploy.context.xml if you have changed this).
 - Tomcat specific configuration can be added in this file e.g., container managed data sources.
4. **Change directory to grouper-ui and type "ant".**
 - A list of build targets is displayed. If you have set deploy.home enter "default". Otherwise type "dist" or "war". If the former copy <dist.home>/grouper to <TOMCAT_HOME>/webapps, or if the latter, copy <dist.home>/grouper.war to <TOMCAT_HOME>/webapps.
 - If you want to take advantage of the 'nice' targets you must uncomment and set appropriate values for all the deploy properties in grouper-ui/build.properties.

Note: The build process will attempt to create a directory peer to the grouper-ui directory. Hence, the directory grouper-ui/.. must be writable.

 Questions or comments?  [Contact us.](#)

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Contribute](#) [WG](#) [Contact](#)

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Database Conversion

Grouper v1.1 works with a Grouper v1.0 database just as well as Grouper v1.0 does. However, we found that by reordering the columns in one key table (the `grouper_memberships` table) significant performance gains occurred across several types of operations. We also reordered the columns in a second table (`grouper_members`) to explore for additional gains, but found the change only marginal. But the database schema proper - the tables, their columns, indices, and relationships - did not change at all between the v1.0 and v1.1 releases, which is why Grouper v1.1 works with a v1.0 database.

But to reap that performance gain, the columns in a Grouper database must be reordered. To do that we'll use the tools first released in Grouper v1.0 to support a database conversion, should one be needed. Basically, we'll export the entire database (including its metadata) into an XML file, reset the database, then reimport it, following the steps described below.

A database conversion can take a substantial amount of time, depending of course on how large it is. One on demo HSQLDB database containing 628 stems, 640 groups, and 14,447 memberships and non-default privilege assignments, on a laptop it took 1 minute 18 seconds to export, and 3 minutes 57 seconds to re-import, taking on average about 6ms to create each direct or indirect membership and privilege. Your mileage will vary, but it will take time to convert a substantial database.

Conversion steps

This process assumes that you have the Grouper API v1.1 RC3 or later installed and configured to work with your Grouper database, and that you have a shell open on the root of the Grouper API distribution directory.

Note that release candidates 1 and 2 had a bug with the XML export capability that prevented this process from being successful.

1. **ant xml-export -Dcmd="GrouperSystem aFileName"**

The default xml-export properties are correct for doing a complete dump of the database. The filename can refer to a file in the current directory, or it can be a relative or absolute pathname.

2. **Create a new database container**

Do whatever you have to do to setup a fresh database with your RDBMS technology.

3. **Setup the SA account used by the Grouper API**

Establish the credential used by the Grouper API for database access in your new database.

4. **Review conf/grouper.hibernate.properties**

Ensure that properties declaring how the Grouper API will connect to your database are correctly declared in the `conf/grouper.hibernate.properties` file.

4. **ant schemaexport**

5. **ant db-init**

These two steps create the Grouper v1.1 schema in your new database.

6. **ant xml-import -Dcmd="GrouperSystem theSameFileName"**

This re-loads everything into the new database.

An alternative approach

Since this change is only to the column order in two tables, you can consider using SQL to export and suitably re-import the tables. Details will vary by RDBMS type, but this is likely to be far faster than the conversion described above. The two changes are:

- grouper_members: removed 'member_uuid' from position 5 and inserted at position 2
- grouper_memberships: removed 'membership_uuid' from position 2 and inserted at position 10 (the last column)

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Database Conversion v1.2.0 - v1.2.1

This page last changed on Apr 17, 2008 by [shilen](#).

Database Conversion

Modifications have been made to the indexes on the Grouper tables for the 1.2.1 release of Grouper. Below you will find a description of each change and an example SQL statement for Oracle that you may execute to make the modification in your database.

For your convenience, we have also included SQL scripts that contain all of these index modifications described below.

1. [Oracle SQL Script](#)
2. [MySQL SQL Script](#)
3. [Postgres SQL Script](#)

Modifications to indexes:

1. Drop the concatenated index on grouper_memberships with columns list_name and list_type.

```
drop index membership_field_idx;
```

2. Drop the index on grouper_memberships.owner_id.

```
drop index membership_owner_idx;
```

3. Drop the index on grouper_memberships.mship_type.

```
drop index membership_type_idx;
```

4. Create a concatenated index on grouper_memberships using the columns member_id, list_name, and list_type.

```
create index membership_member_list_idx on grouper_memberships (member_id, list_name, list_type);
```

5. Create a concatenated index on grouper_memberships using the columns owner_id, list_name, list_type, and mship_type.

```
create index membership_owner_list_type_idx on grouper_memberships (owner_id, list_name, list_type, mship_type);
```

Database Conversion

To help the performance of database reads and to help the integrity of the data, we have added additional indexes and constraints on the Grouper database tables. There are two options available to convert a Grouper 1.2.1 database to a Grouper 1.3.0 database.

1. With the first option, you can execute SQL statements that will update your database to the new version. In order to do this, you will have to review the sample SQL statements provided and possibly adjust them depending on your database system. This is a recommended approach for production databases that have large amounts of data and cannot have a downtime.
2. With the second option, you can use Grouper's Export and Import features to export all of your data to an XML file and then import that into a Grouper 1.3.0 database.

Option 1: Database Conversion using SQL

Modifications have been made to the indexes and constraints on the Grouper tables for the 1.3.0 release of Grouper. Below you will find a description of each change and an example SQL statement for Oracle that you may execute to make the modification in your database.



Database Conversion v1.2.0 - v1.2.1

If you originally installed Grouper 1.2.0, you will need to first make some index modifications that were part of the Grouper 1.2.1 release. Note that even if you upgraded from 1.2.0 to 1.2.1 in the past, you likely did not update your indexes. [Please see the following document for more information.](#)

For your convenience, we have also included SQL scripts that contain all of these index and constraint modifications described below. Please review the file and pay close attention to any comments in the file. Also, if you're not using the Subject tables, you can remove them from the script.

1. [Oracle SQL Script](#)
2. [MySQL SQL Script](#)
3. [Postgres SQL Script](#)

Modifications to indexes:

1. Drop the index on grouper_attributes.field_name.

```
drop index attribute_field_idx;
```
2. Create a concatenated index on grouper_attributes using the columns field_name and value.

```
create index attribute_field_value_idx on grouper_attributes (field_name, value);
```
3. Drop the concatenated index on grouper_composites with columns left_factor and right_factor.

```
drop index composite_factor_idx;
```
4. Create an index on grouper_composites.left_factor.

```
create index composite_left_factor_idx on grouper_composites (left_factor);
```
5. Create an index on grouper_composites.right_factor.

```
create index composite_right_factor_idx on grouper_composites (right_factor);
```
6. Drop the index on grouper_memberships.depth.

```
drop index membership_depth_idx;
```
7. Create a concatenated index on grouper_memberships using the columns member_id and via_id.

```
create index membership_member_via_idx on grouper_memberships (member_id, via_id);
```
8. Create an index on grouper_sessions.member_id.


```
create index session_member_idx on grouper_sessions (member_id);
```

9. Drop the index on grouper_memberships.via_id.

```
drop index membership_via_idx;
```

10. Create an index on SubjectAttribute.value.

```
create index subjectattribute_value_idx on SubjectAttribute (value);
```

11. Drop the concatenated index on Subject with columns subjectId and subjectTypeId.

```
drop index subject_idx;
```

12. Drop the index on SubjectAttribute.subjectId.

```
drop index subjectattribute_id_idx;
```

13. Drop the index on SubjectAttribute.name.

```
drop index subjectattribute_key_idx;
```

Modifications to check constraints:

1. Prevent null values in grouper_attributes.group_id.

```
alter table grouper_attributes modify(group_id not null);
```

2. Prevent null values in grouper_memberships.depth.

```
alter table grouper_memberships modify(depth not null);
```

3. Prevent null values in grouper_memberships.membership_uuid.

```
alter table grouper_memberships modify(membership_uuid not null);
```

4. Prevent null values in grouper_memberships.create_time.

```
alter table grouper_memberships modify(create_time not null);
```

5. Prevent null values in the grouper_sessions.session_uuid.

```
alter table grouper_sessions modify(session_uuid not null);
```

Modifications to unique keys:

1. Drop the concatenated key on grouper_attributes with columns group_id, field_name, and value. **The constraint name was originally generated by the database so substitute constraint_name with the actual name.**

```
alter table grouper_attributes drop constraint constraint_name;
```

2. Create a concatenated key on grouper_attributes with columns group_id and field_name.

```
alter table grouper_attributes add (unique (group_id, field_name));
```

3. Create a key on grouper_composites.uuid.

```
alter table grouper_composites add (unique (uuid));
```

4. Create a key on grouper_fields.field_uuid.

```
alter table grouper_fields add (unique (field_uuid));
```

5. Create a concatenated key on grouper_fields with columns name and type.

```
alter table grouper_fields add (unique (name, type));
```

6. Create a key on grouper_groups.uuid.

```
alter table grouper_groups add (unique (uuid));
```

7. Create a key on grouper_members.member_uuid.

```
alter table grouper_members add (unique (member_uuid));
```

8. Create a key on grouper_memberships.membership_uuid.

```
alter table grouper_memberships add (unique (membership_uuid));
```

9. Create a key on grouper_sessions.session_uuid.

```
alter table grouper_sessions add (unique (session_uuid));
```

10. Create a key on grouper_stems.uuid.

```
alter table grouper_stems add (unique (uuid));
```

11. Create a key on grouper_types.type_uuid.

```
alter table grouper_types add (unique (type_uuid));
```

Modifications to foreign keys:

The sample SQL scripts provided above contain the foreign key modifications. However, if you are not using one of the sample SQL scripts, use ant and the build process for the Grouper 1.3.0 release to accomplish this task: **ant addforeignkeys**

Note that you may receive an error indicating that the Subject or the SubjectAttribute table does not exist. This is expected if you modified or deleted one of those tables. The Subject tables are not required for Grouper. They are available as example tables to store Subject data.

Option 2: Database Conversion using Grouper Export and Import

A database conversion using Grouper Export and Import can take a substantial amount of time, depending of course on how large it is. One on demo HSQLDB database containing 628 stems, 640 groups, and 14,447 memberships and non-default privilege assignments, on a laptop it took 1 minute 18 seconds to export, and 3 minutes 57 seconds to re-import, taking on average about 6ms to create each direct or indirect membership and privilege. Your mileage will vary, but it will take time to convert a substantial database.

Conversion steps

This process assumes that you have the Grouper API v1.3.0 RC1 or later installed and configured to work with your Grouper database, and that you have a shell open on the root of the Grouper API distribution directory.

1. **ant xml-export -Dcmd="GrouperSystem aFileName"**

The default xml-export properties are correct for doing a complete dump of the database. The filename can refer to a file in the current directory, or it can be a relative or absolute pathname.

2. **Create a new database container**

Do whatever you have to do to setup a fresh database with your RDBMS technology.

3. **Setup the SA account used by the Grouper API**

Establish the credential used by the Grouper API for database access in your new database.

4. **Review conf/grouper.hibernate.properties**

Ensure that properties declaring how the Grouper API will connect to your database are correctly declared in the conf/grouper.hibernate.properties file.

4. **ant schemaexport**

5. **ant db-init**

These two steps create the Grouper v1.3.0 schema in your new database.

6. **ant xml-import -Dcmd="GrouperSystem theSameFileName"**

This re-loads everything into the new database.

Entity Relationship Diagram For Grouper 1.3.0

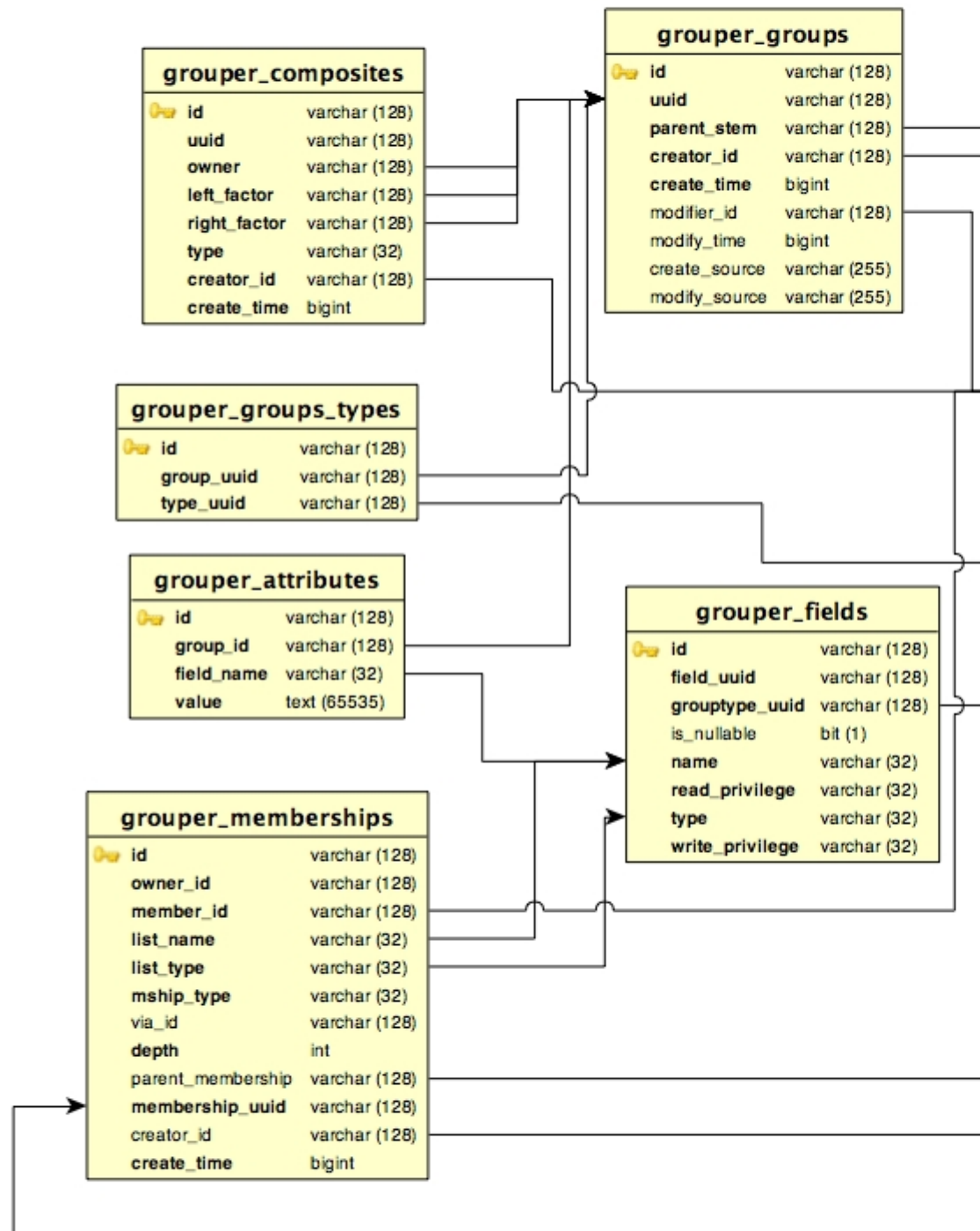
This page last changed on Apr 16, 2008 by [shilen](#).

Entity Relationship Diagram

The following diagram describes the relationships between all the Grouper tables. Also note that grouper_memberships.owner_id is a child of one of the following:

- grouper_groups.uuid for group members and access privileges.

- grouper_stems.uuid for naming



Document generat

privileges.

Grouper change log v1.3

This page last changed on Sep 03, 2008 by [mchzyer](#).

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know.

The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDU, etc. It is assumed if you are running grouper-ui that you will perform both the grouper upgrade notes, and the grouper-ui upgrade notes. These steps start from 2008/05/14, after that date things will be more accurate. It is understood that you will get the new source/javadoc/etc files, this document addresses configurations, jars, etc.

Grouper

- 2008/09/03: Merge the grouper.properties with grouper.example.properties, get the new params about grouperall and groupersystem params
- 2008/09/03: v1.3.1 RC1: Add the following jars:
 - lib/commons-cli-1.1.jar
 - libAnt/ant-contrib-1.0b3.jar
- 2008/05/14: v1.3.0 RC2: Add the following jars:
 - commons-betwixt
 - jakarta-oro
 - jsr107cache-1.0.jar
 - invoker.jar
 - backbort-util-concurrent-3.0.jar
 - cglib2.1_3.jar
 - ehcache-1.4.0.jar
 - asm.jar
 - asm-attrs.jar
 - asm-util.jar
 - hibernate3.2.6.jar
 - p6spy.jar
 - c3p0-0.9.1.2.jar
 - jamon-2.7.jar
 - commons-lang-2.1.jar
 - DdlUtils-1.0.jar
 - activation.jar
 - mailapi.jar
 - smtp.jar
- 2008/05/14: v1.3.0 RC2: Update the following jar: i2mi-common-0.1.0.jar
- 2008/05/14: v1.3.0 RC2: Compare the conf/build.properties for changes. The setting compile.debug was removed (will always compile with debug), add the setting for src.dir.test.conf,
- 2008/05/14: v1.3.0 RC2: Compare the conf/ehcache.xml for changes, some new caches were added
- 2008/05/14: v1.3.0 RC2: Compare the conf/grouper.properties for changes, the wheel group name by default is etc:sysadmingroup, the dao factory is new for hib3 edu.internet2.middleware.grouper.internal.dao.hib3.Hib3DAOFactory, and there are entries and docs for whitelists on db changes from ant
- 2008/05/14: v1.3.0 RC2: Add the new conf/*example* files (not necessary, but easy to show new diffs), e.g. ehcache.example.properties, grouper.hibernate.example.properties, spy.example.properties, README.txt
- 2008/05/14: v1.3.0 RC2: Update the ext dir with new ext-build.xml, and the new extension zips (delete the old extension dirs there)
- 2008/05/14: v1.3.0 RC2: Compare the src/conf/grouper.hibernate.properties files, there are new settings for hib3 and new pooling
- 2008/05/14: v1.3.0 RC2: Compare the src/conf/log4j.properties, the defaults have changed a bit
- 2008/05/14: v1.3.0 RC2: There is a new file: src/conf/grouper.ehcache.xml

Grouper-ui

The nav.properties, media.properties, etc should be edited in localized copies, so these are considered source and are not mentioned here

- 2008/05/14: v1.3.0 RC2: Upgrade the standard.jar

- 2008/05/14: v1.3.0 RC2: There is an additional-build.template.xml, and log4j.template.properties to be used as examples
- 2008/05/14: v1.3.0 RC2: Compare the build.properties.template, there are new settings regarding logging and emails, and remove the debug setting (all compiles are debug enabled)
- 2008/05/14: v1.3.0 RC2: If you use anything from contrib, the build files have changed

Grouper-ws

GSH

- 2008/05/14: v1.3.0 RC2: This is now in CVS in internet2, and is run by a different batch/shell script with a product called invoker.jar

Grouper change log v1.4

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know.

The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDU, etc. It is assumed if you are running grouper-ui that you will perform both the grouper upgrade notes, and the grouper-ui upgrade notes. It is understood that you will get the new source/javadoc/etc files, this document addresses configurations, jars, etc.

Grouper

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH:
 - compare sources.xml with sources.example.xml. Get new section on improved jdbc source. If you are interested in improving the free-form subject search (e.g. from UI), switch to new subject adapter based on example: GrouperJdbcSourceAdapter2
 - compare grouper.properties with grouper.example.properties: get new loader hook and org management section (can be commented out).
 - compare ldappc.xml with ldappc.example.xml, get list-empty-value documentation
- 2009/4/25: v1.4.2 GROUPER_1_4_BRANCH: set this to true in grouper.hibernate.properties: `hibernate.cache.use_query_cache = true`
Compare and merge the new ehcache.example.xml with ehcache.xml. There are stem/group/member caches defined
- 2009/1/25: v1.4.1 GROUPER_1_4_BRANCH: the stem name index was changed to unique from non-unique. Note, if you have problems adding this index, you need to remove stems with duplicate names from the registry. Obviously this needs to be done with care, and you might want to backup before attempting, or contact grouper-dev for help
 - Run: `grouper_home/bin/gsh -registry -check`
 - Then look at the result script, and execute that script, perhaps with:
`grouper_home/bin/gsh -registry -runsqlfile C:/mchyzer/isc/dev/grouper_v1_4/grouper/ddlScripts/grouperDdl_20090125_08_39_44_148.sql`
 - -or- you can do this manually, change the stem name index to unique, and update grouper_ddl table Grouper entry DB_VERSION to 13
 - -or- (not recommended) you could export the registry to backup, rebuild the registry with run:
 - `gsh -xmlexport GrouperSystem /whatever/20090125_1_4.xml`
 - `grouper_home/bin/gsh -registry -drop -runscript`
 - `gsh -xmlimport GrouperSystem /whatever/20090125_1_4.xml`
- 2009/1/25: v1.4.1 GROUPER_1_4_BRANCH: merge grouper.properties with grouper.example.properties, adding properties: `junit.test.loader`, `junit.test.ldappc`, `junit.test.ddl`
- 2008/11/14: v1.4 HEAD: compare and merge all config files with examples, things were rearranged, organized, etc
- 2008/10/29: v1.4 HEAD: update morphString.jar, algorithm changed
- 2008/10/14: v1.4 HEAD: update jars for config checking:
 - delete lib/grouper/commons-discovery-0.4.jar
 - delete lib/grouper/jamon-2.7.jar
 - update invoker.jar
 - add commons-discovery.jar
 - add jamon.jar
 - update morphString.jar
 - update p6spy.jar
 - update subject.jar : note, if you have a custom source, you need to implement some new methods and recompile
- 2008/09/30: v1.4 HEAD: all jars have their version number removed, might want to delete old, and get new. e.g. from quartz-1.6.0.jar to quartz.jar
- 2008/09/23: v1.4 HEAD: [numParameters is now optional in sources.xml](#). You can remove any numParameters params in sources.xml: `<param><param-name>numParameters</param-name><param-value>1</param-value></param>`
- 2008/09/15: v1.4 HEAD: [Subject API converted to use C3P0 pooling instead of DBCP](#).
 - delete: lib/commons-dbc.jar
 - delete: lib/commons-pool.jar
 - update: lib/commons-logging.jar
 - delete: lib/subject-0.3.0-rc1-cvs.jar
 - add: lib/subject.jar

- compare the sources.xml with the sources.example.xml, and add the elements and comments about jdbcConnectionProvider. You can leave the value blank for C3P0, or fill it in. If you are using the same credentials for grouper and a subject source, remove the credentials from sources.xml and put edu.internet2.middleware.grouper.subj.GrouperJdbcConnectionProvider as the jdbcConnectionProvider
- 2008/09/14: v1.4 HEAD: [Passwords encrypted in external files](#). Add the jar: morphString.jar. Add morphString.example.properties, and morphString.properties. In sources.xml, replace the sourceadapter classes
 - FROM: edu.internet2.middleware.subject.provider.JDBCSourceAdapter TO: edu.internet2.middleware.grouper.subj.GrouperJdbcSourceAdapter
 - FROM: edu.internet2.middleware.subject.provider.JNDISourceAdapter TO: edu.internet2.middleware.grouper.subj.GrouperJndiSourceAdapter
 If you want to take advantage of external encrypted passwords (optional):
 - In morphString.properties, set the encrypt.key entry to a random alphanumeric string, or a pathname of a file containing the alphanumeric string
 - In sources.xml, and grouper.hibernate.properties, encrypt the passwords with: java -jar morphString.jar, and put results in a file, and put the file path where the passwords were in sources.xml or grouper.hibernate.properties
- 2008/08/14: v1.4 HEAD: [UUID/ID cols reduced](#), and [Field foreign key is now field id](#). Delete any check constraints you have in your database. E.g. on the grouper_fields table. Start grouper to see in the logs a message that says to run a script, or do an: ant schemaexport. You should review the script before running it, as sometimes ddlutils does weird things. Also, in prod you should export your data before running. Once you are comfortable with the result after running the script and checking your grouper data, set grouper.properties ddlutils.dropBackupUuidCols to true, and ddlutils.dropBackupFieldNameTypeCols to true, then: ant schemaexport, to get the script to drop the backup uuid/id cols. I would run all db scripts in prod with a DB ide (e.g. toad) or manually so that you can address any issues (see link for example issues)
- 2008/07/27: v1.4 HEAD: DDL management changed.
 - Copy the DDL section from the grouper.example.properties into your grouper.properties, configure them
 - Add the jar: ant-1.7.1.jar
 - run: ant schemaexport to sync everything up (or generate the script to sync everything up, depending on grouper.properties)
 - Remove any schema-export.sql files
- 2008/07/22: v1.4 HEAD: If you were already using grouper loader, rename the table grouploader_log to grouper_loader_log. If you already had the grouper_ddl table, delete the java_version column
- 2008/07/21: v1.4 HEAD: Add these jars: quartz-1.6.0.jar, commons-cli-1.1.jar, bsh-2.0b4.jar. GSH, loader, and usdu are in grouper now, with start scripts in grouper_home/bin. See this [jira report](#)
- 2008/07/21: v1.4 HEAD: The java package structure was refactored. If you have code that doesn't compile, organize your imports. See this [jira report](#)
- 2008/07/20: v1.4 HEAD: Compare the grouper.properties with the grouper.example.properties, and add the section regarding group attribute validation via regex
- 2008/07/09: v1.4 HEAD: Everyone needs to tweak your log4j.properties:

Change FROM (remove .ErrorLog and .DebugLog):

```
log4j.logger.edu.internet2.middleware.grouper.ErrorLog      = ERROR, grouper_error
#log4j.logger.edu.internet2.middleware.grouper.DebugLog    = INFO, grouper_debug
```

TO:

```
log4j.logger.edu.internet2.middleware.grouper              = ERROR, grouper_error
#log4j.logger.edu.internet2.middleware.grouper              = INFO, grouper_debug
```

and REMOVE these lines entirely:

```
## Grouper Test Logging
log4j.logger.edu.internet2.middleware.grouper.TestLog      = INFO, grouper_stdout
```

(probably doesn't affect you, but) If you have code that used the classes ErrorLog, DebugLog, or TestLog, these are removed, see GRP-105.

- 2008/07/07: v1.4 HEAD: Merge your grouper.properties with the grouper.example.properties. Pick up the new property: grouper.setters.dont.cause.queries
 - If you set this to true, refactor your the following methods:

- group.setAttribute(), group.setExtension(), group.setDescription(), group.setDisplayExtension() to call group.store() afterwards (could be after multiple calls)
- stem.setDescription(), stem.setExtension(), stem.setDisplayExtension(), to call stem.store() afterwards (could be after multiple calls)
- member.setSubjectSourceId(), member.setSubjectId(), to call member.store() afterwards (could be after multiple calls)

e.g. if the code is:

```
someGroup.setAttribute("whatever", "something");
```

change to:

```
someGroup.setAttribute("whatever", "something");
someGroup.store();
```

- 2008/06/30: v1.4 HEAD: Merge your grouper.properties with the grouper.example.properties. Pick up the new hooks configs, but leave the grouper.setters.dont.cause.queries setting out for now (or set to false).
- 2008/06/05: v1.4 HEAD: Remove the jar: i2mi-common-0.1.0.jar
- 2008/06/05: v1.4 HEAD: Add the jars (note these dont have version numbers since they were extracted from i2mi-common and Im not 100% sure of the version number.. replacements will though):
 - commons-beanutils.jar
 - commons-collections.jar
 - commons-dbc.jar
 - commons-digester.jar
 - commons-logging.jar
 - commons-pool.jar
 - dom4j.jar
 - odmg.jar

Grouper-ui

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare your media.properties with the new media.properties, get new settings
- 2009/04/11: v1.4.2 (GROUPER_UI_V1_4_BRANCH): there is now a grouper-ui.jar, so do a clean build (or delete the deployed WEB-INF/classes/edu dir), and use the new grouper-ui.jar, and the new struts.jar)
- 2008/09/23: v1.4 HEAD: [You can add tooltips to your group types and attributes](#) if you like. Add something like this to the custom nav.properties, though substitute the type name and attribute name for "grouperLoader" and "grouperLoaderdbName"

tooltipTargetted.groupTypes.grouperLoader=Group membership automatically managed via an external source, e.g. SQL query
 tooltipTargetted.groupFields.grouperLoaderDbName=For sql based loader, this is the name of the db connection.

Grouper-ws

- 2009/06/26: v1.4.2: note that 1.4.2 web.xml in grouper-ws was accidentally committed with the authentication stuff commented out. The web services wont work since the user isnt there. You need to use the web.xml of the previous version (or latest 1.4 branch). e.g. [here](#). If you have built from CVS in the 1.4+ branch you shouldnt have this problem.
- 2008/12/4: v1.4.0rc1 HEAD: added WsSubject.identifierLookup. Note, for REST less than 1.4, this field will not be sent. Also, in wsAssignGrouperPrivilegesLiteResult, there was a privilegeResults field, which shouldne be there, it was removed. In wsAssignGrouperPrivilegesLiteResult the privilege type was all caps, it was changed to be all lower
- 2008/12/4: v1.4.0rc1 HEAD: Changed result codes for groupSave and stemSave. Instead of SUCCESS, it will be either: SUCCESS_INSERTED, SUCCESS_UPDATED, SUCCESS_NO_CHANGES_NEEDED. Note, this is only for client version of 1.4+
- 2008/12/2: v1.4.0rc1 HEAD: Added two properties to grouper-ws.properties, compare with the example file
- 2008/11/6: v1.4 HEAD: Added two more fields to the wsGroupDetail object, adjust your clients appropriately:

- compositeType (should be UNION, COMPLEMENT, INTERSECTION, or blank for non-composite)
- params
- 2008/10/28: v1.4 HEAD: some changes in conjunction with adding the privilege services
 - merge grouper-ws build.properties with example, and note the group.lib.dir has changed
 - there was a weird situation where the result code was transmitted as an object...

```
<WsDeleteMemberLiteResult>
  <resultMetadata>
    <wsResultCode
class="edu.internet2.middleware.grouper.ws.soap.WsDeleteMemberLiteResult
$WsDeleteMemberLiteResultCode">SUCCESS</wsResultCode>
  <resultCode>SUCCESS</resultCode>
  <resultMessage></resultMessage>
  <success>T</success>
```

I fixed this so it is not transmitted... clients should not have been using that element, but if so, check them. I believe this was only REST formats...

- 2008/10/21: v1.4 HEAD: [Axis2 1.3 was upgraded to Axis2 1.4](#),
 - you need to get all the lib/axis jars, and delete old dupes
 - replace or merge the axis.xml
 - replace or merge the /services and /modules dirs
- 2008/10/12: v1.4 HEAD: The grouper lib dir changed, so did this property in the grouper-ws.properties (note its not lib/grouper anymore):
 - #e.g. from grouper-ws: lib

Grouper-loader

- 2008/12/12: v1.4.0: Merge grouper-loader.properties with grouper-loader.example.properties, there is a new setting for daily report directory
- 2008/12/9: v1.4.0 RC2 HEAD: if you are using grouper loader, and you are not auto creating types and attributes in grouper config, then add this new attribute with GSH:
 - subj=SubjectFinder.findById("GrouperSystem")
 - sess=GrouperSession.start(subj)
 - type=GroupType.createType(sess, "grouperLoader")
 - read=Privilege.getInstance("read")
 - admin=Privilege.getInstance("admin")
 - type.addAttribute(sess, "grouperLoaderGroupQuery", read, admin, false)
- fd

Grouper client

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare grouper.client.usage.txt with grouper.client.usage.example.txt, fix typos

LDAPPC

- 2009/7/7: v1.4.2 : An error was made in the configuration schema whereby the "grouper-attribute" is required regardless of structure (either "flat" or "bushy") and is no longer restricted to "id" or "name". The "grouper-attribute" should only be required when the structure is flat. The affected schema element in conf/edu/internet2/middleware/ldappc/schema/ldappcConfig.xsd should look like :

```
<!-- ===== GROUPS ELEMENT ===== -->
<xs:element name="groups">
  ...
  <xs:attribute name="grouper-attribute">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="id" />
        <xs:enumeration value="name" />
      </xs:restriction>
```

```
</xs:simpleType>  
</xs:attribute>
```

Grouper change log v1.5

This page last changed on Sep 02, 2009 by [mchzyer](#).

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know.

The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDU, etc. It is assumed if you are running grouper-ui that you will perform both the grouper upgrade notes, and the grouper-ui upgrade notes. It is understood that you will get the new source/javadoc/etc files, this document addresses configurations, jars, etc.

Grouper

- 2009/09/02: v1.5 HEAD: Subject API changed, if you have custom subject sources/Subject implementations, see if they still compile. [Here is the list of changes](#). You should probably subclass SubjectImpl and BaseSourceAdapter. Get the latest subject.jar from the grouper 1.5 lib dir.
- 2009/03/24: v1.5 HEAD: Group attributes changed a bit.
 - If you use them in custom code, make sure your code still compiles, and if you can remove deprecated calls (each deprecated method is overloaded with a new method).
 - You do not need to call group.store() for an attribute change anymore, it is only needed for Group fields (name, displayName, description, etc).
 - If you are making changes to the registry in a hook, you need to not use a finder to find the object you are updating, but rather use the object model provided to you. This will prevent stale state exception, might want to send your hook code to the email list (or just the core grouper developers) for review.
 - If you are using attributes to access core fields (group.getAttributes().get("name")), please refactor so just call group.getName()
 - Non core group fields are not in the group.dbVersionDifferentFields()... you should use an AttributeHook if you need to trigger based on non group core fields (or we can add a high level hook for group edits, discuss with the core team)
- 2009/03/24: v1.5 HEAD: Merge grouper.properties with grouper.example.properties:
 - 2009/03/24: Add in the optimistic locking setting: dao.optimisticLocking = true. If you get stale state object exceptions, you can set this to false, and let the grouper team know (might be due to a custom hook we can advise on tweaking)
 - 2009/03/21: Add the two settings for if group and stem last_membership_change col should be updated (you need it for ldappc or custom things, otherwise it is not needed)
 - 2009/01/02: v1.5 HEAD: compare grouper.properties with grouper.example.properties. After the DDL upgrade is successfully (perhaps a few days later), set this in grouper.properties: ddlutils.dropAttributeBackupTableFromGroupUpgrade = false

Then the next ddl upgrade will delete the backup table BAK_GROUPER_ATTRIBUTES, do this yourself manually, or run: grouper_home/bin/gsh -reregistry -deep, and that will drop the backup table.

- - sdf
- 2009/03/21: v1.5 HEAD: Exception handling has changed so that grouper exceptions are now unchecked. You can look at your custom code including hooks and see if that changes anything (if anything you can optionally not catch grouper exceptions and convert into runtime if you are right now). Also, finders for single objects are deprecated. If you use these in custom code or hooks you can refactor so they use the overload, pass in true as last param so it throw exception if not found. e.g. GroupFinder.findByName()
- 2009/01/28: v1.5 HEAD: the DDL is different, this step will address the following changes:
 - 2009/01/28: there are two new cols in grouper_groups and grouper_members
 - 2009/01/27: the membership table cols owner_id and via_id were expanded to owner_group_id, owner_stem_id, via_group_id, and via_composite_id.
 - 2009/01/02: grouper_groups attributes (name, extension, display_extension, display_name, description) are now columns of grouper_groups table
 - TO UPGRADE:
 - Look through custom code which uses the API (including hooks), and if you have accessed these fields through attributes, then convert to it how accesses via direct getters/setters. e.g. from getAttribute("name") to getName(). or setAttribute("extension") to setExtension(). Including getAttributes() and getAttributesDb()
 - Turn off grouper systems for updates
 - Backup the registry, e.g. with export: gsh -xmlexport GrouperSystem / whatever/20090127_1_5.xml

- Either run the in-place upgrade, or reinstall the registry and import
- [in-place] Run: `grouper_home/bin/gsh -registry -check`
 - Then look at the result script, and execute that script, perhaps with:
`grouper_home/bin/gsh -registry -runsqlfile C:/mchzyer/isc/dev/grouper_v1_4/grouper/ddlScripts/grouperDdl_20090120_08_39_44_148.sql`
- -or- [reinstall] `grouper_home/bin/gsh -registry -drop -runscript`
 - `gsh -xmlimport GrouperSystem /whatever/20090127_1_5.xml`
- 2009/01/27: v1.5 HEAD: merge `grouper.properties` with `grouper.example.properties`, adding property: `ddlutils.dropMembershipBackupColsFromOwnerViaUpgrade`
 - Once the membership col upgrade is done, set this to true and run: `grouper_home/bin/gsh -registry -deep` (and run the resulting script, it will drop the cols `grouper_memberships.owner_id_bak` and `grouper_memberships.via_id_bak`)
- 2009/01/02: v1.5 HEAD: get the latest `junit.jar` from `grouper/lib/grouper`
- 2009/01/02: v1.5 HEAD: . Run this to upgrade: `GROUPER_HOME/bin/gsh -registry -check`
Then review and execute the script per instructions in output. In prod, you should export and import your registry first.
If you are using custom (non builtin) triggers or views, update them

Release Notes

This page last changed on Apr 17, 2008 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Release Notes for Grouper v1.2.1

This page includes the new features and fixes of the current release for the Grouper v1.2.1 API and UI. Download Grouper v1.2.1 on the main [Download](#) page.

Grouper v1.2.1 is a maintenance release in which a few bugs have been fixed and performance has been substantially improved. Some modest functional enhancements have also been made to the UI. Significant gains have been achieved by tuning how the API interacts with the relational back-end and its caching strategy, and both the API and UI use new strategies to check privileges. The project team has also identified some promising new approaches to achieve even further gains that we hope to roll out in Grouper v1.3.0.

Details on the improvements can found in the Internet2 Jira issue tracker for the [API](#) and the [UI](#).

Modifications have been made to the indexes on the Grouper tables for the 1.2.1 release of Grouper. See [Database Conversion v1.2.0 - v1.2.1](#) for a description of each change and an example SQL statement

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

v1.3-Release Notes

This page last changed on Oct 24, 2008 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Release Notes for Grouper v1.3.X

This page includes the new features and fixes for Grouper v1.3.0 and v1.3.1. Download Grouper v1.3.X on the main [Download](#) page.

Grouper v1.3.0 is a significant new release which includes a new and experimental Web Services Interface and an improved user interface. Details on the improvements and outstanding issues for this version can be found in the Internet2 Jira issue tracker for the [API](#), the [UI](#) and the [WS](#)

Note: With release v1.3.0, Grouper requires the ant compiler v1.6.3 or later. It no longer builds correctly using older versions of ant.

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
Grouper v1.3.1	<p><u>Release v1.3.1 includes the following fixes and improvements:</u></p> <ul style="list-style-type: none">• Improved: Directory structure for the API project, as noted in Prerequisites• Fixed: Some circumstances in which memberships are "orphaned", i.e., not deleted but should be.• Added: A command line utility to identify any orphaned memberships.• Fixed: Several UI bugs.• Fixed: Memberships of unresolvable Subjects could not be deleted. <p>The Grouper Jira issues for v1.3.1 shows a detailed list of the issues resolved in v1.3.1.</p>	22-September-2008
Grouper v1.3.0	<p><u>Release v1.3.0 includes the following fixes and improvements:</u></p> <ul style="list-style-type: none">• New: experimental web services interface.• New: extension for removing memberships for unresolvable subjects.• Improved: user interface.• Improved: Hibernate support. Upgraded to Hibernate 3.2.6 and added transaction support	22-May-2008

Hibernate 2.x support is no longer included.

- New: applied transactions to the UI so that a whole action succeeds and is committed, or fails and is rolled back.
- Added: exception handling and logging to the UI.
- Added: ability to disable editing of group attributes / member lists for site configured groups i.e. groups which should be loader maintained.
- Added: foreign keys to database.
- Improved: performance.
- Added: compiled Java in all components includes debug information, which means that stack traces in log files will have line numbers which makes it easier to debug problems.
- Improved: new settings in grouper.properties make it possible to define user / db connection urls which can / cannot have their schemas rebuilt. If not configured the test script will prompt the user to confirm that it is OK to drop a schema. This makes it more difficult to accidentally lose data.
- Improved: transaction handling code will attempt to inject additional information into a caught Exception. If successful the Exception is not logged - assumes that your code will handle logging.
- Added: jsr107cache-1.0.jar - required by default on Solaris.
- Fixed: some CSS issues with IE6.
- Improved: changed the XHTML declaration to use transitional DTD rather than strict - reduces number of errors until we can try to tidy up issues.
- Fixed: Advanced Search link for groups in the UI.
- Added: additional error checking in the UI - whether key properties are set and added option to build the generated client when building the web service.

	<ul style="list-style-type: none"> • Added: example kerberos authentication configuration for the UI. • Added: gsh source to the Internet2 CVS repository and added a page to the Wiki. • Fixed: gsh.bat - was not correctly building the classpath. • Improved: gsh error reporting. Exceptions from Grouper should now be summarised in gsh output and full stack traces written to grouper_error.log. 	
--	--	--

Upgrading from v1.2.1

- v1.3.0 will continue to work well with a v1.2.1 database; however, a new v1.3.0 database has some index optimizations and changes to keys. See [Database Conversion v1.2.1 - v1.3.0](#) for details of these changes and suggestions for upgrading a v1.2.1 database.
- There are [changes to grouper.hibernate.properties](#) to reflect changes in the Hibernate version and the method of connection pooling.
- There are changes to grouper-ui/build.properties.templates and a new file grouper-ui/log4j.template.properties. Review the comments in the files and modify as appropriate.
- By default the UI now has a Logout link. This link will always end the current user session, however, it will not log a user out of HTTP Basic Authentication or some single-sign on systems.
- There have been significant CSS changes to the UI, so it is quite possible that there will be some issues when re-applying customizations as some Grouper CSS will be more specific than custom CSS. The blue vertical bar on the left side of the page is now implemented as a background image on the body tag.
- The UI does not work properly with Internet Explorer 6. We will work to resolve the issue for the final release.

Grouper Change Log

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know.

The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDU, etc. It is assumed if you are running grouper-ui that you will perform both the grouper upgrade notes, and the grouper-ui upgrade notes. These steps start from 2008/05/14, after that date things will be more accurate. It is understood that you will get the new source/javadoc/etc files, this document addresses configurations, jars, etc.

Grouper

- 2008/09/03: Merge the grouper.properties with grouper.example.properties, get the new params about grouperall and grouper system params
- 2008/09/03: v1.3.1 RC1: Add the following jars:
 - lib/commons-cli-1.1.jar
 - libAnt/ant-contrib-1.0b3.jar
- 2008/05/14: v1.3.0 RC2: Add the following jars:
 - commons-betwixt
 - jakarta-oro
 - jsr107cache-1.0.jar
 - invoker.jar
 - backport-util-concurrent-3.0.jar
 - cglib2.1_3.jar
 - ehcache-1.4.0.jar
 - asm.jar
 - asm-attrs.jar
 - asm-util.jar

- hibernate3.2.6.jar
- p6spy.jar
- c3p0-0.9.1.2.jar
- jamon-2.7.jar
- commons-lang-2.1.jar
- DdlUtils-1.0.jar
- activation.jar
- mailapi.jar
- smtp.jar
- 2008/05/14: v1.3.0 RC2: Update the following jar: i2mi-common-0.1.0.jar
- 2008/05/14: v1.3.0 RC2: Compare the conf/build.properties for changes. The setting compile.debug was removed (will always compile with debug), add the setting for src.dir.test.conf,
- 2008/05/14: v1.3.0 RC2: Compare the conf/ehcache.xml for changes, some new caches were added
- 2008/05/14: v1.3.0 RC2: Compare the conf/grouper.properties for changes, the wheel group name by default is etc:sysadmingroup, the dao factory is new for hib3 edu.internet2.middleware.grouper.internal.dao.hib3.Hib3DAOFactory, and there are entries and docs for whitelists on db changes from ant
- 2008/05/14: v1.3.0 RC2: Add the new conf/*example* files (not necessary, but easy to show new diffs), e.g. ehcache.example.properties, grouper.hibernate.example.properties, spy.example.properties, README.txt
- 2008/05/14: v1.3.0 RC2: Update the ext dir with new ext-build.xml, and the new extension zips (delete the old extension dirs there)
- 2008/05/14: v1.3.0 RC2: Compare the src/conf/grouper.hibernate.properties files, there are new settings for hib3 and new pooling
- 2008/05/14: v1.3.0 RC2: Compare the src/conf/log4j.properties, the defaults have changed a bit
- 2008/05/14: v1.3.0 RC2: There is a new file: src/conf/grouper.ehcache.xml

Grouper-ui

The nav.propproperties, media.properties, etc should be edited in localized copies, so these are considered source and are not mentioned here

- 2008/05/14: v1.3.0 RC2: Upgrade the standard.jar
- 2008/05/14: v1.3.0 RC2: There is an additional-build.template.xml, and log4j.template.properties to be used as examples
- 2008/05/14: v1.3.0 RC2: Compare the build.properties.template, there are new settings regarding logging and emails, and remove the debug setting (all compiles are debug enabled)
- 2008/05/14: v1.3.0 RC2: If you use anything from contrib, the build files have changed

Grouper-ws

GSH

- 2008/05/14: v1.3.0 RC2: This is now in CVS in internet2, and is run by a different batch/shell script with a product called invoker.jar

 Questions or comments?  [Contact us.](#)

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

v1.4.0 Getting Started

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Getting started with the Grouper API Binary Distribution v1.4.0

This page contains instructions for installing and configuring a new Grouper v1.4.0 installation.

- Meet required [prerequisites](#) and download the Grouper API binary release
- [API Building & Configuration](#). Config files are in the conf/ directory. Config files ending in ".example" are templates and should not be modified. Config files not ending in ".example" are customizable
- Initialize your database (by default HSQLDB) by running **gsh -registry -check -runscript**

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh -registry -check -runscript
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\conf
Using JAVA:         "c:\dev_inst\java\bin\java"
using MEMORY:      64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\logs\grouper_error.log, at min
level WARN
for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: sa@jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist\run/
grouper;create=true
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider
```

```
Based on grouper.properties: ddlutils.schemaexport.installGrouperData=true
(note, might need to type in your response multiple times (Java stdin is flaky))
(note, you can whitelist or blacklist db urls and users in the grouper.properties)
Are you sure you want to schemaexport all tables (dropThenCreate=F,writeAndRunScript=T) in db user 'sa', db
url 'jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist\run\grouper;create=true'? (y|n):
y
Continuing...
Grouper ddl object type 'Grouper' has dbVersion: 0 and java version: 12
Grouper ddl object type 'Subject' has dbVersion: 0 and java version: 1
Grouper database schema DDL requires updates
(should run script manually and carefully, in sections, verify data before drop statements, backup/export
important data
before starting, follow change log on confluence, dont run exact same script in multiple envs - generate a new
one for
each env),
script file is:
C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\ddlScripts\grouperDdl_20081114_02_00_20_771.sql
Script was executed successfully
```

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>
```

- Verify the installation was successful by examining log files in the logs/ directory.
- When using HSQLDB, a second attempt is sometimes necessary: **gsh -registry -check -runscript**
- It is now possible to run [GrouperShell \(gsh\)](#) and [Initialize Administration of Privileges](#)

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
```

Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\conf
Using JAVA: "c:\dev_inst\java\bin\java"
using MEMORY: 64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: console, at min level WARN for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: sa@jdbc:hsqldb:C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\dist/run/grouper;create=true
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider

2008-11-14 02:12:06,041: [main] WARN ApiConfig.printConfigOnce(189) - Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
Type help() for instructions
gsh 0% addRootStem("penn", "penn");
stem: name='penn' displayName='penn' uuid='b693f8f1-9763-44ae-a984-be25e0b6ea0f'
gsh 1% addGroup("penn", "faculty", "faculty");
group: name='penn:faculty' displayName='penn:faculty' uuid='fbf3646f-eb10-47c1-9cc0-d32b7049a336'
gsh 2%

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

v1.4.1 Upgrade Instructions from v1.3.*

This page last changed on Jan 29, 2009 by [mchalyzer](#).

To upgrade from v1.4.0 grouper to v1.4.1, follow these instructions: (note the [changelog](#))

API / Web Service / User Interface / Extensions

- Ldapppc is now included with grouper, if you are going to use Ldapppc then get the latest binary or source release of grouper, setup the config files similar to how you did for 1.3.*, and kickoff Ldapppc with `gsh -ldapppc`
- If you are upgrading from 1.3.* to 1.4.1, and you have already compared the `grouper.properties` file with `grouper.example.properties`, and run the `ddl` command, then you are done.
- If you upgraded to 1.4.0, and now you are upgrading to 1.4.1, continue:
- Get the latest `grouper.jar`, e.g. from the binary release
- Merge `grouper.properties` with `grouper.example.properties`, adding properties: `junit.test.loader`, `junit.test.ldapppc`, `junit.test.ddl`
- Database conversion. The only change is the index on the name column of `grouper_stems` is now unique. Here are 3 options for upgrade, pick one
 - Run: `grouper_home/bin/gsh -registry -check`
 - Then look at the result script, and execute that script, perhaps with:
`grouper_home/bin/gsh -registry -runsqlfile C:/mchalyzer/isc/dev/grouper_v1_4/grouper/ddlScripts/grouperDdl_20090125_08_39_44_148.sql`
 - -or- you can do this manually, change the stem name index to unique, and update `grouper_ddl` table Grouper entry `DB_VERSION` to 13:
`update grouper_ddl set db_version = 13 where object_name = 'Grouper'`
 - -or- (not recommended) you could export the registry to backup, rebuild the registry with:
 - `gsh -xmlexport GrouperSystem /whatever/20090125_1_4.xml`
 - `grouper_home/bin/gsh -registry -drop -runscript`
 - `gsh -xmlimport GrouperSystem /whatever/20090125_1_4.xml`

Web Services

- Get the latest `grouper-ws.jar` file

Grouper Client

- Merge the `grouper.client.properties` with `grouper.example.properties`. Note new entries for SSL, custom operations, and the new client version

To upgrade from v1.3.* grouper to v1.4, follow these instructions: (note the [changelog](#))

API / Web Service / User Interface / Extensions

- Turn off access to your registry
- Backup your database and export your registry to XML as a backup: (e.g. `ant xml-export -Dcmd="GrouperSystem export.xml"`)
- The libraries (jar file) have largely been renamed and upgraded. Delete all jars specific to grouper, and replace with the new list. You can get the latest jars from the lib subdirs of the binary release
- Merge all of your config files with the new example config files (you can obtain these from the conf directory of the binary release). While merging, read about the new settings, and pick your values.
 - Use a tool such as WinMerge (freeware), or Eclipse to make the task easy. Or, use the new example files, rename, and put your customizations in them
 - Merge `grouper.properties` with `grouper.example.properties`
 - Note that `grouper.properties` was re-organized, so the order of the settings has changed
 - Merge `grouper.hibernate.properties` with `grouper.hibernate.example.properties`
 - Merge `log4j.properties` with `log4j.example.properties`
 - Merge `sources.xml` with `sources.example.xml`

- Note you don't have to have JDBC credentials in this file if they are the same as grouper.hibernate.properties
- Merge ehcache.xml with ehcache.example.xml
- Merge grouper.ehcache.properties with grouper.ehcache.example.properties
- Add grouper-loader.properties
- Add morphString.properties
- If you have a custom subject source, recompile it, there are new methods to implement
- In general recompile any code you have that uses Grouper/Subject references: e.g. package structure has changed, and MemberFinder no longer throws MemberNotFound exception...
- You should probably update your database driver jar... there are examples of recent ones in lib/jdbcSamples... e.g. the error "composite_composite_idx in table grouper_composites references the undefined column "owner"" is fixed by updating the postgres driver

Database conversion (try in test env first) TRACK 1 (do this or track 2)

- This is probably the easier way to go, though possibly more time consuming for large registries. Also, you can try track 2, and if it isn't working for you, then switch to track 1.
 - Note, if you are using HSQL, you should get the newest driver available (e.g. from GROUPER_HOME/lib/jdbcSamples). Upgrade your hsql probably. And do track 1 and not track 2.
- After exporting your data to xml (mentioned above), drop all the objects in your schema. Note, all your UUID's will change
- Run: `gsh -registry -runscript`
- This will create all the tables
- Then import your XML (e.g.):

```
gsh -xmlimport GrouperSystem /opt/grouper/1.3.1rc1/grouper/20081117-1.3.1.xml
```

Database conversion (try in test env first) TRACK 2 (do this or track 1)

Note, if you are using HSQL, you should get the newest driver available (e.g. from GROUPER_HOME/lib/jdbcSamples). Upgrade your hsql probably. And do track 1 and not track 2.

- ◦ This track will convert your data and structures to the new format
- Look in your database (oracle or postgres only, not necessary for mysql) for unique constraints (not unique indices) on grouper tables (not necessary if Oracle). If any exist, remove them
 - For oracle, execute the script generated by this query:

```
select 'alter table ' || table_name || ' drop constraint ' || constraint_name || ';' as the_command from
user_constraints
where constraint_type in ('R', 'U')
and (table_name like 'GROUPER_%' or table_name like 'SUBJECT%')
order by constraint_type, table_name;
```

- ◦ - For postgres, execute the script generated by this query (note, substitute in your schema (public in case below), and catalog (grouper in the case below)

```
SELECT 'alter table ' || table_name || ' drop constraint ' || constraint_name || ';' as
the_command
FROM information_schema.table_constraints where constraint_type in ('FOREIGN KEY',
'UNIQUE')
and lower(table_name) like 'grouper%' and lower(table_schema) = 'public' and
lower(table_catalog) = 'grouper'
order by constraint_type, table_name;
```

- For mysql, it shouldn't be necessary, but here is the script to generate the drop script (change table_schema to match your schema, currently it is grouper):

```
SELECT concat(concat(concat(concat('alter table ', table_name), ' drop foreign key '), constraint_name), ';') as
the_command
FROM information_schema.table_constraints where constraint_type in ('FOREIGN KEY')
and lower(table_name) like 'grouper%' and lower(table_schema) = 'grouper'
order by constraint_type, table_name;
```

- - Run: `gsh -registry -check` (this command is in the bin dir of the binary release... make sure all your config files are configured)
 - Note: Seeing some errors here about tables missing, columns missing, ddl version mismatch is expected.
 - It will give output like this:

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh -registry -check
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\conf
Using JAVA: "c:\dev_inst\java\bin\java"
using MEMORY: 64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\logs\grouper_error.log, at min
level WARN
for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: grouper@jdbc:mysql://localhost:3306/grouper
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider
```

```
Based on grouper.properties: ddlutils.schemaexport.installGrouperData=true
(note, might need to type in your response multiple times (Java stdin is flaky))
(note, you can whitelist or blacklist db urls and users in the grouper.properties)
Are you sure you want to schemaexport all tables (dropThenCreate=F,writeAndRunScript=F) in db user
'grouper', db url 'jd
bc:mysql://localhost:3306/grouper'? (y|n):
y
Continuing...
Grouper ddl object type 'Grouper' has dbVersion: 0 and java version: 12
Grouper ddl object type 'Subject' has dbVersion: 0 and java version: 1
Grouper database schema DDL requires updates
(should run script manually and carefully, in sections, verify data before drop statements, backup/export
important data
before starting, follow change log on confluence, dont run exact same script in multiple envs - generate a new
one for
each env),
script file is:
C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\ddlScripts\grouperDdl_20081114_01_11_45_007.sql
Note: this script was not executed per the grouper.properties: ddlutils.schemaexport.writeAndRunScript
To run script via gsh, carefully review it, then run this:
gsh -registry -runsqlfile C:\dev_inst\workspace_v33\grouper\ddlScripts\
\grouperDdl_20081114_01_11_45_007.sql
```

```
C:\dev_inst\workspace_v33\grouper\bin>
```

- Database conversion (continued)
 - Carefully examine the script (or have a SQL expert do this if you are not one) in that file (in this case: `grouperDdl_20081114_01_11_45_007.sql`)
 - Make sure all operations are ok. e.g. it should not drop any tables, or truncate tables, or anything else that looks bad.
 - Run this script in your DB IDE or grouper can do it with: `gsh -registry -runsqlfile /wherever/filename.sql`
 - **Note: If it has trouble dropping constraints or indexes, you can manually remove those. All the proper indexes (unless you had custom ones) will be recreated. Then continue the script where it left off.**
- Turn on grouper, it should work. Look in the logs, make sure no configuration errors or warnings are thrown. If they are, address them.
- If you are using a source release of grouper, you should get the new source release (including `build.xml`, `build.properties` etc), and copy your properties files into the `/conf` dir. Merge the new `build.example.properties` with your version

- The database conversion script removed some cols, and relinked some foreign keys. The old data is still in the table. After you are satisfied grouper works correctly, you can remove these cols by changing the grouper.properties keys:

```
# after you have converted id's, and are happy with the conversion of removing the uuid col,
# this will remove the backup uuid cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupUuidCols = true
```

```
# after you have converted field id foreign keys, and are happy with the conversion of removing the attribute
name,
# membership list name, and type cols,
# this will remove the backup field name/type cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupFieldNameTypeCols = true
```

- Then generate the DDL again (the -deep option will examine the DB structure, not the DB version from the grouper_ddl table): gsh -registry -deep
 - Then examine the resultant script carefully. Make sure the important operations (e.g. dropping/adding views are innocuous) are dropping the backup columns from the table
 - Turn off access to grouper
 - Backup your registry by database or xml export
 - Run that script
 - Verify the columns are removed
 - Grouper should work

Web services

- Axis jars have changed. Remove the old ones, add new ones from the grouper-ws release in the lib/axis dir
- Nothing important has changed with grouper-ws.properties, but you may want to merge yours with grouper-ws.example.properties to be sure
- Replace or merge the axis.xml with the one from grouper-ws distribution
- Replace or merge the /services and /modules dirs from the grouper-ws distribution
- Two data changes were made to existing services:
 - Added two more fields to the wsGroupDetail object, adjust your clients appropriately (should only matter for SOAP clients or REST which are not coded correctly)
 - compositeType (should be UNION, COMPLEMENT, INTERSECTION, or blank for non-composite)
 - params
 - There was a weird situation where the result code was transmitted as an object...this is not transmitted anymore

```
<WsDeleteMemberLiteResult>
  <resultMetadata>
    <wsResultCode
class="edu.internet2.middleware.grouper.ws.soap.WsDeleteMemberLiteResult
$WsDeleteMemberLiteResultCode">SUCCESS</wsResultCode>
    <resultCode>SUCCESS</resultCode>
    <resultMessage></resultMessage>
    <success>T</success>
```

- Merge grouper-ws build.properties with example, and note the grouper.lib.dir has changed

User Interface

- Merge the grouper nav.properties with yours. Note new properties about tooltips in types and attributes.

v1.4 Release Notes

This page last changed on Jun 04, 2009 by [mchzyer](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Release Notes for Grouper v1.4.2

Grouper v1.4.2 includes 38 fixes and improvements. See the [full list here](#).

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
Grouper v1.4.2	<p><u>Release v1.4.2 includes the following fixes and improvements:</u></p> <ul style="list-style-type: none">• Improved: Better UI performance for large resultsets• Improved: Better caching in the API• Added: New JDBC source adapter with better searching• Added: Ability to manage hierarchical structures with grouper loader (e.g. org lists)• Fixed: Prevent duplicate group names• Improved: Better privilege web service operations• Fixed: Various ldappc, web service, and API bugs	5-June-2009

To upgrade from v1.4.1 grouper to v1.4.2, follow these instructions: (note the [change log](#))

Generally all you need to upgrade from 1.4.1 to 1.4.2 is the new grouper jars: grouper.jar, grouper-ws.jar, etc. You will need to build the UI. There are some config file changes that you should merge so you have up to date documentation in the config files, though all new config file changes should be optional. You can enable caching in the API for improved performance.

Grouper

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH:
 - compare sources.xml with sources.example.xml. Get new section on improved jdbc source. If you are interested in improving the free-form subject search (e.g. from UI), switch to new subject adapter based on example: GrouperJdbcSourceAdapter2
 - compare grouper.properties with grouper.example.properties: get new loader hook and org management section (can be commented out).
 - compare ldappc.xml with ldappc.example.xml, get list-empty-value documentation
- 2009/4/25: v1.4.2 GROUPER_1_4_BRANCH: set this to true in grouper.hibernate.properties:
hibernate.cache.use_query_cache = true

Compare and merge the new ehcache.example.xml with ehcache.xml. There are stem/group/member caches defined

Grouper-ui

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare your media.properties with the new media.properties, get new settings
- 2009/04/11: v1.4.2 (GROUPER_UI_V1_4_BRANCH): there is now a grouper-ui.jar, so do a clean build (or delete the deployed WEB-INF/classes/edu dir), and use the new grouper-ui.jar, and the new struts.jar)

Grouper client

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare grouper.client.usage.txt with grouper.client.usage.example.txt, fix typos

Release Notes for Grouper v1.4.1

Grouper v1.4.1 now includes Ldapdc, and also has some minor fixes and improvements. See the [full list here](#).

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
Grouper v1.4.1	<p>Release v1.4.1 includes the following fixes and improvements:</p> <ul style="list-style-type: none">• Improved: Ldapdc is included in the Grouper API and is launched via Grouper Shell• Improved: WS: replacing existing members and actAs• Improved: Prevent duplicate stem names• Fixed: Replace hsql file mode with server mode in default grouper configs	2-February-2009

Release Notes for Grouper v1.4.0

Grouper v1.4.0 has substantial new integration, automation, and diagnostic capabilities, enabling deployers to meet more requirements with less implementation effort. This page highlights some of the new features and fixes. Download Grouper v1.4.0 on the main [Download](#) page. Further below are [instructions for upgrading](#) from version 1.3.X and the [log of changes](#) from v1.3.1 to v1.4.0.

Details on the improvements and outstanding issues for this version can be found in the Internet2 Jira issue tracker for the [API](#), the [UI](#), and the [WS](#).

Note: With release v1.4.0, building Grouper from source requires the ant compiler v1.7.0 or later.

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
---------	--	------

Grouper v1.4.0	<p>Release v1.4.0 includes the following fixes and improvements:</p> <ul style="list-style-type: none"> • New: Programmatic hooks for local extensions to Grouper. • New: Automatically maintain groups and memberships from external sources with Grouper Loader. • Added: A binary form of the Grouper Toolkit. • Improved: Grouper Shell, a comprehensive command line interface to Grouper and its allied utilities, is now a core part of the Grouper distribution. And there are correspondingly fewer ant targets in the source distribution. • Improved: Transaction support to ensure relational consistency of logical group operations. • Improved: XML import and export capabilities are incorporated into Grouper Shell. • New: Group type to automatically create include & exclude groups. • New: Configuration directives that provide more granular system administration and security. • New: Ability to merge two Members formerly thought to be distinct. • New: Regex-style validation of group attributes. • Improved: Faster, more efficient Grouper relational database schema. • New: Configuration checking, daily health report, views, and other diagnostic aids. • Fixed: Circumstances in which composites and circular membership loops could produce incorrect membership information. • Improved: API internal design and packaging of classes. • Improved: UI tooltips for group types and custom attributes. 	4-January-2009
-----------------------	---	-----------------------

- | | | |
|--|--|--|
| | <ul style="list-style-type: none"> Improved: Support for group details and privileges via Grouper Web Services. | |
|--|--|--|

Upgrading from v1.3.X

To upgrade from v1.4.0 grouper to v1.4.1, follow these instructions: (note the [changelog](#))

API / Web Service / User Interface / Extensions

- Ldapppc is now included with grouper, if you are going to use ldapppc then get the latest binary or source release of grouper, setup the config files similar to how you did for 1.3.*, and kickoff ldapppc with `gsh -ldappc`
- If you are upgrading from 1.3.* to 1.4.1, and you have already compared the `grouper.properties` file with `grouper.example.properties`, and run the `ddl` command, then you are done.
- If you upgraded to 1.4.0, and now you are upgrading to 1.4.1, continue:
- Get the latest `grouper.jar`, e.g. from the binary release
- Merge `grouper.properties` with `grouper.example.properties`, adding properties: `junit.test.loader`, `junit.test.ldapppc`, `junit.test.ddl`
- Database conversion. The only change is the index on the `name` column of `grouper_stems` is now unique. Here are 3 options for upgrade, pick one
 - Run: `grouper_home/bin/gsh -registry -check`
 - Then look at the result script, and execute that script, perhaps with:
`grouper_home/bin/gsh -registry -runsqlfile C:/mchzyer/isc/dev/grouper_v1_4/grouper/ddlScripts/grouperDdl_20090125_08_39_44_148.sql`
 - or- you can do this manually, change the stem name index to unique, and update `grouper_ddl` table `Grouper` entry `DB_VERSION` to 13:
`update grouper_ddl set db_version = 13 where object_name = 'Grouper'`
 - or- (not recommended) you could export the registry to backup, rebuild the registry with:
 - `gsh -xmlexport GrouperSystem /whatever/20090125_1_4.xml`
 - `grouper_home/bin/gsh -registry -drop -runscript`
 - `gsh -xmlimport GrouperSystem /whatever/20090125_1_4.xml`

Web Services

- Get the latest `grouper-ws.jar` file

Grouper Client

- Merge the `grouper.client.properties` with `grouper.example.properties`. Note new entries for SSL, custom operations, and the new client version

To upgrade from v1.3.* grouper to v1.4, follow these instructions: (note the [changelog](#))

API / Web Service / User Interface / Extensions

- Turn off access to your registry
- Backup your database and export your registry to XML as a backup: (e.g. `ant xml-export -Dcmd="GrouperSystem export.xml"`)
- The libraries (jar file) have largely been renamed and upgraded. Delete all jars specific to grouper, and replace with the new list. You can get the latest jars from the `lib` subdirs of the binary release
- Merge all of your config files with the new example config files (you can obtain these from the `conf` directory of the binary release). While merging, read about the new settings, and pick your values.
 - Use a tool such as WinMerge (freeware), or Eclipse to make the task easy. Or, use the new example files, rename, and put your customizations in them
 - Merge `grouper.properties` with `grouper.example.properties`

- Note that grouper.properties was re-organized, so the order of the settings has changed
 - Merge grouper.hibernate.properties with grouper.hibernate.example.properties
 - Merge log4j.properties with log4j.example.properties
 - Merge sources.xml with sources.example.xml
 - Note you dont have to have JDBC credentials in this file if they are the same as grouper.hibernate.properties
 - Merge ehcache.xml with ehcache.example.xml
 - Merge grouper.ehcache.properties with grouper.ehcache.example.properties
 - Add grouper-loader.properties
 - Add morphString.properties
- If you have a custom subject source, recompile it, there are new methods to implement
- In general recompile any code you have that uses Grouper/Subject references: e.g. package structure has changed, and MemberFinder no longer throws MemberNotFound exception...
- You should probably update your database driver jar... there are examples of recent ones in lib/jdbcSamples... e.g. the error "composite_composite_idx in table grouper_composites references the undefined column "owner"" is fixed by updating the postgres driver

Database conversion (try in test env first) TRACK 1 (do this or track 2)

- This is probably the easier way to go, though possibly more time consuming for large registries. Also, you can try track 2, and if it isnt working for you, then switch to track 1.
 - Note, if you are using HSQL, you should get the newest driver available (e.g. from GROUPER_HOME/lib/jdbcSamples). Upgrade your hsql probably. And do track 1 and not track 2.
- After exporting your data to xml (mentioned above), drop all the objects in your schema. Note, all your UUID's will change
- Run: gsh -registry -runscript
- This will create all the tables
- Then import your XML (e.g.):

```
gsh -xmlimport GrouperSystem /opt/grouper/1.3.1rc1/grouper/20081117-1.3.1.xml
```

Database conversion (try in test env first) TRACK 2 (do this or track 1)

Note, if you are using HSQL, you should get the newest driver available (e.g. from GROUPER_HOME/lib/jdbcSamples). Upgrade your hsql probably. And do track 1 and not track 2.

- This track will convert your data and structures to the new format
 - Look in your database (oracle or postgres only, not necessary for mysql) for unique constraints (not unique indices) on grouper tables (not necessary if Oracle). If any exist, remove them
 - For oracle, execute the script generated by this query:

```
select 'alter table ' || table_name || ' drop constraint ' || constraint_name || ';' as the_command from
user_constraints
where constraint_type in ('R', 'U')
and (table_name like 'GROUPER_%' or table_name like 'SUBJECT%')
order by constraint_type, table_name;
```

- For postgres, execute the script generated by this query (note, substitute in your schema (public in case below), and catalog (grouper in the case below)

```
SELECT 'alter table ' || table_name || ' drop constraint ' || constraint_name || ';' as
the_command
FROM information_schema.table_constraints where constraint_type in ('FOREIGN KEY',
'UNIQUE')
and lower(table_name) like 'grouper%' and lower(table_schema) = 'public' and
lower(table_catalog) = 'grouper'
order by constraint_type, table_name;
```

- For mysql, it shouldnt be necessary, but here is the script to generate the drop script (change table_schema to match your schema, currently it is grouper):

```
SELECT concat(concat(concat(concat('alter table ', table_name), ' drop foreign key '), constraint_name), ';') as
the_command
FROM information_schema.table_constraints where constraint_type in ('FOREIGN KEY')
```

and lower(table_name) like 'grouper%' and lower(table_schema) = 'grouper'
order by constraint_type, table_name;

- - Run: gsh -registry -check (this command is in the bin dir of the binary release... make sure all your config files are configured)
 - Note: Seeing some errors here about tables missing, columns missing, ddl version mismatch is expected.
 - It will give output like this:

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>gsh -registry -check
Using GROUPER_HOME: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..
Using GROUPER_CONF: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\conf
Using JAVA:         "c:\dev_inst\java\bin\java"
using MEMORY:       64m-512m
Grouper starting up: version: 1.4.0 build date: 2008/11/13 14:42:25
grouper.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.properties
Grouper current directory is: C:\dev_inst\eclipse\workspace_v33\grouper\bin
log4j.properties read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\log4j.properties
Grouper is logging to file: C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\logs\grouper_error.log, at min
level WARN
for package: edu.internet2.middleware.grouper, based on log4j.properties
grouper.hibernate.properties: C:\dev_inst\eclipse\workspace_v33\grouper\conf\grouper.hibernate.properties
grouper.hibernate.properties: grouper@jdbc:mysql://localhost:3306/grouper
sources.xml read from: C:\dev_inst\eclipse\workspace_v33\grouper\conf\sources.xml
sources.xml grouper source id: g:gsa
sources.xml jdbc source id: jdbc: GrouperJdbcConnectionProvider
```

```
Based on grouper.properties: ddlutils.schemaexport.installGrouperData=true
(note, might need to type in your response multiple times (Java stdin is flaky))
(note, you can whitelist or blacklist db urls and users in the grouper.properties)
Are you sure you want to schemaexport all tables (dropThenCreate=F,writeAndRunScript=F) in db user
'grouper', db url 'jd
bc:mysql://localhost:3306/grouper'? (y|n):
y
Continuing...
Grouper ddl object type 'Grouper' has dbVersion: 0 and java version: 12
Grouper ddl object type 'Subject' has dbVersion: 0 and java version: 1
Grouper database schema DDL requires updates
(should run script manually and carefully, in sections, verify data before drop statements, backup/export
important data
before starting, follow change log on confluence, dont run exact same script in multiple envs - generate a new
one for
each env),
script file is:
C:\dev_inst\eclipse\workspace_v33\grouper\bin\..\ddlScripts\grouperDdl_20081114_01_11_45_007.sql
Note: this script was not executed per the grouper.properties: ddlutils.schemaexport.writeAndRunScript
To run script via gsh, carefully review it, then run this:
gsh -registry -runsqlfile C:\dev_inst\workspace_v33\grouper\ddlScripts\
\grouperDdl_20081114_01_11_45_007.sql
```

```
C:\dev_inst\eclipse\workspace_v33\grouper\bin>
```

- Database conversion (continued)
 - Carefully examine the script (or have a SQL expert do this if you are not one) in that file (in this case: grouperDdl_20081114_01_11_45_007.sql)
 - Make sure all operations are ok. e.g. it should not drop any tables, or truncate tables, or anything else that looks bad.
 - Run this script in your DB IDE or grouper can do it with: gsh -registry -runsqlfile /wherever/filename.sql
 - **Note: If it has trouble dropping constraints or indexes, you can manually remove those. All the proper indexes (unless you had custom ones) will be recreated. Then continue the script where it left off.**
- Turn on grouper, it should work. Look in the logs, make sure no configuration errors or warnings are thrown. If they are, address them.

- If you are using a source release of grouper, you should get the new source release (including build.xml, build.properties etc), and copy your properties files into the /conf dir. Merge the new build.example.properties with your version
- The database conversion script removed some cols, and relinked some foreign keys. The old data is still in the table. After you are satisfied grouper works correctly, you can remove these cols by changing the grouper.properties keys:

```
# after you have converted id's, and are happy with the conversion of removing the uuid col,
# this will remove the backup uuid cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupUuidCols = true
```

```
# after you have converted field id foreign keys, and are happy with the conversion of removing the attribute
name,
# membership list name, and type cols,
# this will remove the backup field name/type cols when running the gsh command: registryInitializeSchema()
ddlutils.dropBackupFieldNameTypeCols = true
```

- Then generate the DDL again (the -deep option will examine the DB structure, not the DB version from the grouper_ddl table): gsh -registry -deep
 - Then examine the resultant script carefully. Make sure the important operations (e.g. dropping/adding views are innocuous) are dropping the backup columns from the table
 - Turn off access to grouper
 - Backup your registry by database or xml export
 - Run that script
 - Verify the columns are removed
 - Grouper should work

Web services

- Axis jars have changed. Remove the old ones, add new ones from the grouper-ws release in the lib/axis dir
- Nothing important has changed with grouper-ws.properties, but you may want to merge yours with grouper-ws.example.properties to be sure
- Replace or merge the axis.xml with the one from grouper-ws distribution
- Replace or merge the /services and /modules dirs from the grouper-ws distribution
- Two data changes were made to existing services:
 - Added two more fields to the wsGroupDetail object, adjust your clients appropriately (should only matter for SOAP clients or REST which are not coded correctly)
 - compositeType (should be UNION, COMPLEMENT, INTERSECTION, or blank for non-composite)
 - params
 - There was a weird situation where the result code was transmitted as an object...this is not transmitted anymore

```
<WsDeleteMemberLiteResult>
  <resultMetadata>
    <wsResultCode
class="edu.internet2.middleware.grouper.ws.soap.WsDeleteMemberLiteResult
$WsDeleteMemberLiteResultCode">SUCCESS</wsResultCode>
    <resultCode>SUCCESS</resultCode>
    <resultMessage></resultMessage>
    <success>T</success>
```

- Merge grouper-ws build.properties with example, and note the group.lib.dir has changed

User Interface

- Merge the grouper nav.properties with yours. Note new properties about tooltips in types and attributes.

Grouper Change Log

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know.

The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDU, etc. It is assumed if you are running grouper-ui that you will perform both the grouper upgrade notes, and the grouper-ui upgrade notes. It is understood that you will get the new source/javadoc/etc files, this document addresses configurations, jars, etc.

Grouper

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH:
 - compare sources.xml with sources.example.xml. Get new section on improved jdbc source. If you are interested in improving the free-form subject search (e.g. from UI), switch to new subject adapter based on example: GrouperJdbcSourceAdapter2
 - compare grouper.properties with grouper.example.properties: get new loader hook and org management section (can be commented out).
 - compare ldappc.xml with ldappc.example.xml, get list-empty-value documentation
- 2009/4/25: v1.4.2 GROUPER_1_4_BRANCH: set this to true in grouper.hibernate.properties:
hibernate.cache.use_query_cache = true
Compare and merge the new ehcache.example.xml with ehcache.xml. There are stem/group/member caches defined
- 2009/1/25: v1.4.1 GROUPER_1_4_BRANCH: the stem name index was changed to unique from non-unique. Note, if you have problems adding this index, you need to remove stems with duplicate names from the registry. Obviously this needs to be done with care, and you might want to backup before attempting, or contact grouper-dev for help
 - Run: grouper_home/bin/gsh -registry -check
 - Then look at the result script, and execute that script, perhaps with:
grouper_home/bin/gsh -registry -runsqlfile C:/mchyzier/isc/dev/grouper_v1_4/grouper/ddlScripts/grouperDdl_20090125_08_39_44_148.sql
 - -or- you can do this manually, change the stem name index to unique, and update grouper_ddl table Grouper entry DB_VERSION to 13
 - -or- (not recommended) you could export the registry to backup, rebuild the registry with run:
 - gsh -xmlexport GrouperSystem /whatever/20090125_1_4.xml
 - grouper_home/bin/gsh -registry -drop -runscript
 - gsh -xmlimport GrouperSystem /whatever/20090125_1_4.xml
- 2009/1/25: v1.4.1 GROUPER_1_4_BRANCH: merge grouper.properties with grouper.example.properties, adding properties: junit.test.loader, junit.test.ldappc, junit.test.ddl
- 2008/11/14: v1.4 HEAD: compare and merge all config files with examples, things were rearranged, organized, etc
- 2008/10/29: v1.4 HEAD: update morphString.jar, algorithm changed
- 2008/10/14: v1.4 HEAD: update jars for config checking:
 - delete lib/grouper/commons-discovery-0.4.jar
 - delete lib/grouper/jamon-2.7.jar
 - update invoker.jar
 - add commons-discovery.jar
 - add jamon.jar
 - update morphString.jar
 - update p6spy.jar
 - update subject.jar : note, if you have a custom source, you need to implement some new methods and recompile
- 2008/09/30: v1.4 HEAD: all jars have their version number removed, might want to delete old, and get new. e.g. from quartz-1.6.0.jar to quartz.jar
- 2008/09/23: v1.4 HEAD: [numParameters is now optional in sources.xml](#). You can remove any numParameters params in sources.xml: `<param><param-name>numParameters</param-name><param-value>1</param-value></param>`
- 2008/09/15: v1.4 HEAD: [Subject API converted to use C3P0 pooling instead of DBCP](#).
 - delete: lib/commons-dbc.jar
 - delete: lib/commons-pool.jar
 - update: lib/commons-logging.jar
 - delete: lib/subject-0.3.0-rc1-cvs.jar
 - add: lib/subject.jar

- compare the sources.xml with the sources.example.xml, and add the elements and comments about jdbcConnectionProvider. You can leave the value blank for C3P0, or fill it in. If you are using the same credentials for grouper and a subject source, remove the credentials from sources.xml and put edu.internet2.middleware.grouper.subj.GrouperJdbcConnectionProvider as the jdbcConnectionProvider
- 2008/09/14: v1.4 HEAD: [Passwords encrypted in external files](#). Add the jar: morphString.jar. Add morphString.example.properties, and morphString.properties. In sources.xml, replace the sourceadapter classes
 - FROM: edu.internet2.middleware.subject.provider.JDBCSourceAdapter TO: edu.internet2.middleware.grouper.subj.GrouperJdbcSourceAdapter
 - FROM: edu.internet2.middleware.subject.provider.JNDISourceAdapter TO: edu.internet2.middleware.grouper.subj.GrouperJndiSourceAdapter
 If you want to take advantage of external encrypted passwords (optional):
 - In morphString.properties, set the encrypt.key entry to a random alphanumeric string, or a pathname of a file containing the alphanumeric string
 - In sources.xml, and grouper.hibernate.properties, encrypt the passwords with: java -jar morphString.jar, and put results in a file, and put the file path where the passwords were in sources.xml or grouper.hibernate.properties
- 2008/08/14: v1.4 HEAD: [UUID/ID cols reduced](#), and [Field foreign key is now field_id](#). Delete any check constraints you have in your database. E.g. on the grouper_fields table. Start grouper to see in the logs a message that says to run a script, or do an: ant schemaexport. You should review the script before running it, as sometimes ddlutils does weird things. Also, in prod you should export your data before running. Once you are comfortable with the result after running the script and checking your grouper data, set grouper.properties ddlutils.dropBackupUuidCols to true, and ddlutils.dropBackupFieldNameTypeCols to true, then: ant schemaexport, to get the script to drop the backup uuid/id cols. I would run all db scripts in prod with a DB ide (e.g. toad) or manually so that you can address any issues (see link for example issues)
- 2008/07/27: v1.4 HEAD: DDL management changed.
 - Copy the DDL section from the grouper.example.properties into your grouper.properties, configure them
 - Add the jar: ant-1.7.1.jar
 - run: ant schemaexport to sync everything up (or generate the script to sync everything up, depending on grouper.properties)
 - Remove any schema-export.sql files
- 2008/07/22: v1.4 HEAD: If you were already using grouper loader, rename the table grouploader_log to grouper_loader_log. If you already had the grouper_ddl table, delete the java_version column
- 2008/07/21: v1.4 HEAD: Add these jars: quartz-1.6.0.jar, commons-cli-1.1.jar, bsh-2.0b4.jar. GSH, loader, and usdu are in grouper now, with start scripts in grouper_home/bin. See this [jira report](#)
- 2008/07/21: v1.4 HEAD: The java package structure was refactored. If you have code that doesn't compile, organize your imports. See this [jira report](#)
- 2008/07/20: v1.4 HEAD: Compare the grouper.properties with the grouper.example.properties, and add the section regarding group attribute validation via regex
- 2008/07/09: v1.4 HEAD: Everyone needs to tweak your log4j.properties:

Change FROM (remove .ErrorLog and .DebugLog):

```
log4j.logger.edu.internet2.middleware.grouper.ErrorLog      = ERROR, grouper_error
#log4j.logger.edu.internet2.middleware.grouper.DebugLog    = INFO, grouper_debug
```

TO:

```
log4j.logger.edu.internet2.middleware.grouper              = ERROR, grouper_error
#log4j.logger.edu.internet2.middleware.grouper              = INFO, grouper_debug
```

and REMOVE these lines entirely:

```
## Grouper Test Logging
log4j.logger.edu.internet2.middleware.grouper.TestLog      = INFO, grouper_stdout
```

(probably doesn't affect you, but) If you have code that used the classes ErrorLog, DebugLog, or TestLog, these are removed, see GRP-105.

- 2008/07/07: v1.4 HEAD: Merge your grouper.properties with the grouper.example.properties. Pick up the new property: grouper.setters.dont.cause.queries
 - If you set this to true, refactor your the following methods:

- group.setAttribute(), group.setExtension(), group.setDescription(), group.setDisplayExtension() to call group.store() afterwards (could be after multiple calls)
- stem.setDescription(), stem.setExtension(), stem.setDisplayExtension(), to call stem.store() afterwards (could be after multiple calls)
- member.setSubjectSourceId(), member.setSubjectId(), to call member.store() afterwards (could be after multiple calls)

e.g. if the code is:

```
someGroup.setAttribute("whatever", "something");
```

change to:

```
someGroup.setAttribute("whatever", "something");
someGroup.store();
```

- 2008/06/30: v1.4 HEAD: Merge your grouper.properties with the grouper.example.properties. Pick up the new hooks configs, but leave the grouper.setters.dont.cause.queries setting out for now (or set to false).
- 2008/06/05: v1.4 HEAD: Remove the jar: i2mi-common-0.1.0.jar
- 2008/06/05: v1.4 HEAD: Add the jars (note these dont have version numbers since they were extracted from i2mi-common and Im not 100% sure of the version number.. replacements will though):
 - commons-beanutils.jar
 - commons-collections.jar
 - commons-dbc.jar
 - commons-digester.jar
 - commons-logging.jar
 - commons-pool.jar
 - dom4j.jar
 - odmg.jar

Grouper-ui

- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare your media.properties with the new media.properties, get new settings
- 2009/04/11: v1.4.2 (GROUPER_UI_V1_4_BRANCH): there is now a grouper-ui.jar, so do a clean build (or delete the deployed WEB-INF/classes/edu dir), and use the new grouper-ui.jar, and the new struts.jar)
- 2008/09/23: v1.4 HEAD: [You can add tooltips to your group types and attributes](#) if you like. Add something like this to the custom nav.properties, though substitute the type name and attribute name for "grouperLoader" and "grouperLoaderdbName"

tooltipTargetted.groupTypes.grouperLoader=Group membership automatically managed via an external source, e.g. SQL query
 tooltipTargetted.groupFields.grouperLoaderDbName=For sql based loader, this is the name of the db connection.

Grouper-ws

- 2009/06/26: v1.4.2: note that 1.4.2 web.xml in grouper-ws was accidentally committed with the authentication stuff commented out. The web services wont work since the user isnt there. You need to use the web.xml of the previous version (or latest 1.4 branch). e.g. [here](#). If you have built from CVS in the 1.4+ branch you shouldnt have this problem.
- 2008/12/4: v1.4.0rc1 HEAD: added WsSubject.identifierLookup. Note, for REST less than 1.4, this field will not be sent. Also, in wsAssignGrouperPrivilegesLiteResult, there was a privilegeResults field, which shouldne be there, it was removed. In wsAssignGrouperPrivilegesLiteResult the privilege type was all caps, it was changed to be all lower
- 2008/12/4: v1.4.0rc1 HEAD: Changed result codes for groupSave and stemSave. Instead of SUCCESS, it will be either: SUCCESS_INSERTED, SUCCESS_UPDATED, SUCCESS_NO_CHANGES_NEEDED. Note, this is only for client version of 1.4+
- 2008/12/2: v1.4.0rc1 HEAD: Added two properties to grouper-ws.properties, compare with the example file
- 2008/11/6: v1.4 HEAD: Added two more fields to the wsGroupDetail object, adjust your clients appropriately:

- compositeType (should be UNION, COMPLEMENT, INTERSECTION, or blank for non-composite)
- params
- 2008/10/28: v1.4 HEAD: some changes in conjunction with adding the privilege services
 - merge grouper-ws build.properties with example, and note the group.lib.dir has changed
 - there was a weird situation where the result code was transmitted as an object...

```
<WsDeleteMemberLiteResult>
  <resultMetadata>
    <wsResultCode
class="edu.internet2.middleware.grouper.ws.soap.WsDeleteMemberLiteResult
$WsDeleteMemberLiteResultCode">SUCCESS</wsResultCode>
  <resultCode>SUCCESS</resultCode>
  <resultMessage></resultMessage>
  <success>T</success>
```

I fixed this so it is not transmitted... clients should not have been using that element, but if so, check them. I believe this was only REST formats...

- 2008/10/21: v1.4 HEAD: [Axis2 1.3 was upgraded to Axis2 1.4](#),
 - you need to get all the lib/axis jars, and delete old dupes
 - replace or merge the axis.xml
 - replace or merge the /services and /modules dirs
- 2008/10/12: v1.4 HEAD: The grouper lib dir changed, so did this property in the grouper-ws.properties (note its not lib/grouper anymore):
 - #e.g. from grouper-ws: lib

Grouper-loader

- 2008/12/12: v1.4.0: Merge grouper-loader.properties with grouper-loader.example.properties, there is a new setting for daily report directory
- 2008/12/9: v1.4.0 RC2 HEAD: if you are using grouper loader, and you are not auto creating types and attributes in grouper config, then add this new attribute with GSH:
 - subj=SubjectFinder.findById("GrouperSystem")
 - sess=GrouperSession.start(subj)
 - type=GroupType.createType(sess, "grouperLoader")
 - read=Privilege.getInstance("read")
 - admin=Privilege.getInstance("admin")
 - type.addAttribute(sess, "grouperLoaderGroupQuery", read, admin, false)
- fd

Grouper client



- 2009/6/4: v1.4.2 GROUPER_1_4_BRANCH: compare grouper.client.usage.txt with grouper.client.usage.example.txt, fix typos

LDAPPC

- 2009/7/7: v1.4.2 : An error was made in the configuration schema whereby the "grouper-attribute" is required regardless of structure (either "flat" or "bushy") and is no longer restricted to "id" or "name". The "grouper-attribute" should only be required when the structure is flat. The affected schema element in conf/edu/internet2/middleware/ldappc/schema/ldappcConfig.xsd should look like :

```
<!-- ===== GROUPS ELEMENT ===== -->
<xs:element name="groups">
  ...
  <xs:attribute name="grouper-attribute">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="id" />
        <xs:enumeration value="name" />
      </xs:restriction>
```

```
</xs:simpleType>  
</xs:attribute>
```

 Questions or comments?  [Contact us.](#)

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Specsheet

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Product Specs as of v1.4.0

Component	Details
RDBMS	<ul style="list-style-type: none">Object persistence is provided by Hibernate v3.2.6, which in turn uses JDBC to connect with a back-end RDBMS.DDL is generated by Hibernate. Grouper is currently tested against Oracle 9i, MySQL ?? and PostgreSQL 8 but should work with most databases supported by Hibernate.
Java Servlet Container	<ul style="list-style-type: none">Servlet API Version 2.3.Known to run properly in Apache Tomcat 5.5.Have not tested/verified other servlet containers.
Authentication	Through servlet container via REMOTE_USER, or via user-installable filter. <i>Contributions:</i> Yale CAS authentication filter.
Java	JDK v5.0.6 or later.
Compiler	Ant v1.7.0 or later. Ensure that Ant can process Tomcat tasks by verifying that tools.jar (found in \$JAVA_HOME/lib) and catalina-ant.jar (in \$TOMCAT_HOME/server/lib) are in the classpath.
Browser Requirements	<ul style="list-style-type: none">XHTML 1.0CSS 2.1cookies must be enabledjavascript must be enabled

Grouper Product Specs as of v1.3.0

Component	Details
RDBMS	<ul style="list-style-type: none">Object persistence is provided by Hibernate v3.2.6, which in turn uses JDBC to connect with a back-end RDBMS.DDL is generated by Hibernate. Grouper is currently tested against HSQLDB 1.7.2, Oracle 9i and PostgreSQL 8 but should work with most databases supported by Hibernate.
Java Servlet Container	<ul style="list-style-type: none">Servlet API Version 2.3.Known to run properly in Apache Tomcat 5.5.Have not tested/verified other servlet containers.

Authentication	Through servlet container via REMOTE_USER, or via user-installable filter. <i>Contributions:</i> Yale CAS authentication filter.
Java	JDK v5.0.6 or later.
Compiler	Ant v1.6.3 or later.
Browser Requirements	<ul style="list-style-type: none"> • XHTML 1.0 • CSS 2.1 • cookies must be enabled • javascript must be enabled

Grouper Product Spec Sheet through v1.2.1

Component	Details
RDBMS	<ul style="list-style-type: none"> • Object persistence is provided by Hibernate v2.1.8, which in turn uses JDBC to connect with a back-end RDBMS. • DDL is generated by Hibernate. Grouper is currently tested against HSQLDB 1.7.2, Oracle 9i and PostgreSQL 8 but should work with most databases supported by Hibernate.
Java Servlet Container	<ul style="list-style-type: none"> • Servlet API Version 2.3. • Known to run properly in Apache Tomcat 4.1 and 5.5. • Have not tested/verified other servlet containers.
Authentication	Through servlet container via REMOTE_USER, or via user-installable filter. <i>Contributions:</i> Yale CAS authentication filter.
Java	JDK v5.0.6 or later.
Compiler	Ant v1.6 or later.
Browser Requirements	<ul style="list-style-type: none"> • XHTML 1.0 • CSS 2.1 • cookies must be enabled • no javascript needed, except for debug mode used only by UI developers

Reference

Java: If needed, download and install the latest version of JDK 1.5.0+ (5.0) from <http://java.sun.com/j2se/1.5.0>.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Technical FAQ

This page last changed on Feb 02, 2009 by tbarton@uchicago.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

Technical FAQ about Grouper

Grouper is [Licensed](#) under the Apache 2.0 license.

1. [How do I get group information out of Grouper and into my operational systems?](#)
2. ["ant schemaexport" creates 14 tables, 2 of which are "subject" and "subjectattribute". Do I need these?](#)
3. [How do I bootstrap membership in the wheel group?](#)
4. [Can I add custom attributes to Grouper groups for my custom purposes?](#)
5. [How do I shibbolize Grouper?](#)
6. [I am using Oracle for my Grouper database, and when I try to add more groups or members, I am getting this error: "hibernate commit error: Could not execute JDBC batch update." What causes that?](#)
7. [Grouper is failing to query my LDAP server over a SSL connection because it cannot find the certificate for the CA that signed the cert the LDAP server presents. How can I help Grouper find the CA cert?](#)
8. [Are there examples for how to create a Grouper WS client using PHP?](#)

1. How do I get group information out of Grouper and into my operational systems?

With the 1.0 release, Grouper includes an [XML import and export tool](#) that can be used for episodic or periodic provisioning of group info to other contexts. The [GrouperShell](#) can likewise be used to load and retrieve group information.

With the release of Ldapppc 1.0 (the LDAP Provisioning Connector) we now have a near-real-time "provisioning connector" that can update LDAP directories or other run-time security infrastructure services. See [\[LDAP Provisioning Connector\]](#) for more information.

With the release of Grouper 1.2.0 there is also a Web Services interface to Grouper. See [\[https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Product\]](https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Product) for more information.

2. "ant schemaexport" creates 14 tables, 2 of which are "subject" and "subjectattribute". Do I need these?

No. They are there only to support the [quickstart demo](#) and testing the API. They can safely be removed or ignored if you are using an outside subject source such as an LDAP directory.

3. How do I bootstrap membership in the wheel group?

The [GrouperShell](#) can be used for this purpose. See [Initializing Administration of Privileges](#) for the details.

4. Can I add custom attributes to Grouper groups for my custom purposes?

Yes. Custom single-valued string attributes and lists of subjects can be added to Grouper groups and subsequently managed by the API and the UI. See [Custom Group Types](#) for all of the details.

5. How do I shibbolize Grouper?

By default, Grouper relies on an external authentication service to identify authenticated principals to it through the servlet container's REMOTE_USER, so configure your shibboleth AAP to provide a suitable

identifier to Grouper as REMOTE_USER. In addition, you'll need to arrange that the same identifiers are provided to Grouper through a [source adapter](#) so that shibboleth-authenticated principals can have a security context created for them.

6. I am using Oracle for my Grouper database, and when I try to add more groups or members, I am getting this error: "hibernate commit error: Could not execute JDBC batch update." What causes that?

One cause may be that you have run out of tablespace - try extending your tablespace for the Grouper database.

7. Grouper is failing to query my LDAP server over a SSL connection because it cannot find the certificate for the CA that signed the cert the LDAP server presents. How can I help Grouper find the CA cert?

One way is to add the CA cert to the list of trusted CAs that the Java JRE keeps. The JRE provides the keytool executable to help you manage the list. With JAVA_HOME set appropriately and JAVA_HOME/bin in your path you should be able to run

```
keytool -import -file /path/to/cacert/file.pem -keystore $JAVA_HOME/jre/lib/security/cacerts
```

Note that the default password is 'changeit'. You should change it! See <http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html> for details on the keytool and how to change the password for the trusted CA keystore.

8. Are there examples for how to create a Grouper WS client using PHP?

[Click here for some simple client examples that use the PHP SOAP extension](#)

 Questions or comments?  [Contact us](#).

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

This page last changed on Jan 09, 2009 by skoranda@gravity.phys.uwm.edu.

Examples of Grouper WS clients in PHP

- [creating a new stem](#)
- [creating a new group](#)
- [adding a member to a group](#)
- [querying for the members of a group](#)
- [deleting a member from a group](#)

PHPClientAddMember

This page last changed on Jan 09, 2009 by skoranda@gravity.phys.uwm.edu.

PHP example for adding a member to a group with Grouper WS

Requires the PHP SOAP extension

```
<?php

$client = new SoapClient("https://some.server.edu/grouper-ws/services/GrouperService?wsdl",
    array(
        'login' => 'GrouperSystem',
        'password' => 'XXXXXXXXX',
        'trace' => 1)
);

$group = 'etc:simpleteststem:simplegroup';

// should be an id known to the Grouper API
$name = 'jane.user';

$params = array(
    'clientVersion' => 'v1_4_000',
    'wsGroupLookup' => array(
        'groupName' => $group
    ),
    'subjectLookups' => array(
        'subjectId' => $name
    )
);

$return = $client->addMember($params);

$success = $return->return->resultMetadata->success;

if ($success == 'T'){
    echo "Added $name to group $group\n";
} else {
    echo "Failed to add $name to group $group\n";
}

// uncomment to see the SOAP payload sent to the server
// echo "REQUEST:\n" . $client->__getLastRequest() . "\n";

// uncomment to see the SOAP payload returned by the server
// echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";

?>
```

PHPClientDeleteMembers

This page last changed on Jan 09, 2009 by skoranda@gravity.phys.uwm.edu.

PHP example for deleting a member from a group with Grouper WS

Requires the PHP SOAP extension

```
<?php

$client = new SoapClient("https://some.server.edu/grouper-ws/services/GrouperService?wsdl",
    array(
        'login' => 'GrouperSystem',
        'password' => 'XXXXXXXXX',
        'trace' => 1)
);

$group = 'etc:simpleteststem:simplegroup';

// should be Id recognized by Grouper API
$name = 'jane.user';

$params = array(
    'clientVersion' => 'v1_4_000',
    'wsGroupLookup' => array(
        'groupName' => $group
    ),
    'subjectLookups' => array(
        'subjectId' => $name
    )
);

$return = $client->deleteMember($params);

$success = $return->return->resultMetadata->success;

if ($success == 'T'){
    echo "Deleted $name from group $group\n";
} else {
    echo "Failed to delete $name from group $group\n";
}

// uncomment to see the SOAP payload sent to the server
// echo "REQUEST:\n" . $client->__getLastRequest() . "\n";

// uncomment to see the SOAP payload returned by the server
// echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";

?>
```

PHPClientGetMembers

This page last changed on Jan 09, 2009 by skoranda@gravity.phys.uwm.edu.

PHP example for querying the members of a group with Grouper WS

Requires the PHP SOAP extension

```
<?php

$client = new SoapClient("https://some.server.edu/grouper-ws/services/GrouperService?wsdl",
    array(
        'login' => 'GrouperSystem',
        'password' => 'XXXXXXXXX',
        'trace' => 1
    )
);

$groupName = 'etc:sysadmingroup';

$params = array(
    'clientVersion' => 'v1_4_000',
    'wsGroupLookups' => array(
        'groupName' => $groupName,
        'uuid' => ""
    ),
    'memberFilter' => 'All'
);

$return = $client->getMembers($params);

echo "Members of group " . $groupName . "\n";

foreach ($return->return->results->wsSubjects as $subject) {
    echo "Member ID: " . $subject->id . "\n";
}

// uncomment to see the SOAP payload sent to the server
// echo "REQUEST:\n" . $client->__getLastRequest() . "\n";

// uncomment to see the SOAP payload returned by the server
// echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";

?>
```

PHPClientGroupSave

This page last changed on Jan 09, 2009 by skoranda@gravity.phys.uwm.edu.

PHP example for creating a new group with Grouper WS

Requires the PHP SOAP extension

```
<?php

$client = new SoapClient("https://some.server.edu/grouper-ws/services/GrouperService?wsdl",
    array(
        'login' => 'GrouperSystem',
        'password' => 'XXXXXXXX',
        'trace' => 1)
);

$name = 'etc:simpleteststem:simplegroup';

$params = array(
    'clientVersion' => 'v1_4_000',
    'wsGroupToSaves' => array(
        'saveMode' => 'Insert',
        'wsGroup' => array(
            'description' => 'a simple test for group creation',
            'displayExtension' => 'Simple Test Group',
            'displayName' => "",
            'extension' => 'simplegroup',
            'name' => $name,
        ),
        'wsGroupLookup' => array(
            'groupName' => "",
            'uuid' => ""
        )
    ),
    'actAsSubjectLookup' => array(
        'subjectId' => "",
        'subjectIdentifier' => "",
        'subjectSourceId' => ""
    ),
    'txType' => 'READ_WRITE_NEW'
);

$return = $client->groupSave($params);

$success = $return->return->resultMetadata->success;

if ($success == 'T'){
    echo "Created group $name\n";
} else {
    echo "Failed to create group $name\n";
}

// uncomment to see the SOAP payload sent to the server
// echo "REQUEST:\n" . $client->__getLastRequest() . "\n";

// uncomment to see the SOAP payload returned by the server
// echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";

?>
```

PHP example for creating a new stem with Grouper WS

Requires the PHP SOAP extension

```
<?php

$client = new SoapClient("https://some.server.edu/grouper-ws/services/GrouperService?wsdl",
    array(
        'login' => 'GrouperSystem',
        'password' => 'XXXXXXXXX',
        'trace' => 1
    )
);

$name = 'etc:simpleteststem';

$params = array(
    'clientVersion' => 'v1_4_000',
    'wsStemToSaves' => array(
        'saveMode' => 'Insert',
        'wsStem' => array(
            'description' => 'a simple test for stem creation',
            'displayExtension' => 'Simple Test',
            'displayName' => '',
            'extension' => 'simplestem',
            'name' => $name,
        ),
        'wsStemLookup' => array(
            'stemName' => '',
            'uuid' => ''
        )
    ),
    'actAsSubjectLookup' => array(
        'subjectId' => '',
        'subjectIdentifier' => '',
        'subjectSourceId' => ''
    ),
    'txType' => 'READ_WRITE_NEW'
);

$return = $client->stemSave($params);

$success = $return->return->resultMetadata->success;

if ($success == 'T'){
    echo "Created stem $name\n";
} else {
    echo "Failed to create stem $name\n";
}

// uncomment to see the SOAP payload sent to the server
// echo "REQUEST:\n" . $client->__getLastRequest() . "\n";

// uncomment to see the SOAP payload returned by the server
// echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";

?>
```

UI Building and Configuration

This page last changed on Dec 05, 2007 by jbibbee@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Configuring and Deploying the Grouper UI v1.2.1

In this section we describe how to configure, build, and deploy the Grouper UI.

UI Configuration

For many purposes, UI customization needs can be met by altering declarations in the grouper-ui/resources/grouper/media.properties file. Logos, use of subject attributes in various search and display contexts, sorting behavior, and much more is specified in this file. See [Media Properties](#) for the details.

The UI is designed to be deeply customizable while remaining "upgrade proof". Readers needing all of the gory details should consult [Customising the Grouper UI](#).

Building & Deploying

1. **Copy grouper-ui/build.properties.template to grouper-ui/build.properties.**
2. **Review grouper-ui/build.properties.**
 - If you want the build script to automatically install the UI in your Tomcat instance, uncomment and set the appropriate value for deploy.home. If you do not set this you will need to copy the UI to your Tomcat installation's webapps directory. You will probably want to define the default.webapp.folder to suit how you intend to develop or customise the UI. See the [Grouper UI Development Environment](#) for options.
 - Make sure you set the grouper.folder property to the location of your Grouper installation.
3. **Copy grouper-ui/template-tomcat-context.xml to grouper-ui/tomcat-context.xml** (or the value of the property deploy.context.xml if you have changed this).
 - Tomcat specific configuration can be added in this file e.g., container managed data sources.
4. **Change directory to grouper-ui and type "ant".**
 - A list of build targets is displayed. If you have set deploy.home enter "default". Otherwise type "dist" or "war". If the former copy <dist.home>/grouper to <TOMCAT_HOME>/webapps, or if the latter, copy <dist.home>/grouper.war to <TOMCAT_HOME>/webapps.
 - If you want to take advantage of the 'nice' targets you must uncomment and set appropriate values for all the deploy properties in grouper-ui/build.properties.

Note: The build process will attempt to create a directory peer to the grouper-ui directory. Hence, the directory grouper-ui/.. must be writable.

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Media Properties

This page last changed on May 21, 2008 by tbarton@uchicago.edu.

[GROUPER: FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

media.properties

This guide to media.properties was released with Grouper v1.3.0.

`grouper-ui/resources/grouper/media.properties` controls many aspects of the Grouper UI's appearance and behaviour

- [Simple Look and feel](#)
- [Menu Configuration](#)
- [Miscellaneous UI configuration](#)
- [Membership Import and export](#)
- [Displaying lists](#)
- [Searching](#)
- [Sorting](#)
- [Plugin browse / search](#)
- [Dynamic tiles](#)
- [ObjectAsMap Implementations](#)

Simple Look and feel

How to change logos and CSS

#You may specify a logo for your organisation and for Grouper. Off-the-shelf
#your organisation logo appears on the left of the header and the Grouper logo
#appears on the right. Typically you would make the logos the same height.

```
image.organisation-logo =grouper/images/organisation-logo.gif
image.grouper-logo      =grouper/images/grouper.gif
```

#A space separated list of one or more .css files which are inserted into the
#HEAD of all Grouper pages. The .css files are referenced in order and after
#any Grouper CSS files. This means that your CSS files can override any
#Grouper style definition

css.additional =

#You can omit the Grouper CSS files completely by setting grouper-css.hide=true

grouper-css.hide = false

Menu Configuration

Out-of-the-box Grouper defines a standard set of menu items. It is possible to add additional menu items and to change the order in which they appear

#space separated list of files - see default for format - which define menu items

```
menu.resource.files =resources/grouper/menu-items.xml
```

```
#space separated list of menu item names (which must exist in 'menu.resource.files')
```

menu.order = MyGroups ManageGroups CreateGroups JoinGroups
AllGroups SearchSubjects SavedGroups SavedSubjects Help

```
#space separated list of MenuFilters - in the order they are tested
```

menu.filters =edu.internet2.middleware.grouper.ui.RootMenuFilter
edu.internet2.middleware.grouper.ui.GroupMembershipMenuFilter

#Determines if the menu is processed once at the start of a user session or whether
#it is processed with each request. Use 'true' for production and 'false' if you
#are actively developing the menu and want to see changes immediately

menu.cache =true

Miscellaneous UI configuration

#Specifies whether the error page should include a ticket. The ticket appears
#in the error log and can be used to filter relevant messages

error.ticket =@error.ticket@

#Specifies whether Grouper should display a logout link. Not all authentication
#schemes allow logout, including Basic authentication.
#This value can be set in the Grouper UI build.properties file

logout.link.show =@logout.link.show@

##Set this to 'all' to remove all cookies, or set to a comma or space separated list of
##cookie names to delete. Java code will do a Cookie.getName().equals or .matches
##so valid regular expressions may be used

logout.cookies-to-delete =none

#The ROOT stem has no name. Setting default.browse.stem causes the UI to display
#the configured value

default.browse.stem =@default.browse.stem@

#If you have admin privileges this is where you go initially

admin.browse.path =/populateAllGroups.do

#When creating a group, which access privs will be granted to GrouperAll?
#groups.create.grant.all allows the UI to override the defaults in grouper.properties

#If not set, the defaults from the grouper.properties file will be used
#NB in the QuickStart, no privs are automatically assigned - grouper.properties was
#modified so that all 'groups.create.grant.all.<priv>' are false,

groups.create.grant.all =view read

#If true, on the 'Subject Search' page there will be a link to your 'Subject Summary'

allow.self-subject-summary =true

#Unless otherwise configured, the UI starts browsing at the ROOTstem. set default.browse.stem
#to start browsing from a different stem

###default.browse.stem =uob

#Grouper has no formal notion of 'personal' stems vs 'institutional' stems, however, setting
#personal.browse.stem, will trim this portio of the hierarchy when a user is browsing in 'All' mode
#TODO members of Wheel group / GrouperSystem should be able to browse regardless.

###personal.browse.stem =uob:personal

#The UI has a 'Saved Groups' feature intended to make it easier to find groups of interest
#and used when ceating comosite groups. This property, if true, causes any new or updated group
#to be automatically added to your list of saved groups

put.in.session.updated.groups =true

#Turn off the debug functionality - NOT implemented yet

#Can be set in the Grouper UI build.properties file

debug.off =@debug.off@

#The directory where preferences are saved

debug.prefs.dir =@debug.prefs.dir@

Membership Import and export

As of V1.2 the UI provides a framework for allowing users to export membership lists to flat files i.e. comma separated files, including in Excel compatible format. It is also possible to import simple delimited files.

Both import and export require configuration and appropriate configuration will vary from site to site. For this reason, the UI does not come pre-configured for import/export, however, sample files are provided (see [Enabling import-export of group memberships](#))

membership-export.config =resources/grouper/membership-export.xml

membership-import.config =resources/grouper/membership-import.xml

If the user does not select a file to import allow text to be typed / pasted into textarea
Since version 1.2.1

membership-import.allow-textarea =true

Displaying lists

#When browsing or searching the UI will present lists of various objects. The following settings
#allow sites to control default page sizes and a list of user-selectable page sizes

pager.pagesize.default =50
pager.pagesize.selection =10 25 50 100

#Typically, when browsing it is sufficient to show the extension/displayExtension for a group/stem
#as the parent stems are already shown and are common. When searching, however, this context is lost
#so sites can configure which field to display in the context of a search where results may come from
#different locations

search.group.result-field =displayName
search.stem.result-field =displayName

#By setting the 'result-field-choice' properties, sites can allow users to select which
#field to use for displaying search results

search.group.result-field-choice =displayName displayExtension name
search.stem.result-field-choice =displayName displayExtension name

#Prior to V1.2 sites could do little to control how subjects, groups or stems were displayed
#in the UI, beyond the display of stem/group search results, unless they created dynamic tiles
#It is now possible to control the display of stems, groups and subjects in different contexts
#In the case of subjects, an attribute can be configured based on the subject's SourceAdapter

#Provides backwards compatibility - it was assumed that all Subjects would have a 'description' attribute

subject.display.default =description

#Used for groups when displayed as a subject i.e. when displayed as member of another group

subject.display.g\gsa =displayExtension

#Used for internal Grouper subjects i.e. GrouperAll and GrouperSystem

subject.display.g\isa =name

#Default attribute to display for groups

group.display =displayExtension

#Attribute to use when browsing and the user has selected to hide the hierarchy -
#thus losing context

group.display.flat =displayName

#Default attribute for stems

stem.display =displayExtension

Searching

#Configuration affecting how simple default group/stem searches are carried out

#Determines if the name or extension field (or neither) are searched

search.default.search-in-name-or-extension =

#Determines if the display name or display extension (or neither) is searched

search.default.search-in-display-name-or-extension =name

#On the advanced groups search screen determines how many search fields are displayed

search.max-fields =5

#On the advanced groups search screen determines how many group type select lists are displayed

search.max-group-types =3

#On the advanced stems search screen determines how many search fields are displayed

search.stems.max-fields =4

#Control whether default search can search any attribute. Valid values=only or true or false

search.default.any =false

#Control default search option

search.default =name

#Allow filtering of membership lists by subject source

members.filter.by-source =true

members.filter.limit =500

#Displays source specific form elements using keys:

#subject.search.form-fragment.<sourceId>

subject.search.form-fragment.g\:gsa =subjectSearchGroupFragmentDef

Sorting

As of V1.2 the Grouper UI allows sorting of various lists of objects

See [Sort order of lists](#) for explanation

comparator.impl =edu.internet2.middleware.grouper.ui.DefaultComparatorImpl

comparator.helper.edu.internet2.middleware.grouper.Group

=edu.internet2.middleware.grouper.ui.GroupComparatorHelper

comparator.helper.edu.internet2.middleware.grouper.ui.util.GroupAsMap

=edu.internet2.middleware.grouper.ui.GroupComparatorHelper

comparator.helper.edu.internet2.middleware.grouper.Stem=edu.internet2.middleware.grouper.ui.StemComparatorHelp

comparator.helper.edu.internet2.middleware.grouper.ui.util.StemAsMap

=edu.internet2.middleware.grouper.ui.StemComparatorHelper

comparator.helper.edu.internet2.middleware.subject.Subject

=edu.internet2.middleware.grouper.ui.SubjectComparatorHelper

comparator.helper.edu.internet2.middleware.grouper.ui.util.SubjectAsMap

```

        =edu.internet2.middleware.grouper.ui.SubjectComparatorHelper
comparator.helper.edu.internet2.middleware.grouper.Member
        =edu.internet2.middleware.grouper.ui.SubjectComparatorHelper
comparator.helper.edu.internet2.middleware.grouper.Membership
        =edu.internet2.middleware.grouper.ui.SubjectComparatorHelper
comparator.helper.edu.internet2.middleware.grouper.ui.util.MembershipAsMap
        =edu.internet2.middleware.grouper.ui.SubjectComparatorHelper
comparator.helper.edu.internet2.middleware.grouper.ui.util.SubjectPrivilegeAsMap
        =edu.internet2.middleware.grouper.ui.GroupOrStemComparatorHelper

```

#Sorting large lists can be computationally expensive - and slow. Use this property to turn off sorting for
#large lists

```
comparator.sort.limit =200
```

To control the order in which subject attributes are listed on the Subject Summary
page: #subject.attributes.order.<SOURCE_ID>=comma separated list of case sensitive attribute names

```

subject.attributes.order.g
\ :gsa =displayExtension,displayName,name,extension,createTime,createSubjectId,createSubjectType,
subject.attributes.order.qsuob =LFNAME,LOGINID,subjectType,id

```

Plugin browse / search

The UI has a pluggable interface for browsing and searching. See [Customising Browsing and Searching](#) for explanation

```

repository.browser.my.class =edu.internet2.middleware.grouper.ui.MyMembershipsRepositoryBrowse
repository.browser.my.flat-capable =true
repository.browser.my.root-node =
repository.browser.my.hide-pre-root-node =true
repository.browser.my.flat-privs =MEMBER
repository.browser.my.flat-type =group
repository.browser.my.search =groups
repository.browser.create.class =edu.internet2.middleware.grouper.ui.CreateRepositoryBrowser
repository.browser.create.flat-capable =true
repository.browser.create.root-node =
repository.browser.create.hide-pre-root-node =true
repository.browser.create.flat-privs =CREATE STEM
repository.browser.create.flat-type =stem
repository.browser.create.search =stems
repository.browser.manage.class =edu.internet2.middleware.grouper.ui.ManageRepositoryBrowser
repository.browser.manage.flat-capable =true
repository.browser.manage.root-node =
repository.browser.manage.hide-pre-root-node =true
repository.browser.manage.flat-privs =ADMIN UPDATE CREATE STEM
repository.browser.manage.flat-type =group
repository.browser.manage.search =groups
repository.browser.join.class =edu.internet2.middleware.grouper.ui.JoinRepositoryBrowser
repository.browser.join.flat-capable =true
repository.browser.join.root-node =
repository.browser.join.hide-pre-root-node =true
repository.browser.join.flat-privs =OPTIN
repository.browser.join.flat-type =group
repository.browser.join.search =groups
repository.browser.all.class =edu.internet2.middleware.grouper.ui.AllRepositoryBrowser
repository.browser.all.flat-capable =false
repository.browser.all.root-node =
repository.browser.all.hide-pre-root-node =true
repository.browser.all.flat-privs =
repository.browser.all.search =groups
repository.browser.subjectsearch.class =edu.internet2.middleware.grouper.ui.AllRepositoryBrowser
repository.browser.subjectsearch.flat-capable =true
repository.browser.subjectsearch.root-node =
repository.browser.subjectsearch.hide-pre-root-node =true

```

```

repository.browser.subjectsearch.flat-privs =
repository.browser.subjectsearch.search =groups
repository.browser.savedgroups.class =edu.internet2.middleware.grouper.ui.AllRepositoryBrowser
repository.browser.savedgroups.flat-capable =true
repository.browser.savedgroups.root-node =
repository.browser.savedgroups.hide-pre-root-node =true
repository.browser.savedgroups.flat-privs =
repository.browser.savedgroups.search =groups
repository.browser.savedsubjects.class =edu.internet2.middleware.grouper.ui.AllRepositoryBrowser
repository.browser.savedsubjects.flat-capable =true
repository.browser.savedsubjects.root-node =
repository.browser.savedsubjects.hide-pre-root-node =true
repository.browser.savedsubjects.flat-privs =
repository.browser.savedsubjects.search =groups

```

Dynamic tiles

The UI uses dynamic tiles to determine, at run time, how to display various objects
 See [Defining Custom Dynamic Templates](#) for more details

```

composite.view.default =/WEB-INF/jsp/compositeView.jsp
composite.view.asFactor =/WEB-INF/jsp/compositeAsFactorView.jsp
composite.view.chainPath =/WEB-INF/jsp/compositeChainPathView.jsp
composite.view.chain =/WEB-INF/jsp/compositeChainView.jsp
subject.view.default =/WEB-INF/jsp/subjectView.jsp
subject.view.memberLink =/WEB-INF/jsp/memberLinkView.jsp
subject.view.subjectSearchResultLink =/WEB-INF/jsp/subjectSearchResultLinkView.jsp
=
subject.view.subjectInfo =/WEB-INF/jsp/subjectInfo.jsp
subject.view.groupSearchResultLink =/WEB-INF/jsp/groupSearchResultLinkView.jsp
subject.view.stemSearchResultLink =/WEB-INF/jsp/stemSearchResultLinkView.jsp
subject.view.assignFoundMember =/WEB-INF/jsp/assignFoundMemberView.jsp
subject.view.subjectAccessPriv =/WEB-INF/jsp/subjectAccessPrivView.jsp
subject.view.subjectNamingPriv =/WEB-INF/jsp/subjectNamingPrivView.jsp
subject.view.subjectSummaryLink =/WEB-INF/jsp/subjectSummaryLinkView.jsp
subject.view.current =/WEB-INF/jsp/currentSubjectView.jsp
subject.view.isMemberOf =/WEB-INF/jsp/subjectIsMemberOfView.jsp
subject.view.isIndirectMemberOf =/WEB-INF/jsp/subjectIsIndirectMemberOfView.jsp
subject.view.hasPrivilege =/WEB-INF/jsp/subjectHasPrivilegeView.jsp
subject.view.savedSubject =/WEB-INF/jsp/subjectSearchResultLinkView.jsp
group.view.hasPrivilege =/WEB-INF/jsp/subjectHasPrivilegeView.jsp
stem.view.browseHierarchy =/WEB-INF/jsp/browseChildStem.jsp
stem.view.assignFoundMember =/WEB-INF/jsp/browseChildStem.jsp
stem.view.stemSearchResultLink =/WEB-INF/jsp/stemSearchResultLinkView.jsp
stem.view.searchResultItem =/WEB-INF/jsp/stemSearchResultItemView.jsp
stem.view.default =/WEB-INF/jsp/stemView.jsp
subjectType.group.view.assignFoundMember =/WEB-INF/jsp/browseForFindChildGroup.jsp
subjectType.group.view.subjectSearchResult =/WEB-INF/jsp/
groupAsSubjectSearchResultView.jsp

```

##for subject searches which arent groups, this is the view (to put the subject image)

```

subject.view.subjectSearchResult =/WEB-INF/jsp/subjectSearchResultView.jsp
group.view.linkGroupMembers =/WEB-INF/jsp/groupLinkMembersView.jsp
group.view.compositeMember =/WEB-INF/jsp/groupChainPathView.jsp
group.view.compositeOwner =/WEB-INF/jsp/groupChainPathView.jsp
group.view.compositeGroupChainMember =/WEB-INF/jsp/
compositeGroupChainMemberView.jsp
group.view.isMemberOf =/WEB-INF/jsp/subjectIsMemberOfView.jsp
group.view.current =/WEB-INF/jsp/currentSubjectView.jsp
group.view.browseHierarchy =/WEB-INF/jsp/browseChildGroup.jsp
group.view.assignFoundMember =/WEB-INF/jsp/browseForFindChildGroup.jsp
group.view.groupSearchResultLink =/WEB-INF/jsp/groupSearchResultLinkView.jsp
group.view.groupSearchResultWithPrivs =/WEB-INF/jsp/groupSearchResultWithPrivsView.jsp

```


group.view.savedGroup	=/WEB-INF/jsp/groupSearchResultLinkView.jsp
group.view.groupMember	=/WEB-INF/jsp/subjectView.jsp
group.view.chainPath	=/WEB-INF/jsp/groupChainPathView.jsp
group.view.subjectSummaryGroupLink	=/WEB-INF/jsp/groupChainPathView.jsp
group.view.searchResultItem	=/WEB-INF/jsp/groupSearchResultItemView.jsp
group.view.groupChain	=/WEB-INF/jsp/groupChainView.jsp
group.view.default	=/WEB-INF/jsp/subjectView.jsp
membership.view.subjectSummaryMemberLink	=/WEB-INF/jsp/
subjectSummaryMemberLinkView.jsp	
membership.view.subjectSummary	=/WEB-INF/jsp/subjectSummaryMembershipView.jsp
membership.view.memberLink	=/WEB-INF/jsp/memberLinkView.jsp
membership.view.memberWithoutLink	=/WEB-INF/jsp/memberWithoutLinkView.jsp
membership.view.default	=/WEB-INF/jsp/defaultMembershipView.jsp
membership.view.removableMembershipInfo	=/WEB-INF/jsp/removableMembershipView.jsp
membership.view.compositeMember	=/WEB-INF/jsp/compositeMembershipView.jsp
subjectprivilege.view.subjectSummaryPrivilege	=/WEB-INF/jsp/subjectSummaryPrivilegeView.jsp
subjectprivilege.view.default	=/WEB-INF/jsp/defaultSubjectPrivilegeView.jsp
subjectprivilege.access.view.privilegesLink	=/WEB-INF/jsp/accessPrivilegesLinkView.jsp
subjectprivilege.naming.view.privilegesLink	=/WEB-INF/jsp/namingPrivilegesLinkView.jsp
list.view.default	=/WEB-INF/jsp/genericListView.jsp
list.view.groupSummaryGroupTypes	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.groupSummaryFields	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.editGroupAttributes	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.editAttributesFields	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.compositesAsFactor	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.searchAttributesFields	=/WEB-INF/jsp/genericItemsOnlyListView.jsp
list.view.searchForPrivAssignHeader	=/WEB-INF/jsp/
searchForPrivAssignmentListHeaderView.jsp	
list.view.searchForPrivAssignFooter	=/WEB-INF/jsp/
searchForPrivAssignmentListFooterView.jsp	
list.view.browseStemsFindHeader	=/WEB-INF/jsp/browseStemsFindListHeaderView.jsp
list.view.browseStemsFindFooter	=/WEB-INF/jsp/browseStemsFindListFooterView.jsp
list.view.removableMemberLinksHeader	=/WEB-INF/jsp/
removableMemberLinksHeaderView.jsp	
list.view.removableMemberLinksFooter	=/WEB-INF/jsp/
removableMemberLinksFooterView.jsp	
list.view.genericListHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.genericListFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.memberLinksHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.privilegeLinksHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.browseHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.findNewHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.assignHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.searchResultHeader	=/WEB-INF/jsp/genericListHeaderView.jsp
list.view.memberLinksFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.privilegeLinksFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.browseFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.findNewFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.assignFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.searchResultFooter	=/WEB-INF/jsp/genericListFooterView.jsp
list.view.chain	=/WEB-INF/jsp/chainPath.jsp
field.list.view.default	=/WEB-INF/jsp/fieldLISTView.jsp
field.list.view.withValue	=/WEB-INF/jsp/fieldLISTWithValueView.jsp
field.attribute.view.withValue	=/WEB-INF/jsp/fieldATTRIBUTEWithValueView.jsp
field.attribute.view.editValue	=/WEB-INF/jsp/fieldATTRIBUTEEditValueView.jsp
field.attribute.view.search	=/WEB-INF/jsp/fieldATTRIBUTESearchValueView.jsp
groupType.view.groupSummary	=/WEB-INF/jsp/groupTypeSummaryView.jsp
groupType.view.editGroupAttributes	=/WEB-INF/jsp/groupTypeEditAttributesView.jsp

ObjectAsMap Implementations



Allow sites to provide local implementations of Map wrappers of Grouper objects

objectasmap.StemAsMap.impl	=edu.internet2.middleware.grouper.ui.util.StemAsMap
----------------------------	---

objectasmap.GroupAsMap.impl	=edu.internet2.middleware.grouper.ui.util.GroupAsMap
objectasmap.FieldAsMap.impl	=edu.internet2.middleware.grouper.ui.util.FieldAsMap
objectasmap.MembershipAsMap.impl	=edu.internet2.middleware.grouper.ui.util.MembershipAsMap
objectasmap.SubjectAsMap.impl	=edu.internet2.middleware.grouper.ui.util.SubjectAsMap
objectasmap.SubjectPrivilegeAsMap.impl	=edu.internet2.middleware.grouper.ui.util.SubjectPrivilegeAsMap

##if the little yellow "i" images that show help should be enabled

infodot.enable =true


 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

UI Customization Guide

This page last changed on Dec 06, 2007 by jbibbee@internet2.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

 Questions or comments?  [Contact us](#).

Document	Description
Grouper UI Components	An overview of Grouper UI components.
Architecture	An explanation of the technologies used to create the Grouper UI and the specific approaches used.
Struts actions and tiles	An overview of the Struts actions and tiles defined by the Grouper UI.
Customising the Grouper UI	The QuickStart distribution contains UI customisations. This document explains those customisations - which can be used as the basis for your own site-specific customisations. Customisations may be limited to branding, page layout and text display, but can also include integrating authentication schemes and adding completely new functionality to integrate Grouper with other systems.
Grouper UI Development Environment	A description of the actual development environment used to develop the Grouper UI.
Grouper UI Contributed Code	A list of contributed code.

Unresolvable Subject Deletion Utility (USDU)

This page last changed on Jan 04, 2009 by tbarton@uchicago.edu.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Unresolvable Subject Deletion Utility (USDU)

This document is released alongside Grouper v1.4.0.

The Unresolvable Subject Deletion Utility finds and optionally deletes memberships for subjects which can not be found by their source.

An unresolvable subject is a subject that can not be found by its source. A subject may be unresolvable because of a temporary or permanent source failure, or because it was removed from its source before memberships or privileges were deleted or revoked.

This utility attempts to lookup every member's subject. If a subject can not be found, it's immediate memberships are printed and optionally deleted.

A future version may extend the Source class to provide more efficient lookups of subjects.

Usage

```
$GROUPER_HOME/bin/gsh.sh -usdu <command line arguments>
```

Without any arguments, usdu prints its usage

```
usage: USDU -all | -source <arg> | -uuid <arg> [-delete] [-start <arg>]
-all          find unresolvable subjects from all sources
-delete       delete memberships and privileges
-source <arg>  find unresolvable subjects from source
-start <arg>   start session as this subject, default GrouperSystem
-uuid <arg>    find unresolvable subject with member uuid
```

Unresolvable subjects are printed to stdout.

If an unresolvable subject is not a member of any groups:
member_uuid='<uuid>' subject='<id>' no_memberships

For every group or stem and list that an unresolvable subject is a member of:
member_uuid='<uuid>' subject='<id>' group|stem='<name>' list='<name>' [delete]

For every unresolvable subject, usdu prints one line for every immediate membership. If an unresolvable subject is not a member of any groups and has no privileges, usdu prints no_memberships.

Find unresolvable subjects from all sources

```
$GROUPER_HOME/bin/gsh.sh -usdu -all
member_uuid='e58697e4-...' subject='id1'/'source'/'person' group='stem:group1' list='members'
member_uuid='e58697e4-...' subject='id2'/'source'/'person' group='stem:group2' list='members'
...
```

Find unresolvable subjects from a specified source

```
$GROUPER_HOME/bin/gsh.sh -usdu -source CustomSource
```

Find unresolvable subject via member uuid

```
$GROUPER_HOME/bin/gsh.sh -usdu -uuid e58697e4-11a5-4082-b318-cb1e79191923
```



Delete unresolvable subject from all groups

```
$GROUPER_HOME/bin/gsh.sh -usdu -uuid e58697e4-... -delete  
member_uuid='e58697e4-...' subject='id1'/'source'/'person' group='stem:group1' list='members' delete
```

Details

This utility finds and deletes memberships and privileges. It is possible for an unresolvable subject to be a creator or modifier of a group, in that case, calling `Group.getCreateSubject()` or `Group.getModifySubject()` will result in a `SubjectNotFoundException`.


Unresolvable subjects are not deleted from the `grouper_members` table. If an unresolvable subject becomes resolvable again, it will retain its member uuid.

 Questions or comments?  Contact us.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Hooks

This page last changed on Oct 13, 2009 by mchalyzer@idp.protectnetwork.org.

 [Contact us](#) if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

3rd Party Hooks as of v1.4.0

Hooks: points in the grouper API that will callback custom code if configured to do so. The custom code has the opportunity to be notified that something happened, change data as it is being operated on, veto operations, kickoff other logic (log something, call another grouper API method, etc).

Hooks are available for the following database persistable objects: Group, Stem, Member, Membership, Composite, Field, GrouperSession, GroupType, GroupTypeTuple

[Getting started with hooksProof of concept](#)

[Getting started with hooks](#)

Each of these objects has low level hooks associated with it: preInsert, postInsert, postCommitInsert, preUpdate, postUpdate, postCommitUpdate, preDelete, postDelete, postCommitDelete. Note, some of these operations dont really make sense but are there anyway (e.g. there is no way to update a GroupType, and no way to delete Member). The low level hooks are called with each hibernate call for an object which generally corresponds to a row in a DB table (not for groups/attributes which have one hook but which are a one-to-many). The pre is before the sql statement is sent to the DB, and the post is after, though both are before the SQL commit. The postCommit kicks off right after the commit is called in the database transaction (note: since we have long running transactions this could be after a long workflow). You would use the pre to edit the object or to insert SQL before the sql. The post is used to get the hibernate id of an insert (e.g. to add rows with foreign keys). Both can be used to veto a call if synchronous [default]. Asynchronous hooks, and postCommit hooks cannot veto. Post commit hooks do not participate in the same database transaction as the business logic. Post commit hooks can be used for economical notifications (note, there are more robust strategies also). Post commit and asynchronous hooks receive copies (clones) of the business objects so the data is consistent and safe.

There are high level hooks which encompass multiple low level operations, e.g. addMember, removeMember (only 2 so far)

Finally there are hooks on general lifecycle activities in Grouper, called LifecycleHooks. Examples are: when grouper starts up, when hooks start (to register suites of hooks), and when hibernate is being configured (to register another hibernate mapping into the grouper hibernate configuration session factory).

Configure

To configure a hook, subclass one of the [hooks base classes](#), e.g. GroupHooks, MembershipHooks, etc. Then either register this in the grouper.properties (see [grouper.example.properties](#) for documentation).

Note the class must be threadsafe, as an instance is cached and called repeatedly (i.e. dont store instance vars etc)

```
#implement a group hook by extending edu.internet2.middleware.grouper.hooks.GroupHooks
hooks.group.class=edu.yourSchool.it.YourSchoolGroupHooks,edu.yourSchool.it.YourSchoolGroupHooks2
```

Note you can register multiple hooks by comma separating them. Or you can register runtime with:

```
GrouperHookType.addHookManual("hooks.group.class", YourSchoolGroupHooks2.class);
```

If you want to register a suite of hooks, just register a lifecycle hook:

```
hooks.lifecycle.class=edu.internet2.middleware.grouper.hooks.LifecycleHooksImpl
```

Then in the hook init method, register some hooks for the suite:

```
package edu.internet2.middleware.grouper.hooks;
```

```

import edu.internet2.middleware.grouper.hooks.beans.HooksContext;
import edu.internet2.middleware.grouper.hooks.beans.HooksLifecycleHooksInitBean;
import edu.internet2.middleware.grouper.hooks.logic.GrouperHooksUtils;

/**
 *
 */
public class LifecycleHooksImpl extends LifecycleHooks {

    /**
     * @see
     edu.internet2.middleware.grouper.hooks.LifecycleHooks#hooksInit(edu.internet2.middleware.grouper.hooks.beans.HooksContext,
     *      edu.internet2.middleware.grouper.hooks.beans.HooksLifecycleHooksInitBean)
     */
    @Override
    public void hooksInit(HooksContext hooksContext,
        HooksLifecycleHooksInitBean hooLifecycleHooksInitBean) {
        GrouperHooksUtils.addHookManual("hooks.group.class", MyGroupHooks.class);
        GrouperHooksUtils.addHookManual("hooks.stem.class", MyStemHooks.class);
        GrouperHooksUtils.addHookManual("hooks.member.class", MyMemberHooks.class);
    }
}

```

Finally, I picture built-in grouper suites to just insert the registration based on grouper config property in GrouperHooksUtils in the initHooks block. To enable the suite the grouper properties file would just have a boolean switch. When someone wants to get this working, let me know the grouper properties config property and the LifecycleHooks subclass and I can do an example.

API

Each hook method gets two arguments as callbacks. The hooks context, and the bean that holds the relevant data for the hook.

The hooks context (still needs to be completed), holds information such as the current user, the current environment (UI vs WS etc), threadlocal data (e.g. the current http request), etc. The hooks bean holds information such as the Group which is being deleted. If any new arguments need to be added to a hook method, they will be added to one of these beans, so the signature of the method won't change, so people don't have to recompile on upgrade.

To find out how the data has changed (in an update or delete), you can use the [dbVersion API](#).

To veto a hook, throw a HookVeto which is a runtime exception. The specific hooks which is vetoing will be assigned to the exception, and will be known wherever it is caught. This HookVeto takes a system name and a friendly description of a reason why it is being vetoed. The system name can be used to look up a localized error message. The friendly version can be used for logging or if a localized message doesn't exist. You can only veto pre and post synchronous hooks. You cannot veto asynchronous or postCommit hooks. You also cannot veto lifecycle hooks (e.g. hibernate init).

Hooks rely on all DB code going through the Grouper hibernate API (see the class HibernateSession). If any DB code uses the Session object (from hibernate) directly, then hooks will not fire. You can use the Grouper Hibernate API and circumvent hooks like this (e.g. in a reset() method):

```
hibernateSession.byObject.setIgnoreHooks(true).delete(_type.getFields());
```

If you want to make a hook non-reentrant (while in hook, do not let that particular hook fire), you can easily setup a threadlocal to accommodate. e.g.

```

private static ThreadLocal<Boolean> inOnPreUpdate = new ThreadLocal<Boolean>();

/**
 * @see edu.internet2.middleware.grouper.hooks.StemHooks#stemPostUpdate()
 */
@Override

```

```

public void stemPreUpdate(HooksContext hooksContext, HooksStemBean postUpdateBean) {

    Boolean inOnPreUpdateBoolean = inOnPreUpdate.get();
    try {

        if (inOnPreUpdateBoolean == null || !inOnPreUpdateBoolean) {
            inOnPreUpdate.set(true);      ... do logic ...
        }
    } finally {
        //if we changed it
        if (inOnPreUpdateBoolean == null || !inOnPreUpdateBoolean) {
            //change it back
            inOnPreUpdate.remove();
        }
    }
}
}

```

The HooksContext contains a map of attributes. These attributes can be set with static methods in HooksContext, and each attribute is designated as allowed to be copied to another thread or not. e.g. the HttpServletRequest is only valid during a request, so it shouldn't be available in another thread which might last longer than the request. There are some constants in HooksContext for common keys. Some of the built-in values for the attributes are: HttpServletRequest, HttpServletResponse, HttpSession (Http classes for UI and WS only). The hooks context also holds the current context name. This is retrieved with hooksContext.getGrouperContextType(). This can be assigned (with static method in GrouperContextTypeBuiltIn) in the current thread or global default. This is available everywhere in grouper, not just hooks. If it is not set, it is UNKNOWN. Also the current user is available from the context. There is the logged in user, the actAs user (perhaps WS only), and the GrouperSession user. You can make decisions based on which user is using the app. There are also convenience methods (e.g. boolean hooksContext.isSubjectFromGrouperSessionInGroup(groupName))

Asynchronous hooks

Hooks by default are synchronous: they run in the same thread as the thread which is doing the work that is hooked.

If you want the login of a hook to be asynchronous (e.g. to not slow down the current thread), then it will not be in the same transaction or have the same data available (e.g. not the HttpServletRequest in a web related hook). You cannot veto an asynchronous hook, or participate in the same database transaction as the business logic. Asynchronous hooks retrieve a different HooksContext (threadsafe), and a copy of the hooks bean (so no toes are stepped on, and the data is a snapshot). To do asynchronous hooks, there are two ways:

1. Make the hook implementation class implement the marker interface HookAsynchronousMarker (note all hook methods in this hook implementation will be asynchronous), e.g.

```

public class MembershipHooksImplAsync2 extends MembershipHooks implements HookAsynchronousMarker {

```

- or- 2. Call the asynchronous callback, anything in the callback is in a different thread [asynchronous]

```

public class MembershipHooksImplAsync extends MembershipHooks {

    /**
     *
     */
    @Override
    public void membershipPreAddMember(HooksContext hooksContext,
        HooksMembershipChangeBean preAddMemberBean) {
        //do logic in same thread/transaction
        //also you can get data from the beans above, set final, and use below (if it wont be available otherwise, this
        might be rare)
        HookAsynchronous.callbackAsynchronous(hooksContext, preAddMemberBean, new
        HookAsynchronousHandler() {

            public void callback(HooksContext hooksContext, HooksBean hooksBean) {

```

```

HooksMembershipChangeBean preAddMemberBeanThread = (HooksMembershipChangeBean)hooksBean;
    //do logic in a different thread/transaction

}

});

}

}

```

Logging

There is logging about which hooks execute when, for how long, and how they end (normal, veto, exception). To enable logging, add something like this to the log4j.properties:

```

# see hook debug info
log4j.logger.edu.internet2.middleware.grouper.hooks = DEBUG, grouper_debug

```

Then in the debug log, you will see info about all hooks (note, the ID's are for the purpose of matching the log statements start and end, also can be accessed from the HooksContext:

```

2008/07/09 02:18:05.482 GrouperHooksUtils.executeHook(174) - START: Hook
GroupTypeTupleHooksImpl.groupTypeTuplePreInsert id: PSPTRJ8J
2008/07/09 02:18:05.497 GrouperHooksUtils.executeHook(181) - END (normal): Hook
GroupTypeTupleHooksImpl.groupTypeTuplePreInsert id: PSPTRJ8J (15ms)
2008/07/09 02:18:05.497 GrouperHooksUtils.executeHook(174) - START: Hook
GroupTypeTupleHooksImpl.groupTypeTuplePostInsert id: PSPTRJ8K
2008/07/09 02:18:05.497 GrouperHooksUtils.executeHook(186) - END (veto): Hook
GroupTypeTupleHooksImpl.groupTypeTuplePostInsert id: PSPTRJ8K (0ms),
veto key: hook.veto.groupTypeTuple.insert.name.not.test4, veto message: name cannot be test4

```

Use cases

- [Validate that a group extension conforms to standards, veto if not](#)
- [Only grouperLoader itself \(or someone in the wheel group\), can add a member to a group with type "grouperLoader"](#)
- [A new or updated group emailAddress attribute should not be in use by another group or subject](#)

To do

- Add more high level hooks
- Collect and code use cases

Hooks code

Here is an [email to signet](#) describing how grouper hooks work and where the code is

 [Contact us](#) if you have additional comments or suggestions.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Include exclude and require groups

This page last changed on Nov 05, 2008 by [mchyzer](#).

Grouper has optional support for types, attributes, and hooks which facilitate easily setting up include and exclude membership list, and being able to require memberships in other groups to enable a membership in a group.

Using

To enable this feature, set the two settings to true in the grouper.properties config

```
#if the addIncludeExclude and requireInGroups should be enabled, and if the type(s) should be
#auto-created, and used to auto create groups to facilitate include and exclude lists, and require lists
grouperIncludeExclude.use = false
grouperIncludeExclude.requireGroups.use = false
```

Decide which groups should be built-in types or attributes for "require groups". If it is a type, it should probably be global so that most users can benefit from it (e.g. activeEmployee). If it is an attribute, it is a little more hidden. If not neither, users can still enter in group names in the requireGroups freeform attribute:

```
#for requireGroups (groups that the members must be to be in the overall group). name is the name of the
attribute or type
#attributeOrType is either attribute for an attribute underneath the requireInGroups type, or type to be a top
level type
#group is the group to be added in. note attributes are a global namespace, so you might want to use a
naming convention,
#e.g. prefix with "require". description is the tooltip. add as many as you like.
#grouperIncludeExclude.requireGroup.name.0 = requireActiveEmployee
#grouperIncludeExclude.requireGroup.attributeOrType.0 = type
#grouperIncludeExclude.requireGroup.group.0 = school:community:activeEmployee
#grouperIncludeExclude.requireGroup.description.0 = If value is true, members of the overall group must be
an active employee (in the school:community:activeEmployee group). Otherwise, leave this value not filled in.

#grouperIncludeExclude.requireGroup.name.1 = requireActiveStudent
#grouperIncludeExclude.requireGroup.attributeOrType.1 = attribute
#grouperIncludeExclude.requireGroup.group.1 = school:community:activeStudent
#grouperIncludeExclude.requireGroup.description.1 = If value is true, members of the overall group must be
an active student (in the school:community:activeStudent group). Otherwise leave this value not filled in.
```

Screenshots

Here is a stem:

Explore






You can look for groups throughout the hierarchy.
(You might not be able to see some groups if you lack appropriate privileges.)

Browse or list groups

Current location is:

 Root:  aStem

Showing 1-5 of 5 items


-  activeEmployee
-  activeStudent
-  aGroup
-  anotherGroup
-  yetAnotherGroup

Search groups

[Advanced groups search](#)

Search groups

Search from

Root 

er is sponsored by

Include and exclude tooltip

EXPLORE

Edit group ⓘ

Current location is:

📁 Root: 📁 aStem: 👤 aGroup

Name	<input type="text" value="aGroup"/>
ID	<input type="text" value="aGroup"/>
Description	<input type="text" value="Group containing list of aGroup after adding the include"/>
Assign privileges to everyone	<input type="checkbox"/> admin <div>Select this type to auto-create other groups which facilitate include and exclude list</div>
Select group types	<input type="checkbox"/> addIncludeExclude <input type="checkbox"/> requireActiveStudent <input type="checkbox"/> requireInGroups

Save

[Back to group summary](#)

Custom require group type tooltip (configured in grouper.properties)

EXPLORE

Edit group ⓘ

Current location is:

📁 Root: 📁 aStem: 👤 aGroup

Name	<input type="text" value="aGroup"/>
ID	<input type="text" value="aGroup"/>
Description	<input type="text" value="Group containing list of aGroup after adding the include"/>
Assign privileges to everyone	<input type="checkbox"/> admin <input type="checkbox"/> update <input checked="" type="checkbox"/> <div>If value is true, members of the overall group are active students (in the school:community:activeStudents) leave this value not filled in.</div>
Select group types	<input type="checkbox"/> addIncludeExclude <input type="checkbox"/> requireActiveStudent <input type="checkbox"/> requireInGroups

Save

[Back to group summary](#)

Require in groups tooltip (this group type holds custom attributes for require groups, also a freeform attribute):

EXPLORE

Edit group ⓘ

Current location is:
Root: aStem: aGroup

Name	aGroup
ID	aGroup
Description	Group containing list of aG
Assign privileges to everyone	<input type="checkbox"/> admin <input type="checkbox"/> update <input checked="" type="checkbox"/>
Select group types	<input type="checkbox"/> addIncludeExclude <input type="checkbox"/> requireActiveStudent <input type="checkbox"/> requireInGrou

Save

[Back to group summary](#)

Select this type to auto-create other so that other groups can be required activeEmployee)

d by

Groups created when include exclude is selected

privileges.)

Browse or list groups ⓘ

Current location is:
📁 Root: 📁 aStem

Showing 1-9 of 9 items

- 👤👤 activeEmployee
- 👤👤 activeStudent
- 👤👤 aGroup
- 👤👤 aGroup excludes
- 👤👤 aGroup includes
- 👤👤 aGroup system of record
- 👤👤 aGroup system of record and includes
- 👤👤 anotherGroup
- 👤👤 yetAnotherGroup

[Search groups](#) [Advanced groups search](#)

Selected a require group would look like this:

(You might not be able to see some groups if you lack appropriate privileges.)

Browse or list groups ⓘ

Current location is:
📁 Root: 📁 aStem

Showing 1-6 of 6 items

- 👤👤 activeEmployee
- 👤👤 activeStudent
- 👤👤 aGroup
- 👤👤 aGroup system of record
- 👤👤 anotherGroup
- 👤👤 yetAnotherGroup

This is the composite membership of the overall group when a require group is selected

Current location is:

 Root:  aStem:  aGroup



Membership list

- ☒ Show DIRECT members of this group
- ☐ Show INDIRECT members of this group
- ☐ Show ALL members of this group (direct and indirect)

Change display

Showing 1-1 of 1 items

Click an entity name to view entity details, or click a member

- This is a composite group
 -  **aStem:aGroup system of record**
intersection
 -  **aStem:activeStudent**

This is how the require in groups attributes appear:


Group summary

Note: The group [aGroup] was successfully saved

Current location is:

 Root:  aStem:  aGroup

<u>Name</u>	aGroup		
<u>Path</u>	aStem:aGroup		
<u>Description</u>	Group containing list of aGroup after adding the includes and subtracting the exclu		
<u>ID</u>	aGroup		
<u>ID Path</u>	aStem:aGroup		
<u>UUID</u>	08850138-b7cf-4d6f-b9eb-8301544fd704		
<u>Types</u>	requireActiveStudent		
	requireInGroups	requireActiveEmployee	true
		requireAlsoInGroups	aStem:anotherGroup,aStem:

Show entities with ADMIN  privilege

Here are the groups created with multiple require groups:

EXPLORE

Browse groups hierarchy ⓘ










You can look for groups throughout the hierarchy.
(You might not be able to see some groups if you lack appropriate privileges.)

Browse or list groups ⓘ

Current location is:

 Root:  aStem

Showing 1-9 of 9 items

-  activeEmployee
-  activeStudent
-  aGroup
-  aGroup requireGroups 1
-  aGroup requireGroups 2
-  aGroup requireGroups 3
-  aGroup system of record
-  anotherGroup
-  yetAnotherGroup

Here is a require group membership, based on another require group

Members ⓘ

Current location is:

 Root:  aStem:  aGroup requireGroups 2



Membership list

- ☒ Show DIRECT members of this group
- ☐ Show INDIRECT members of this group
- ☐ Show ALL members of this group (direct and indirect)

Change display

Showing 1-1 of 1 items

Click an entity name to view entity details, or click a membership description

- This is a composite group
 -  aStem:aGroup requireGroups 3
intersection
 -  aStem:activeStudent

[Remove composite group](#) [Replace composite factors](#) [Back to group](#)

d by

Here is a group with an include and exclude list, and a bunch of require groups, all chained together.

You can look for groups throughout the hierarchy.
(You might not be able to see some groups if you lack appropriate privileges.)

Browse or list groups ⓘ

Current location is:
📁 Root: 📁 aStem

Showing 1-13 of 13 items

👤

 activeEmployee

👤

 activeStudent

👤

 aGroup

👤

 aGroup excludes

👤

 aGroup includes

👤

 aGroup includes minus excludes

👤

 aGroup requireGroups 1

👤

 aGroup requireGroups 2

👤

 aGroup requireGroups 3

👤

 aGroup system of record

👤

 aGroup system of record and includes

👤

 anotherGroup

👤

 yetAnotherGroup

Configuration

```
#####
## Grouper include / exclude and requireGroups
## If enabled, will make sure the Type is installed, and when that type is
## applied to a group, it will auto-create the other groups needed to manage the include and exclude lists
## see: https://bugs.internet2.edu/jira/browse/GRP-178
## the naming settings below are only used when the type is applied to a group, will not affect
## existing include/exclude groups
#####

#if the addIncludeExclude and requireInGroups should be enabled, and if the type(s) should be
#auto-created, and used to auto create groups to facilitate include and exclude lists, and require lists
grouperIncludeExclude.use = false
grouperIncludeExclude.requireGroups.use = false

#for requireGroups (groups that the members must be to be in the overall group). name is the name of the
attribute or type
#attributeOrType is either attribute for an attribute underneath the requireInGroups type, or type to be a top
level type
#group is the group to be added in. note attributes are a global namespace, so you might want to use a
naming convention,
#e.g. prefix with "require". description is the tooltip. add as many as you like.
#grouperIncludeExclude.requireGroup.name.0 = requireActiveEmployee
#grouperIncludeExclude.requireGroup.attributeOrType.0 = type
#grouperIncludeExclude.requireGroup.group.0 = school:community:activeEmployee
#grouperIncludeExclude.requireGroup.description.0 = If value is true, members of the overall group must be
an active employee (in the school:community:activeEmployee group). Otherwise, leave this value not filled in.
```

```

#grouperIncludeExclude.requireGroup.name.1 = requireActiveStudent
#grouperIncludeExclude.requireGroup.attributeOrType.1 = attribute
#grouperIncludeExclude.requireGroup.group.1 = school:community:activeStudent
#grouperIncludeExclude.requireGroup.description.1 = If value is true, members of the overall group must be
an active student (in the school:community:activeStudent group). Otherwise leave this value not filled in.

# set some names and tooltips
grouperIncludeExclude.type.name = addIncludeExclude
grouperIncludeExclude.tooltip = Select this type to auto-create other groups which facilitate having include and
exclude list

grouperIncludeExclude.requireGroups.type.name = requireInGroups
grouperIncludeExclude.requireGroups.tooltip = Select this type to auto-create other groups which set up group
math so that other groups can be required for membership (e.g. activeEmployee)

#leave grouperIncludeExclude.andGroups.attributeName blank if you dont want to use this attribute...
#though if you were using it, it wont remove already configured groups
grouperIncludeExclude.requireGroups.attributeName = requireAlsoInGroups
grouperIncludeExclude.requireGroups.attribute.tooltip = Enter in comma separated group path(s). An entity
must be in these groups for it to be in the overall group. e.g. stem1:stem2:group1, stem1:stem3:group2

#suffixes for various include/exclude groups (can use ${space} for space).
#note, these should uniquely identify various parts of the include/exclude.
#i.e. if the grouperIncludeExclude type is applied to a group with a suffix of the include suffix,
#the other groups will not be created...
grouperIncludeExclude.systemOfRecord.extension.suffix = _systemOfRecord
grouperIncludeExclude.include.extension.suffix = _includes
grouperIncludeExclude.exclude.extension.suffix = _excludes
grouperIncludeExclude.systemOfRecordAndIncludes.extension.suffix = _systemOfRecordAndIncludes
grouperIncludeExclude.includesMinusExcludes.extension.suffix = _includesMinusExcludes
#note, put a ${i} in there for where the 1 based index will go
grouperIncludeExclude.requireGroups.extension.suffix = _requireGroups${i}

#suffixes for various include/exclude groups (can use ${space} for space)
grouperIncludeExclude.systemOfRecord.displayExtension.suffix = ${space}system of record
grouperIncludeExclude.include.displayExtension.suffix = ${space}includes
grouperIncludeExclude.exclude.displayExtension.suffix = ${space}excludes
grouperIncludeExclude.systemOfRecordAndIncludes.displayExtension.suffix = ${space}system of record and
includes
grouperIncludeExclude.includesMinusExcludes.displayExtension.suffix = ${space}includes minus excludes
#note, put a ${i} in there for where the 1 based index will go
grouperIncludeExclude.requireGroups.displayExtension.suffix = ${space}requireGroups ${i}

#can use ${extension} as the group extension, or ${displayExtension} for group display extension
grouperIncludeExclude.overall.description = Group containing list of ${displayExtension} after adding the
includes and subtracting the excludes
grouperIncludeExclude.systemOfRecord.description = Group containing list of ${displayExtension} (generally
straight from the system of record) without yet considering manual include or exclude lists
grouperIncludeExclude.include.description = Group containing manual list of includes for group
${displayExtension} which will be added to the system of record list (unless the subject is also in the excludes
group)
grouperIncludeExclude.exclude.description = Group containing manual list of excludes for group
${displayExtension} which will not be in the overall group
grouperIncludeExclude.systemOfRecordAndIncludes.description = Internal utility group for group
${displayExtension} which facilitates the group math for the include and exclude lists
grouperIncludeExclude.includesMinusExclude.description = Internal utility group for group ${displayExtension}
which facilitates includes, excludes, and required groups (e.g. activeEmployee)
#note, put a ${i} in there for where the 1 based index will go
grouperIncludeExclude.requireGroups.description = Internal utility group for group ${displayExtension} which
facilitates required groups (e.g. activeEmployee)

```


Features

- There is a GSH command to do this manually without the types and attributes enabled

```
group.manageIncludesExcludes(grouperSession,isIncludeExclude)
group.manageIncludesExcludes(grouperSession,isIncludeExclude,groupThatMembersMustAlsoBeIn)
group.manageIncludesExcludes(grouperSession,isIncludeExclude,setOfGroupsThatMembersMustAlsoBeIn)
```

- All groups above are auto-created or in some cases auto-fixed when the types or attributes are selected
- Config file entries are validated with descriptive error messages when not filled in
- Built-in hook is installed automatically if enabled
- If overall group originally has list members, then move them to the system of record group so the overall can be converted into a composite group
- Since multiple require groups can be configured, variable substitution occurs in names and descriptions
- See which state we are in (on sor or overall). If the types or attributes are selected on a group with a suffix which is the system of record configured suffix, then the overall group will be created without that suffix. This is especially useful for loader related groups where the system of record group is created first
- Transactions are tested, if a failure occurs in the hook, all actions are rolled back
 - note that since this will run in one transaction, for huge groups (tens of thousands of members) it might cause memory problems...
- If an include exclude group has that type removed, nothing will change. This is because it would be confusing if the include and exclude groups existed and weren't being used. And if they contain members that are useful they should not be deleted. To delete these groups, first take out exclude out of the composite group, then delete the exclude and include groups, then remove the include exclude type and the other groups will be cleaned up.
- If include exclude are selected or require groups, then removed (if legal, see above point about include and exclude), then the overall group will remain with the system of record group as its only member
- Add debug logging to Group class and GroupTypeTupleIncludeExcludeHook class. To see debug info on include exclude and require groups, add this to thelog4j.properties

```
log4j.logger.edu.internet2.middleware.grouper.hooks.examples.GroupTypeTupleIncludeExcludeHook = DEBUG
log4j.logger.edu.internet2.middleware.grouper.Group = DEBUG
```

- Many configurations and changes of state are unit tested
- Auto-create system groups (well, only the wheel group) if configured in grouper.properties, also leave entries to auto create other groups, and assign memberships (e.g. the uiUsers group)

v1.4.0 Grouper Web Services

This page last changed on Oct 13, 2009 by mchzyer@idp.protectnetwork.org.

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Grouper Web Services as of v1.4.0

Introduction

Grouper web services (grouper-ws) is a J2EE web application which exposes common Grouper business logic through SOAP and REST. See [FAQ](#).

To deploy the services, download the warfile and configure the property files (e.g. subject sources, databases, logging, etc). Configure [authentication](#).

Note: there is a command line and java API web service client called [Grouper Client](#)

To implement a web service client:

1. Understand the object model. All grouper-ws services are operations based on simple data structures. The structures support Strings, ints, arrays, and structure references.
 - a. [Core web service API](#)
 - b. [Example structure](#) (only "getters" and "setters" are applicable properties)
 - c. Each operation has many samples (authmated captures, versioned, and up to date). [Here is an example](#)
 - d. Most options has a sensible default (e.g. MemberFilter defaults to All members)
 - e. Lookup objects in various (consistent) ways. e.g. to delete a group, you can pass the name or uuid of the group.
2. Decide if you are using SOAP or REST (this is real REST, not Axis HTTP/XML)
 - a. Both SOAP and REST support the same API
3. Inside SOAP and REST, each operation has two levels of complexity, the normal one, and the Lite one.
 - a. Normal operation: can usually be batched (support a list of inputs, e.g. add multiple groups at once), supports complex inputs (arrays or structures)
 - b. Lite operation: supports only inputs of scalars (no structures, no arrays... only Strings, ints, etc). In REST this also means that the request can be sent via query string only
4. If SOAP:
 - a. Implement based on the [WSDL](#)
 - b. There is a [sample Java client](#) with [sample calls](#)
5. If REST:
 - a. Decide what format you want to send and receive data. grouper-ws supports [XHTML](#), [XML](#), and [JSON](#), as well as query strings for input (in URL or message body)
 - b. There are many [samples](#)

[.NET client development guide](#)

[PHP client development guide](#)

Guidelines For Working With Grouper Web Services

1. There is a bug we are tracking with Axis, where if you skip String params, it will mix up the params. So, if you are passing a param to a web service, make sure you pass empty strings for all null params before the param
2. Code clients with a mindset that the service might change in subtle ways. e.g. a result code might be added (check for success flag element, not success result code), an element might be added in a result object, another input element might be added to end of list, etc. Expect elements to be added in data
3. Make sure there is a property in the client of the URL and version for the service. The version of the service might change the URL (up to service deployer)...

Operations

- [addMember](#): assign a member to a group
 - If already a member, that is ok
 - Accepts batches of members (non-Lite)
 - Accepts flag to say that any members not in batch should be removed (e.g. replace list)
- [deleteMember](#): unassign a member from a group
 - If not a member, that is ok
 - Accepts batches of members (non-Lite)
- [getMembers](#): return the members (including subject data) in a group (from direct or indirect membership)
 - Will accept member filter (All, Effective, Immediate, Composite)
 - Accepts batches of groups (non-Lite)
- [getMemberships](#): under construction
 - Will accept member filter (All, Effective, Immediate, Composite)
 - Accepts batches of subjects and groups (non-Lite)
- [hasMember](#): see if a subject is a member of a group
 - Will return true or false
 - Accepts batches of subject ids or identifiers (returns batches of true's / false's) (non-Lite)
 - Will accept member filter (All, Effective, etc)
 - Can query on field (permission)
- [getGroups](#): list groups for a subject
 - Will accept member filter (All, Effective, etc)
 - Accepts batches of subjects (non-Lite)
- [groupSave](#)
 - Create / update a group
 - Accepts batches of groups (non-Lite)
- [groupDelete](#)
 - Delete a group
 - Accepts batches of groups (non-Lite)
- [getGrouperPrivileges](#)
 - View privileges for a subject and (group or stem)
 - Can view all privileges for the subject and (group or stem) or a specific privilege
- [assignGrouperPrivileges](#)
 - Add or remove a privilege for a subject and (group or stem)
 - Will not fail if the privilege is already assigned or revoked
- [findGroups](#)
 - Can query for groups based on name, uuid, parent stem, or a substring query
 - Can create complex queries with group match (AND, OR, MINUS) (non-Lite)
- [findStems](#)
 - Can query for stems based on name, uuid, parent stem, or a substring query
 - Can create complex queries with group match (AND, OR, MINUS) (non-Lite)
- [stemSave](#)
 - Create / update a stem
 - Accepts batches of stems (non-Lite)
- [stemDelete](#)
 - Delete a stem
 - Accepts batches (non-Lite)
- [memberChangeSubject](#)
 - Change the subject of a current member
 - Accepts batches (non-Lite)

Features

- **API**
 - Batched operations (e.g. add 100 subjects to a group at once). There is a separate server-side max-in-batch param in the grouper-ws.properties.
 - Transaction support (if any fails in one batch request, rollback all in that single batch request)
- **Authentication**
 - Let container or web server handle
 - PKI
 - http-simple-auth
 - Source IP address filtering (TODO)
 - Custom authenticator

- WS-Security
 - PKI
 - Kerberos
- Proxying. The web service can execute operations based on an underlying user, not the authenticating user. Note the authenticating user must have appropriate permissions
- **Error Handling**
 - Error codes and error messages are sent in responses, as well as warnings. In batched mode, batches of response codes are returned. In REST, the http status code is used as well.
- **Clients**
 - Grouper will provide a quick start with Java, and it is up to users to create their own clients. The SOAP and REST are based on the HTTP documents, so any programming language will work
- **Web Service Implementation**
 - Apache Axis for SOAP, and home-grown for REST

Quick start

Checkout the appropriate projects under [grouper-ws](#), read the [README.txt](#) in the grouper-ws/grouper-ws cvs directory

Build Script

The build script for grouper-ws is pretty basic. Generally just do the default (dist). There is also an "ant grouper" target to build a new grouper jar, and "ant quick" to do everything but generate the Axis files (takes 3 minutes).

```
C:\mchyzer\isc\dev\grouper\grouper-ws>ant
Buildfile: build.xml
```

```
dist:
```

```
clean:
```

```
[delete] Deleting directory C:\mchyzer\isc\dev\grouper\grouper-ws\build
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\grouper-ws
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws
```

```
compile:
```

```
[javac] Compiling 10 source files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\grouper-ws
[javac] C:\mchyzer\isc\dev\grouper\grouper-ws\src\grouper-ws\edu\internet2\middleware\grouper
\websservices\GrouperSer
viceServlet.java:33: warning: [deprecation] getEPRForService(java.lang.String,java.lang.String) in
org.apache.axis2.trans
sport.TransportListener has been deprecated
[javac] public class GrouperServiceServlet extends AxisServlet {
[javac]      ^
[javac] 1 warning
```

```
generate-aar:
```

```
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[delete] Deleting directory C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[copy] Copying 13 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\webservices\classes
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\webapp\WEB-INF\services\GrouperService.aar
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws.jar
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\classes
[mkdir] Created dir: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\lib
[copy] Copying 12 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\classes
[copy] Copying 30 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws\WEB-INF\lib
[copy] Copying 11 files to C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws
[jar] Building jar: C:\mchyzer\isc\dev\grouper\grouper-ws\build\dist\grouper-ws.war
```

```
BUILD SUCCESSFUL
```

```
Total time: 22 seconds
```

```
The system cannot find the batch label specified - end
```

```
C:\mchyzer\isc\dev\grouper\grouper-ws>
```

Notice the generate-aar target. This is what makes the axis archive, which is all the classes needed for axis to determine the wsdl, along with the services.xml config file.

Axis is ~40 jars, though most of them are pretty axis specific. There is an ant target which will compress most of these into one jar (axisBundle.jar). Here is the ant help:

```
C:\mchyzer\isc\dev\grouper\grouper-ws>ant help
Buildfile: build.xml
```

help:

```
[echo] Please ensure you have read the documentation -
[echo] and created a build.properties file based on the template provided
[echo]
```

```
[echo] The following targets are available - type the appropriate name:
```

```
[echo]
```

```
[echo] 1) default (dist)
```

```
[echo]     Simply builds, without cleaning, to the webapp.folder
```

```
[echo] 2) clean
```

```
[echo]     Clean the webapp folder, and classfiles, and build
```

```
[echo] 3) generate-aar
```

```
[echo]     Make the axis archive, which is the classfiles and services.xml that axis needs. You need to do
this i
```

```
f you ever change anything that changes the wsdl. You can do this automatically in dist by setting a property
in the bu
```

```
ild.properties
```

```
[echo] 4) generate-axis-bundle-jar
```

```
[echo]     Take all the bundlable axis jars (in lib/axis-bundle), unjar, and jar back up into one jar
```

```
[echo]
```

```
BUILD SUCCESSFUL
```

```
Total time: 0 seconds
```

```
The system cannot find the batch label specified - end
```

```
C:\mchyzer\isc\dev\grouper\grouper-ws>
```

To do's (post 1.4.0)

1. add find subject service
2. make some params to test stuff... (junit to throw exceptions in the middle of tx?)
3. come up with formatter and code style and remove all warnings
4. add logging filter
5. fix javadoc warnings
6. look into axis 1.5 when it is out, see if error fixed, see if samples/wsdl changes, see about enums
7. add metadata service
8. add getGroups with batched groupLookup input
9. add batched privilege service, and add more url options to REST
10. add back in memberships service
11. filter getMember by privileges (find member?)
12. in rest add GET starting points with links to resources
13. improve auto-toString methods in resultMessage
14. look at acegi
15. add ip source filtering to grouper

Add Member

This page last changed on Oct 13, 2009 by mchalyzer@idp.protectnetwork.org.

Description

Add member will add or replace the membership of a group. This affects only direct memberships, not indirect memberships. If the user is already a member of the group it is still a success

Features

- Can assign membership to a "field" or "list", this is a custom type of membership to the group
- Lookup subjects by subject lookup (by id, source, identifier, etc). To add a group to another group, lookup the groupToAdd by putting the group uuid (e.g. fa2dd790-d3f9-4cf4-ac41-bb82e63bff66) in the subject id of the subject lookup. Optionally you can use g:gsa as the source id.
- Lookup groups by group lookup (by name or uuid)
- Returns group / subject information, can be detailed or not
- Can actAs another user

Add member Lite service

- Accepts one group and one member to add direct membership to group
- Documentation: [SOAP](#) (click on addMemberLite), [REST](#) (click on addMemberLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members/10021368
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Add member service

- Accepts one group and many members to add direct membership to group
- Can either add multiple members to a group, or can replace all existing members
- Can operate in one transaction, or can let each membership add in its own separate unit
- Documentation: [SOAP](#) (click on addMember), [REST](#) (click on addMember)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Add or remove grouper privileges

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

"[Add or remove grouper privileges](#)" will add or remove privileges for a subject and (group or stem). If you are adding a privilege and it already exists (immediate privilege), then it will not fail, and it will notify you in the response (still considered a success). If you are removing a privilege, and there was no immediate privileges to remove, it will be a success, and will notify you by response code. If you remove a privilege, and there is an "effective" privilege still there (which means the subject has the privilege, just not directly), it will be a success, and you will be notified via response code.

Features

- Will only edit the privileges the web service user (or actAs) is allowed to see
- Lookup subjects/members by subject lookup (by id, source, identifier, etc)
- Lookup groups/stems by group lookup or stem lookup (name or uuid)
- Returns subject information of the subject
- Returns the group or stem information
- Can actAs another user
- Failsafe, will not fail if adding a privilege that is already there, or removing one that is already gone
- Descriptive response codes give information about the existing privileges

Assign grouper privileges Lite service

- Accepts one subject, one privilege type and name, if allowed, and one group or stem lookup
- Documentation: [SOAP](#) (click on assignGrouperPrivileges), [REST](#) (click on assignGrouperPrivileges)
- For REST, a request body is required (probably via POST)
- REST request (colon is escaped to %3A): PUT (or POST) /grouper-ws/servicesRest/v1_4_000/grouperPrivileges
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Authentication

This page last changed on Sep 02, 2008 by sanjay.vivek@ncl.ac.uk.

Default authentication

Out of the box, grouper-ws uses container authentication (non-rampart). The web.xml protects all services and expects the users to be in the role "grouper_user". For tomcat, in the tomcat-users.xml, just have entries like this, and you are all set:

```
<role rolename="grouper_user"/>
  <user username="jota" password="whatever" roles="grouper_user"/>
  <user username="jobr" password="whatever" roles="grouper_user"/>
  <user username="eldo" password="whatever" roles="grouper_user"/>
```

Note that users to the web service need to be Subjects, and you can configure the default source in the grouper-ws.properties especially if you have subjectId overlap in various sources.

Note the default authentication in grouper-ws is http-basic, so for this and other reasons make sure your deployments of grouper-ws are protected with SSL.

Note that, for some container technologies, container authentication can be externalized in various ways. A common deployment configuration is to externalize tomcat authentication to Apache 2.2+ using the AJP protocol. This permits several popular authentication technologies to be used in conjunction with grouper-ws.

If you do not want to use the servlet container simple auth (even if you front with apache), you need to remove the security settings at the bottom of the web.xml

HTTP basic with kerberos

Grouper-ws comes with an option to authenticate REST or SOAP with HTTP basic auth, but using the user/pass to authenticate to a kerberos kdc. This exists so that kerberos can be used with REST (for SOAP, ws-security would be more secure), and as an example of how to customize the authentication. You should use SSL if you use this authentication. This defeats the purpose of kerberos since it transmits the password over the wire, and it only authenticates to the kdc and not an SSL service, so it might be possible for someone to spoof the kdc. To use this, make the following settings in the grouper-ws.properties (obviously you need to configure the kerberos settings to fit your institution):

```
# to provide custom authentication (instead of the default httpServletRequest.getUserPrincipal())
# for non-Rampart authentication. Class must implement the interface:
# edu.internet2.middleware.grouper.ws.security.WsCustomAuthentication
# class must be fully qualified. e.g. edu.school.whatever.MyAuthenticator
# blank means use default: edu.internet2.middleware.grouper.ws.security.WsGrouperDefaultAuthentication
ws.security.non-rampart.authentication.class =
edu.internet2.middleware.grouper.ws.security.WsGrouperKerberosAuthentication

##### KERBEROS settings, only needed if doing kerberos simple auth
#####
# realm, whatever your realm is, e.g. SCHOOL.EDU
kerberos.realm = SCHOOL.EDU
# address of your kdc, e.g. kdc.school.edu
kerberos.kdc.address = kdc.school.edu
```

Custom authentication plugin

If you want custom authentication (e.g. pass in a token, and decode it), then implement the interface edu.internet2.middleware.grouper.ws.security.WsCustomAuthentication and configure your fully qualified classname in the grouper-ws.properties. The default is an implementation of this interface as an example: edu.internet2.middleware.grouper.ws.security.WsGrouperDefaultAuthentication, which just gets the user from the container: httpServletRequest.getUserPrincipal().getName()

Rampart

Rampart is Jakarta's WS-Security implementation. We have vanilla [Rampart authentication](#) working with grouper-ws (thanks to Sanjay Vivek). Unfortunately it doesn't work out of the box since it seems Rampart and basic auth cannot work together in the web app. If you want to run basic auth and rampart at the same time, you should deploy two separate web apps.

Note the URL for rampart in grouper-ws is the same, it will look like this: /grouper-ws/services/GrouperService

Also, for Rampart, you need custom logic to authenticate users. To use rampart, configure the grouper-ws.properties entry: ws.security.rampart.authentication.class. An example is: edu.internet2.middleware.grouper.ws.security.GrouperWssecSample. Until you configure that, clients will get a 404 http status code. This assumes you are using WSPasswordCallback, if not, just provide your own class directly to the services.xml file (and grouper-ws requires you have an implementation of the interface anyway which won't be executed).

You need to tell grouper that wssec is enabled in the web.xml servlet param (uncomment):

```
<servlet>
  <servlet-name>AxisServlet</servlet-name>
  <display-name>Apache-Axis Servlet</display-name>
  <servlet-class>edu.internet2.middleware.grouper.ws.GrouperServiceAxisServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
  <!-- hint that this is the wssec servlet -->
  <init-param>
    <param-name>wssec</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>
```

Also you need to comment out the container auth in web.xml:

```
<!-- security-constraint>
  <web-resource-collection>
    <web-resource-name>Web services</web-resource-name>
    <url-pattern>/services/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>grouper_user</role-name>
  </auth-constraint>
</security-constraint -->
```

Then you need to enable the correct .aar file.

- If you are using a binary grouper-ws.war, just rename the following two files
 - /WEB-INF/services/GrouperService.aar to /WEB-INF/services/GrouperService.aar.ondeck
 - /WEB-INF/services/GrouperServiceWssec.aar.ondeck to /WEB-INF/services/GrouperServiceWssec.aar
- If you are building, just set the param in the build.properties: webapp.authentication.use.rampart
Here is a sample client

HTTP basic authentication (use)

In the web.xml for the grouper-ws project, protect all services:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Web services</web-resource-name>
    <url-pattern>/services/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <!-- NOTE: This role is not present in the default users file -->
    <role-name>grouper_user</role-name>
  </auth-constraint>
</security-constraint>
```

```

<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Grouper Application</realm-name>
</login-config>

<!-- Security roles referenced by this web application -->
<security-role>
  <description>
    The role that is required to log in to the Manager
    Application
  </description>
  <role-name>grouper_user</role-name>
</security-role>
</web-app>

```

Now send the user and pass in the web service client.
Here is an example with Axis generated clients:

```

GrouperServiceStub stub = new GrouperServiceStub(
    "http://localhost:8090/grouper-ws/services/GrouperService");
Options options = stub._getServiceClient().getOptions();
HttpTransportProperties.Authenticator auth = new HttpTransportProperties.Authenticator();
auth.setUsername("user");
auth.setPassword("pass");

options.setProperty(HTTPConstants.AUTHENTICATE, auth);

```

Here is an example with a manual HttpClient:

```

HttpClient httpClient = new HttpClient();
GetMethod getMethod = new GetMethod(
    "http://localhost:8091/grouper-ws/services/GrouperService/addMemberSimple?
groupName=aStem:aGroup&subjectId=10021368&actAsSubjectId=GrouperSystem");

httpClient.getParams().setAuthenticationPreemptive(true);
Credentials defaultcreds = new UsernamePasswordCredentials("user", "pass");
httpClient.getState().setCredentials(new AuthScope("localhost", 8091), defaultcreds);

httpClient.executeMethod(getMethod);

```

ActAs configuration

To enable web service users to act as another user (proxy), enable the setting in the grouper-ws grouper.properties

```

# Web service users who are in the following group can use the actAs field to act as someone else
ws.act.as.group = aStem:aGroup

```

If you specify a group name in there, you can pass in the actAs field if you connect to the web service as a user who is in the ws.act.as.group group. Here is an example with the axis generated client.

```

//set the act as id
WsSubjectLookup actAsSubject = WsSubjectLookup.class.newInstance();
actAsSubject.setSubjectId("GrouperSystem");
addMember.setActAsSubjectLookup(actAsSubject);

```

There are advanced settings, you can specify multiple groups in the grouper-ws.properties, and you can even limit who the users can act as (in a specific group).

Delete Member

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Delete member will delete or replace the membership of a group. This affects only direct memberships, not indirect memberships. If the user is in an indirect membership, this is still a success

Features

- Can unassign membership to a "field" or "list", this is a custom type of membership to the group
- Lookup subjects by subject lookup (by id, source, identifier, etc)
- Lookup groups by group lookup (by name or uuid)
- Returns group / subject information, can be detailed or not
- Can actAs another user

Delete member Lite service

- Accepts one group and one member to delete direct membership to group
- Documentation: [SOAP](#) (click on deleteMemberLite), [REST](#) (click on deleteMemberLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): DELETE /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members/10021368
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Delete member service

- Accepts one group and many members to delete direct membership to group
- Can either delete multiple members to a group, or can replace all existing members
- Can operate in one transaction, or can let each membership delete in its own separate unit
- Documentation: [SOAP](#) (click on deleteMember), [REST](#) (click on deleteMember)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Find Groups

This page last changed on Oct 13, 2009 by mchzyer@idp.protectnetwork.org.

Description

Find groups search for groups based on name, attribute, parent stem, etc. Can build queries with group math (AND / OR / MINUS)

Features

- Use [query type](#) to build one query object
- For AND|OR|MINUS you can link up multiple queries into one
- Returns groups, can be detailed or not
- Can actAs another user

Find groups Lite service

- Accepts one query to search (cannot use group math)
- Documentation: [SOAP](#) (click on findGroupsLite), [REST](#) (click on findGroupsLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_3_000/groups
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Find groups service

- Accepts multiple query objects in a graph (can use group math)
- Documentation: [SOAP](#) (click on findGroups), [REST](#) (click on findGroups)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/groups
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Find Stems

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Find stems search for stems based on name, attribute, parent stem, etc. Can build queries with group math (AND / OR / MINUS)

Features

- Use [query type](#) to build one query object
- For AND|OR|MINUS you can link up multiple queries into one
- Returns stems
- Can actAs another user

Find stems Lite service

- Accepts one query to search (cannot use group math)
- Documentation: [SOAP](#) (click on findStemsLite), [REST](#) (click on findStemsLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_3_000/stems
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Find stems service

- Accepts multiple query objects in a graph (can use group math)
- Documentation: [SOAP](#) (click on findStems), [REST](#) (click on findStems)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/stems
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Get grouper privileges

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

"[Get grouper privileges](#)" will retrieve the privileges for a subject and or (group or stem). If you dont specify the privilege name, you will get all permissions for the user and or (group or stem). If you specify the subject, (group or stem) and privilege you are looking for, you will get also get the response in the return code (which is an HTTP header). You must specify a subject or stem or group. You cannot specify a group and a stem at once.

Features

- Will only get the privileges the user (or actAs) is allowed to see
- Lookup subjects/members by subject lookup (by id, source, identifier, etc)
- Lookup groups/stems by group lookup or stem lookup (name or uuid)
- Returns subject information of the subject
- Returns the group or stem information
- Can actAs another user

Get grouper privileges Lite service

- Accepts one subject and one group or stem lookup
- Documentation: [SOAP](#) (click on getGrouperPrivileges), [REST](#) (click on getGrouperPrivileges)
- For REST, a request body is required (probably via POST)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_4_000/grouperPrivileges
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Example grouper client output

```
C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --groupName=aStem:aGroup
Index 0: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: admin
Index 1: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: read
Index 2: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: update
Index 3: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: view
Index 4: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperAll: access: read
Index 5: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperAll: access: view
Index 6: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperSystem: access: admin
Index 7: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperSystem: access: read
Index 8: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperSystem: access: update
Index 9: success: T: code: SUCCESS: group: aStem:aGroup: subject: GrouperSystem: access: view
Index 10: success: T: code: SUCCESS: group: aStem:aGroup: subject: test.subject.0: access: admin
Index 11: success: T: code: SUCCESS: group: aStem:aGroup: subject: test.subject.0: access: read
Index 12: success: T: code: SUCCESS: group: aStem:aGroup: subject: test.subject.0: access: view
```

```
C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --subjectId=10021368
Index 0: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: create
Index 1: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: stem
Index 2: success: T: code: SUCCESS: stem: aStem:aStem0: subject: 10021368: naming: create
Index 3: success: T: code: SUCCESS: stem: aStem:aStem0: subject: 10021368: naming: stem
Index 4: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: admin
Index 5: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: read
Index 6: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: update
Index 7: success: T: code: SUCCESS: group: aStem:aGroup: subject: 10021368: access: view
Index 8: success: T: code: SUCCESS: group: aStem:activeEmployee: subject: 10021368: access: admin
Index 9: success: T: code: SUCCESS: group: aStem:activeEmployee: subject: 10021368: access: read
Index 10: success: T: code: SUCCESS: group: aStem:activeEmployee: subject: 10021368: access: update
Index 11: success: T: code: SUCCESS: group: aStem:activeEmployee: subject: 10021368: access: view
Index 12: success: T: code: SUCCESS: group: aStem:activeStudent: subject: 10021368: access: admin
Index 13: success: T: code: SUCCESS: group: aStem:activeStudent: subject: 10021368: access: read
Index 14: success: T: code: SUCCESS: group: aStem:activeStudent: subject: 10021368: access: update
```

Index 15: success: T: code: SUCCESS: group: aStem:activeStudent: subject: 10021368: access: view
 Index 16: success: T: code: SUCCESS: group: etc:sysadmingroup: subject: 10021368: access: admin
 Index 17: success: T: code: SUCCESS: group: etc:sysadmingroup: subject: 10021368: access: read
 Index 18: success: T: code: SUCCESS: group: etc:sysadmingroup: subject: 10021368: access: update
 Index 19: success: T: code: SUCCESS: group: etc:sysadmingroup: subject: 10021368: access: view
 Index 20: success: T: code: SUCCESS: group: etc:webServiceActAsGroup: subject: 10021368: access: admin
 Index 21: success: T: code: SUCCESS: group: etc:webServiceActAsGroup: subject: 10021368: access: read
 Index 22: success: T: code: SUCCESS: group: etc:webServiceActAsGroup: subject: 10021368: access: update
 Index 23: success: T: code: SUCCESS: group: etc:webServiceActAsGroup: subject: 10021368: access: view
 Index 24: success: T: code: SUCCESS: group: etc:webServiceClientUsers: subject: 10021368: access: admin
 Index 25: success: T: code: SUCCESS: group: etc:webServiceClientUsers: subject: 10021368: access: read
 Index 26: success: T: code: SUCCESS: group: etc:webServiceClientUsers: subject: 10021368: access: update
 Index 27: success: T: code: SUCCESS: group: etc:webServiceClientUsers: subject: 10021368: access: view
 Index 28: success: T: code: SUCCESS: group: penn:etc:sysAdminGroup: subject: 10021368: access: admin
 Index 29: success: T: code: SUCCESS: group: penn:etc:sysAdminGroup: subject: 10021368: access: read
 Index 30: success: T: code: SUCCESS: group: penn:etc:sysAdminGroup: subject: 10021368: access: update
 Index 31: success: T: code: SUCCESS: group: penn:etc:sysAdminGroup: subject: 10021368: access: view
 Index 32: success: T: code: SUCCESS: group: penn:etc:userInterfaceUsers: subject: 10021368: access: admin
 Index 33: success: T: code: SUCCESS: group: penn:etc:userInterfaceUsers: subject: 10021368: access: read
 Index 34: success: T: code: SUCCESS: group: penn:etc:userInterfaceUsers: subject: 10021368: access: update
 Index 35: success: T: code: SUCCESS: group: penn:etc:userInterfaceUsers: subject: 10021368: access: view
 Index 36: success: T: code: SUCCESS: group: penn:etc:webServiceActAsGroup: subject: 10021368: access: admin
 Index 37: success: T: code: SUCCESS: group: penn:etc:webServiceActAsGroup: subject: 10021368: access: read
 Index 38: success: T: code: SUCCESS: group: penn:etc:webServiceActAsGroup: subject: 10021368: access: update
 Index 39: success: T: code: SUCCESS: group: penn:etc:webServiceActAsGroup: subject: 10021368: access: view
 Index 40: success: T: code: SUCCESS: group: penn:etc:webServiceClientUsers: subject: 10021368: access: admin
 Index 41: success: T: code: SUCCESS: group: penn:etc:webServiceClientUsers: subject: 10021368: access: read
 Index 42: success: T: code: SUCCESS: group: penn:etc:webServiceClientUsers: subject: 10021368: access: update
 Index 43: success: T: code: SUCCESS: group: penn:etc:webServiceClientUsers: subject: 10021368: access: view

C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --stemName=aStem
 Index 0: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: create
 Index 1: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: stem
 Index 2: success: T: code: SUCCESS: stem: aStem: subject: GrouperSystem: naming: stem
 Index 3: success: T: code: SUCCESS: stem: aStem: subject: test.subject.0: naming: create
 Index 4: success: T: code: SUCCESS: stem: aStem: subject: test.subject.0: naming: stem

C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --subjectId=10021368 --privilegeType=naming
 Index 0: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: create
 Index 1: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: stem
 Index 2: success: T: code: SUCCESS: stem: aStem:aStem0: subject: 10021368: naming: create
 Index 3: success: T: code: SUCCESS: stem: aStem:aStem0: subject: 10021368: naming: stem

C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --stemName=aStem --privilegeName=create
 Index 0: success: T: code: SUCCESS: stem: aStem: subject: 10021368: naming: create
 Index 1: success: T: code: SUCCESS: stem: aStem: subject: test.subject.0: naming: create

C:\temp>java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --stemName=aStem --privilegeName=create --subjectId=10021368
 Index 0: success: T: code: SUCCESS_ALLOWED: stem: aStem: subject: 10021368: naming: create

Get Groups

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Get groups will get the groups that a subject is in

Features

- Can base member list based on memberfilter (e.g. All, Immediate, Effective, Composite)
- Lookup subjects by subject lookup (by id, source, identifier, etc)
- Lookup groups by group lookup (by name or uuid)
- Returns group / subject information, can be detailed or not
- Can actAs another user

Get groups Lite service

- Accepts one subject to list groups
- Documentation: [SOAP](#) (click on getGroupsLite), [REST](#) (click on getGroupsLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_3_000/subjects/10021368
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Get groups service

- Accepts multiple subjects to retrieve multiple lists of groups
- Documentation: [SOAP](#) (click on getGroups), [REST](#) (click on getGroups)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/subjects
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Get Members

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Get members will retrieve subjects assigned to a group.

Features

- Can base member list based on memberfilter (e.g. All, Immediate, Effective)
- Lookup subjects by subject lookup (by id, source, identifier, etc)
- Lookup groups by group lookup (by name or uuid)
- Returns group / subject information, can be detailed or not
- Can actAs another user

Get members Lite service

- Accepts one group to get members for
- Documentation: [SOAP](#) (click on getMembersLite), [REST](#) (click on getMembersLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Get members service

- Accepts multiple groups to retrieve lists of lists of subjects
- Documentation: [SOAP](#) (click on getMembers), [REST](#) (click on getMembers)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Get Memberships

This page last changed on Feb 04, 2008 by [mchyszer](#).

Description

There are two getMemberships services. One can take complex inputs, one takes only strings, and is very lightweight and does not require XML input if using loose-REST. Both can be used in loose-REST or SOAP, but the lightweight one will be easier in loose-REST (if no advanced features are needed). See below for:

1. An example of the loose-REST service
2. An example of the SOAP service

Get Memberships supports the following:

1. GroupFinder: based on extension and stem extensions (optional)
2. ActAsSubject: to proxy based on a logged in user (for example)
3. Returns a status for success/error, error messages, codes for the overall request
4. Filter for All, Immediate, Effective, Composite
5. Clients passes a flag to indicate whether the Subject data (in addition to id) needs to be queried and returned
6. Returns memberships based on filter, and their id and type. If extended data is requested, then the name, description, and up to 3 attributes can be returned (based on grouper-ws.properties)

Get Memberhips (simple) supports the following:

1. Find subject based on id or identifier
2. Find group based on extension or uuid
3. ActAsSubjectId: to proxy based on a logged in user (for example)
4. Returns a status for success/error, error messages, codes for the overall request
5. Filter for All, Immediate, Effective, Composite
6. Clients passes a flag to indicate whether the Subject data (in addition to id) needs to be queried and returned
7. Returns memberships based on filter, and their id and type. If extended data is requested, then the name, description, and up to 3 attributes can be returned (based on grouper-ws.properties)
8. Returns the same structure as the non-simple one

Get Memberships Client Example SOAP With Axis Generated Objects

Note: This can easily be called from REST as well by manipulating the XML, but this is the Java generated client:

```
/**
 *
 */
package edu.internet2.middleware.grouper.webservicesClient;

import org.apache.axis2.client.Options;
import org.apache.axis2.transport.http.HTTPConstants;
import org.apache.axis2.transport.http.HttpTransportProperties;
import org.apache.commons.lang.builder.ToStringBuilder;

import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.GetMemberships;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGetMembersResult;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGetMembershipsResult;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGetMembershipsResults;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGroupLookup;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsSubjectLookup;

/**
 *
 * @author mchyszer
 */
```

```

public class RunGrouperServiceGetMemberships {
    /**
     * @param args
     */
    public static void main(String[] args) {
        getMemberships();
    }

    /**
     *
     */
    public static void getMemberships() {
        try {
            GrouperServiceStub stub = new GrouperServiceStub(
                "http://localhost:8091/grouper-ws/services/GrouperService");
            Options options = stub._getServiceClient().getOptions();
            HttpTransportProperties.Authenticator auth = new HttpTransportProperties.Authenticator();
            auth.setUsername("GrouperSystem");
            auth.setPassword("pass");

            options.setProperty(HTTPConstants.AUTHENTICATE, auth);

//            options.setProperty(Constants.Configuration.ENABLE_REST,
//                Constants.VALUE_TRUE);

            GetMemberships getMembers = GetMemberships.class.newInstance();

            // set the act as id
            WsSubjectLookup actAsSubject = WsSubjectLookup.class.newInstance();
            actAsSubject.setSubjectId("GrouperSystem");
            getMembers.setActAsSubjectLookup(actAsSubject);

            WsGroupLookup wsGroupLookup = WsGroupLookup.class.newInstance();
            wsGroupLookup.setGroupName("aStem:aGroup");
            getMembers.setWsGroupLookup(wsGroupLookup);

            getMembers.setMembershipFilter("All");
            getMembers.setRetrieveExtendedSubjectData("true");

            WsGetMembershipsResults wsGetMembershipsResults = stub.getMemberships(getMembers)
                .get_return();

            System.out.println(ToStringBuilder.reflectionToString(
                wsGetMembershipsResults));

            WsGetMembershipsResult[] wsGetMembershipsResultArray = wsGetMembershipsResults.getResults();

            for (WsGetMembershipsResult wsGetMembershipsResult : wsGetMembershipsResultArray) {
                System.out.println(ToStringBuilder.reflectionToString(
                    wsGetMembershipsResult));
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}

```

Request XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:getMemberships xmlns:ns1="http://webservices.grouper.middleware.internet2.edu/xsd">
      <ns1:wsGroupLookup>
        <ns1:groupName>aStem:aGroup</ns1:groupName>
      </ns1:wsGroupLookup>
    </ns1:getMemberships>
  </soapenv:Body>
</soapenv:Envelope>

```

```

    <ns1:membershipFilter>All</ns1:membershipFilter>
    <ns1:retrieveExtendedSubjectData>true</ns1:retrieveExtendedSubjectData>
    <ns1:actAsSubjectLookup>
      <ns1:subjectId>GrouperSystem</ns1:subjectId>
    </ns1:actAsSubjectLookup>
  </ns1:getMemberships>
</soapenv:Body>
</soapenv:Envelope>

```

Response XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns:getMembershipsResponse xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd">
      <ns:return type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResults">
        <ns:attributeName0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:attributeName1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:attributeName2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:resultCode>SUCCESS</ns:resultCode>
        <ns:resultMessage></ns:resultMessage>
        <ns:results type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResult">
          <ns:attribute0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:attribute1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:attribute2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:createTime>2008/02/03 00:11:03.861</ns:createTime>
          <ns:depth>0</ns:depth>
          <ns:groupName>aStem:aGroup</ns:groupName>
          <ns:listName>members</ns:listName>
          <ns:listType>list</ns:listType>
          <ns:membershipId>b1a6fe70-9f66-4b7d-8d46-855f9451d112</ns:membershipId>
          <ns:membershipType>immediate</ns:membershipType>
          <ns:stemName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:subjectDescription>GrouperSystem</ns:subjectDescription>
          <ns:subjectId>GrouperSystem</ns:subjectId>
          <ns:subjectName>GrouperSystem</ns:subjectName>
          <ns:subjectType>application</ns:subjectType>
        </ns:results>
        <ns:success>T</ns:success>
      </ns:return>
    </ns:getMembershipsResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

Get Memberships REST Example

```

package edu.internet2.middleware.grouper.webservicesClient;

import java.io.InputStream;
import java.io.Reader;

import org.apache.commons.httpclient.Credentials;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.methods.RequestEntity;
import org.apache.commons.httpclient.methods.StringRequestEntity;
import org.apache.commons.io.IOUtils;

/**
 * @author mchzyer
 */
public class RunGrouperServiceNonAxisGetMemberships {

```

```

/**
 *
 */
@SuppressWarnings("unchecked")
public static void getMembershipsRest() {
    //lets load this into jdom, since it is xml
    Reader xmlReader = null;

    try {
        HttpClient httpClient = new HttpClient();
        PostMethod method = new PostMethod(
            "http://localhost:8091/grouper-ws/services/GrouperService");

        method.setRequestHeader("Content-Type", "application/xml; charset=UTF-8");
        httpClient.getParams().setAuthenticationPreemptive(true);
        Credentials defaultcreds = new UsernamePasswordCredentials("GrouperSystem", "pass");
        httpClient.getState().setCredentials(new AuthScope("localhost", 8091), defaultcreds);
        String xml = "<ns1:getMemberships xmlns:ns1=\"http://
webservices.grouper.middleware.internet2.edu/xsd\">\" +
            "<ns1:wsGroupLookup><ns1:groupName>aStem:aGroup</ns1:groupName>\" +
            "</ns1:wsGroupLookup><ns1:membershipFilter>All</ns1:membershipFilter>\" +
            "<ns1:retrieveExtendedSubjectData>true</ns1:retrieveExtendedSubjectData>\" +
            "<ns1:actAsSubjectLookup><ns1:subjectId>GrouperSystem</ns1:subjectId></
ns1:actAsSubjectLookup>\" +
            "</ns1:getMemberships>\";
        RequestEntity requestEntity = new StringRequestEntity(xml);
        method.setRequestEntity(requestEntity);
        httpClient.executeMethod(method);

        int statusCode = method.getStatusCode();

        // see if request worked or not
        if (statusCode != 200) {
            throw new RuntimeException("Bad response from web service: " +
                statusCode);
        }
        //there is a getResponseAsString, but it logs a warning each time...
        InputStream inputStream = method.getResponseBodyAsStream();
        String response = null;
        try {
            response = IOUtils.toString(inputStream);
        } finally {
            IOUtils.closeQuietly(inputStream);
        }

        System.out.println(response);
        //parse XML here
    } catch (Exception e) {
        throw new RuntimeException(e);
    } finally {
        try {
            if (xmlReader != null) {xmlReader.close();}
        } catch (Exception e) {
        }
    }
}

/**
 * @param args
 */
@SuppressWarnings("unchecked")
public static void main(String[] args) {
    getMembershipsRest();
}

```

```
}
```

Request XML:

```
<ns1:getMemberships xmlns:ns1="http://webservices.grouper.middleware.internet2.edu/xsd">
  <ns1:wsGroupLookup>
    <ns1:groupName>aStem:aGroup</ns1:groupName>
  </ns1:wsGroupLookup>
  <ns1:membershipFilter>All</ns1:membershipFilter>
  <ns1:retrieveExtendedSubjectData>true</ns1:retrieveExtendedSubjectData>
  <ns1:actAsSubjectLookup>
    <ns1:subjectId>GrouperSystem</ns1:subjectId>
  </ns1:actAsSubjectLookup>
</ns1:getMemberships>
```

Response XML:

```
<ns:getMembershipsResponse xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd">
  <ns:return type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResults">
    <ns:attributeName0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:attributeName1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:attributeName2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:resultCode>SUCCESS</ns:resultCode>
    <ns:resultMessage></ns:resultMessage>
    <ns:results type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResult">
      <ns:attribute0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:attribute1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:attribute2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:createTime>2008/02/03 00:11:03.861</ns:createTime>
      <ns:depth>0</ns:depth>
      <ns:groupName>aStem:aGroup</ns:groupName>
      <ns:listName>members</ns:listName>
      <ns:listType>list</ns:listType>
      <ns:membershipId>b1a6fe70-9f66-4b7d-8d46-855f9451d112</ns:membershipId>
      <ns:membershipType>immediate</ns:membershipType>
      <ns:stemName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:subjectDescription>GrouperSystem</ns:subjectDescription>
      <ns:subjectId>GrouperSystem</ns:subjectId>
      <ns:subjectName>GrouperSystem</ns:subjectName>
      <ns:subjectType>application</ns:subjectType>
    </ns:results>
    <ns:success>T</ns:success>
  </ns:return>
</ns:getMembershipsResponse>
```

Get Memberships Simple Axis SOAP Generated Objects Example

```
/**
 *
 */
package edu.internet2.middleware.grouper.webservicesClient;

import org.apache.axis2.client.Options;
import org.apache.axis2.transport.http.HTTPConstants;
import org.apache.axis2.transport.http.HttpTransportProperties;
import org.apache.commons.lang.builder.ToStringBuilder;

import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.GetMembershipsSimple;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGetMembershipsResult;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsGetMembershipsResults;

/**
 *
 * @author mchzyer
 */
```

```

*/
public class RunGrouperServiceGetMembershipsSimple {
    /**
     *
    */
    public static void getMembershipsSimple() {
        try {
            GrouperServiceStub stub = new GrouperServiceStub(
                "http://localhost:8091/grouper-ws/services/GrouperService");
            Options options = stub._getServiceClient().getOptions();
            HttpTransportProperties.Authenticator auth = new HttpTransportProperties.Authenticator();
            auth.setUsername("GrouperSystem");
            auth.setPassword("pass");

            options.setProperty(HTTPConstants.AUTHENTICATE, auth);
            options.setProperty(HTTPConstants.SO_TIMEOUT, new Integer(3600000));
            options.setProperty(HTTPConstants.CONNECTION_TIMEOUT,
                new Integer(3600000));

            //options.setProperty(Constants.Configuration.ENABLE_REST,
            //    Constants.VALUE_TRUE);
            GetMembershipsSimple getMembershipsSimple = GetMembershipsSimple.class.newInstance();

            // set the act as id
            getMembershipsSimple.setActAsSubjectId("GrouperSystem");

            getMembershipsSimple.setGroupName("aStem:aGroup");
            getMembershipsSimple.setGroupUuid("");
            getMembershipsSimple.setMembershipFilter("All");
            getMembershipsSimple.setRetrieveExtendedSubjectData("true");

            WsGetMembershipsResults wsGetMembershipsResults =
            stub.getMembershipsSimple(getMembershipsSimple)
                .get_return();

            System.out.println(ToStringBuilder.reflectionToString(
                wsGetMembershipsResults));

            WsGetMembershipsResult[] wsGetMembershipsResultArray = wsGetMembershipsResults.getResults();

            for (WsGetMembershipsResult wsGetMembershipsResult : wsGetMembershipsResultArray) {
                System.out.println(ToStringBuilder.reflectionToString(
                    wsGetMembershipsResult));
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /**
     * @param args
    */
    public static void main(String[] args) {
        getMembershipsSimple();
    }
}

```

Request XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:getMembershipsSimple xmlns:ns1="http://webservices.grouper.middleware.internet2.edu/xsd">
      <ns1:groupName>aStem:aGroup</ns1:groupName>
      <ns1:groupUuid></ns1:groupUuid>
      <ns1:membershipFilter>All</ns1:membershipFilter>
    </ns1:getMembershipsSimple>
  </soapenv:Body>
</soapenv:Envelope>

```

```

    <ns1:retrieveExtendedSubjectData>true</ns1:retrieveExtendedSubjectData>
    <ns1:actAsSubjectId>GrouperSystem</ns1:actAsSubjectId>
  </ns1:getMembershipsSimple>
</soapenv:Body>
</soapenv:Envelope>

```

Response XML:

```

<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns:getMembershipsSimpleResponse xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd">
      <ns:return type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResults">
        <ns:attributeName0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:attributeName1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:attributeName2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
        <ns:resultCode>SUCCESS</ns:resultCode>
        <ns:resultMessage></ns:resultMessage>
        <ns:results type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResult">
          <ns:attribute0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:attribute1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:attribute2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:createTime>2008/02/03 00:11:03.861</ns:createTime>
          <ns:depth>0</ns:depth>
          <ns:groupName>aStem:aGroup</ns:groupName>
          <ns:listName>members</ns:listName>
          <ns:listType>list</ns:listType>
          <ns:membershipId>b1a6fe70-9f66-4b7d-8d46-855f9451d112</ns:membershipId>
          <ns:membershipType>immediate</ns:membershipType>
          <ns:stemName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
          <ns:subjectDescription>GrouperSystem</ns:subjectDescription>
          <ns:subjectId>GrouperSystem</ns:subjectId>
          <ns:subjectName>GrouperSystem</ns:subjectName>
          <ns:subjectType>application</ns:subjectType>
        </ns:results>
        <ns:success>T</ns:success>
      </ns:return>
    </ns:getMembershipsSimpleResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

Get Memberships Simple Rest Client Example

Note: This can be called from SOAP as well as REST (shown)

```

package edu.internet2.middleware.grouper.webservicesClient;

import java.io.Reader;

import org.apache.commons.httpclient.Credentials;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.UsernamePasswordCredentials;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.GetMethod;

/**
 *
 * @author mchzyer
 */
public class RunGrouperServiceNonAxisGetMembershipsSimple {

  /**
   * get members simple web service with REST
   */

```



```

public static void getMembershipsSimpleRest() {
    Reader xmlReader = null;
    try {
        HttpClient httpClient = new HttpClient();

        GetMethod method = new GetMethod(
            "http://localhost:8091/grouper-ws/services/GrouperService/getMembershipsSimple?
groupName=aStem:aGroup" +

"&groupUuid=&membershipFilter=All&retrieveExtendedSubjectData=true&actAsSubjectId=GrouperSystem");

        httpClient.getParams().setAuthenticationPreemptive(true);
        Credentials defaultcreds = new UsernamePasswordCredentials("GrouperSystem", "pass");
        httpClient.getState().setCredentials(new AuthScope("localhost", 8091), defaultcreds);

        httpClient.executeMethod(method);

        int statusCode = method.getStatusCode();

        // see if request worked or not
        if (statusCode != 200) {
            throw new RuntimeException("Bad response from web service: " +
                statusCode);
        }

        String response = method.getResponseBodyAsString();

        System.out.println(response);
    } catch (Exception e) {
        throw new RuntimeException(e);
    } finally {
        try {
            xmlReader.close();
        } catch (Exception e) {
        }
    }
}

}

/**
 * @param args
 */
@SuppressWarnings("unchecked")
public static void main(String[] args) {
    getMembershipsSimpleRest();
}
}

```

XML request: none (data is in the URL)

XML response:

```

<ns:getMembershipsSimpleResponse xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd">
  <ns:return type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResults">
    <ns:attributeName0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:attributeName1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:attributeName2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
    <ns:resultCode>SUCCESS</ns:resultCode>
    <ns:resultMessage></ns:resultMessage>
    <ns:results type="edu.internet2.middleware.grouper.webservices.WsGetMembershipsResult">
      <ns:attribute0 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:attribute1 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:attribute2 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
      <ns:createTime>2008/02/03 00:11:03.861</ns:createTime>
      <ns:depth>0</ns:depth>
      <ns:groupName>aStem:aGroup</ns:groupName>
    
```

```
<ns:listName>members</ns:listName>
<ns:listType>list</ns:listType>
<ns:membershipId>b1a6fe70-9f66-4b7d-8d46-855f9451d112</ns:membershipId>
<ns:membershipType>immediate</ns:membershipType>
<ns:stemName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true" />
<ns:subjectDescription>GrouperSystem</ns:subjectDescription>
<ns:subjectId>GrouperSystem</ns:subjectId>
<ns:subjectName>GrouperSystem</ns:subjectName>
<ns:subjectType>application</ns:subjectType>
</ns:results>
<ns:success>T</ns:success>
</ns:return>
</ns:getMembershipsSimpleResponse>
```

Group Delete

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Group delete will insert or update a group's uuid, extension, display name, or description (with restrictions)

Features

- If group does not exist, the call will not fail (special result code)
- Lookup group to delete by group lookup (by name or uuid)
- Returns group, can be detailed or not
- Can actAs another user

Group delete Lite service

- Accepts one group to delete
- Documentation: [SOAP](#) (click on groupDeleteLite), [REST](#) (click on groupDeleteLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): DELETE /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Group delete service

- Accepts multiple groups to delete
- Documentation: [SOAP](#) (click on groupDelete), [REST](#) (click on groupDelete)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/groups
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Grouper WS Lite - REST input - output XHTML - XML - JSON

This page last changed on Mar 23, 2008 by [mchzyer](#).

This document describes the strategy for parsing and generating XHTML, XML, JSON for Grouper WS Lite / REST

Summary

There is a custom automatic converter for the parsing and generation of XHTML. If we want to eventually support XML / JSON / whatever, we can plug that in. The parsing is very flexible, and will give warnings if things are not as expected (e.g. unexpected element / attribute). It is also possible to hose the parser (e.g. send the wrong type for a field). We can also add strict mode if we like so that invalid input (though preparing for WS upgrades) will be ok (e.g. unexpected element name or missing a required attribute).

For XML the same beans are translated with default XStream.

For JSON the same beans are translated with default XStream and Jettison.

Use

The server needs to know the request and response content type. If sending data in the request, if it is not HTTP params, then set the Content-type http header. Must be one of the content types specified in the javadoc for the enum `WsLiteRequestContentType`. Currently the valid values are: `application/xhtml+xml`, `text/xml`, `text/x-json`, and for http params (either dont set content type or set to `application/x-www-form-urlencoded`).

The response content type will be the same as the request content type. If the request content type is http params (null or form encoded), then the response will be whatever is specified in the `grouper-ws.properties`, and it defaults to `xhtml`. If the client wants to override this decision (to either get a different content type than was requested, or to specify a different than the one specified in the `grouper-ws.properties`), then it can be placed in the url.

e.g. here is a url where the content type is the same as request or what is configured in `grouper-ws.properties`: http://localhost/grouper-ws/servicesLite/v3_0_000/group/a:b

e.g. here is a url where the request is an http param request, and the configured response content type is `xhtml`, but the client wants `xml`: http://localhost/grouper-ws/servicesLite/*xml*/v3_0_000/group/a:b?includeGroupDetail=T

Guidelines

When working with the XHTML, please prepare for upgrades on the server. e.g. when parsing assume there might be different types of elements, new attributes, etc. If there is something not expected, then probably ignore it (unless something is badly malformed or something). Assume that things will be added to the service, and not changed or removed (unless there is a drastic upgrade). Also, the XML/XHTML/JSON is not indented, though this shouldnt matter. The XHTML should be valid based on the strict doctype (can validate from validator.w3.org). The XML and JSON should obviously be valid

Mapping of Java Object to XHTML

The objects used for the Grouper SOAP / XML-HTTP web service will be used for this web service also (though they dont have to be), it supports beans based on javabean properties (read / getters, write / setters). It supports only:

1. String fields
2. int fields
3. String arrays
4. int arrays
5. Bean fields
6. Bean arrays
7. Will not work with circular references

Will throw exception if something is not right... Does not support any other structures. Inheritance is not supported. Use the objects `WsXhtmlOutputConverter`, and `WsXhtmlInputConverter` once and throw away.

Each Object will be written to a div or li (if in list). Each scalar (currently only Strings or ints) will be written to p tags or li tags for lists. Each array will be a ul with li's for each item. Each div (except the root), p, and ul will have a "class" attribute of the fieldName. Each div and li (which is a bean and not scalar) will have a "title" attribute which is the simple name of the java class (non-fully qualified).

Null and the empty string are interchangeable and are represented as an empty element <p />

Example

For these three classes:

```
/*
 * @author mchzyer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;

/**
 *
 */
public class BeanGrandparent {

    /** field 1 */
    private String field1;

    /** field 2*/
    private String field2;

    /** string array */
    private String[] stringArray;

    /** field */
    private BeanParent beanParent;

    /** field */
    private BeanParent[] beanParents;

    /**
     * empty
     */
    public BeanGrandparent() {
        //empty
    }

    /**
     * @param _field1
     * @param _field2
     * @param _stringArray
     * @param _beanParent
     * @param _beanParents
     */
    public BeanGrandparent(String _field1, String _field2, String[] _stringArray,
        BeanParent _beanParent, BeanParent[] _beanParents) {
        super();
        this.field1 = _field1;
        this.field2 = _field2;
        this.stringArray = _stringArray;
        this.beanParent = _beanParent;
        this.beanParents = _beanParents;
    }

    /**
     * @return the field1
     */
}
```

```

public String getField1() {
    return this.field1;
}

/**
 * @param _field1 the field1 to set
 */
public void setField1(String _field1) {
    this.field1 = _field1;
}

/**
 * @return the field2
 */
public String getField2() {
    return this.field2;
}

/**
 * @param _field2 the field2 to set
 */
public void setField2(String _field2) {
    this.field2 = _field2;
}

/**
 * @return the stringArray
 */
public String[] getStringArray() {
    return this.stringArray;
}

/**
 * @param _stringArray the stringArray to set
 */
public void setStringArray(String[] _stringArray) {
    this.stringArray = _stringArray;
}

/**
 * @return the beanParent
 */
public BeanParent getBeanParent() {
    return this.beanParent;
}

/**
 * @param _beanParent the beanParent to set
 */
public void setBeanParent(BeanParent _beanParent) {
    this.beanParent = _beanParent;
}

/**
 * @return the beanParents
 */
public BeanParent[] getBeanParents() {
    return this.beanParents;
}

```

```

    }

    /**
     * @param _beanParents the beanParents to set
     */
    public void setBeanParents(BeanParent[] _beanParents) {
        this.beanParents = _beanParents;
    }
}

/**
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;

/**
 * parent bean
 */
public class BeanParent {
    /** field */
    private String parentField1;

    /** field */
    private String parentField2;

    /** field */
    private String[] parentStringArray;

    /** field */
    private int intField;

    /** ean child */
    private BeanChild beanChild;

    /** child */
    private BeanChild beanChild2;

    /** child */
    private BeanChild[] beanArray;

    /** child */
    private BeanChild[] beanArray2;

    /**
     * @return the parentField1
     */
    public String getParentField1() {
        return this.parentField1;
    }

    /**
     * @param _parentField1 the parentField1 to set
     */
    public void setParentField1(String _parentField1) {
        this.parentField1 = _parentField1;
    }

    /**
     * @return the parentField2
     */
    public String getParentField2() {

```

```

    return this.parentField2;
}

/**
 * @param _parentField2 the parentField2 to set
 */
public void setParentField2(String _parentField2) {
    this.parentField2 = _parentField2;
}

/**
 * @return the parentStringArray
 */
public String[] getParentStringArray() {
    return this.parentStringArray;
}

/**
 * @param _parentStringArray the parentStringArray to set
 */
public void setParentStringArray(String[] _parentStringArray) {
    this.parentStringArray = _parentStringArray;
}

/**
 * empty
 */
public BeanParent() {
    //empty
}

/**
 * @param _parentField1
 * @param _parentField2
 * @param _parentStringArray
 * @param _intField
 * @param _beanChild
 * @param _beanChild2
 * @param _beanArray
 * @param _beanArray2
 */
public BeanParent(String _parentField1, String _parentField2, String[] _parentStringArray,
    int _intField, BeanChild _beanChild, BeanChild _beanChild2, BeanChild[] _beanArray,
    BeanChild[] _beanArray2) {
    super();
    this.parentField1 = _parentField1;
    this.parentField2 = _parentField2;
    this.parentStringArray = _parentStringArray;
    this.intField = _intField;
    this.beanChild = _beanChild;
    this.beanChild2 = _beanChild2;
    this.beanArray = _beanArray;
    this.beanArray2 = _beanArray2;
}

/**
 * @return the intField
 */
public int getIntField() {
    return this.intField;
}

```



```
/**
 * @param _intField the intField to set
 */
public void setIntField(int _intField) {
    this.intField = _intField;
}
```

```
/**
 * @return the beanChild
 */
public BeanChild getBeanChild() {
    return this.beanChild;
}
```

```
/**
 * @param _beanChild the beanChild to set
 */
public void setBeanChild(BeanChild _beanChild) {
    this.beanChild = _beanChild;
}
```

```
/**
 * @return the beanChild2
 */
public BeanChild getBeanChild2() {
    return this.beanChild2;
}
```

```
/**
 * @param _beanChild2 the beanChild2 to set
 */
public void setBeanChild2(BeanChild _beanChild2) {
    this.beanChild2 = _beanChild2;
}
```

```
/**
 * @return the beanArray
 */
public BeanChild[] getBeanArray() {
    return this.beanArray;
}
```

```
/**
 * @param _beanArray the beanArray to set
 */
public void setBeanArray(BeanChild[] _beanArray) {
    this.beanArray = _beanArray;
}
```

```

/**
 * @return the beanArray2
 */
public BeanChild[] getBeanArray2() {
    return this.beanArray2;
}

/**
 * @param _beanArray2 the beanArray2 to set
 */
public void setBeanArray2(BeanChild[] _beanArray2) {
    this.beanArray2 = _beanArray2;
}

}

/*
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;

/**
 * child bean
 */
public class BeanChild {
    /** field */
    private String childField1;

    /** field */
    private String childField2;

    /** field */
    private String[] childStringArray;

    /** int array */
    private int[] childIntegerArray;

    /**
     * empty
     */
    public BeanChild() {
        //empty
    }

    /**
     * @param _childField1
     * @param _childField2
     * @param _childStringArray
     * @param _childIntegerArray
     */
    public BeanChild(String _childField1, String _childField2, String[] _childStringArray,
        int[] _childIntegerArray) {
        super();
        this.childField1 = _childField1;
        this.childField2 = _childField2;
        this.childStringArray = _childStringArray;
        this.childIntegerArray = _childIntegerArray;
    }
}

```

```

/**
 * field
 * @return the childField1
 */
public String getChildField1() {
    return this.childField1;
}

/**
 * field
 * @param _childField1 the childField1 to set
 */
public void setChildField1(String _childField1) {
    this.childField1 = _childField1;
}

/**
 * field
 * @return the childField2
 */
public String getChildField2() {
    return this.childField2;
}

/**
 * field
 * @param _childField2 the childField2 to set
 */
public void setChildField2(String _childField2) {
    this.childField2 = _childField2;
}

/**
 * field
 * @return the childStringArray
 */
public String[] getChildStringArray() {
    return this.childStringArray;
}

/**
 * field
 * @param _childStringArray the childStringArray to set
 */
public void setChildStringArray(String[] _childStringArray) {
    this.childStringArray = _childStringArray;
}

/**
 * field
 * @return the childIntegerArray
 */
public int[] getChildIntegerArray() {
    return this.childIntegerArray;
}

/**

```

```

* field
* @param _childIntegerArray the childIntegerArray to set
*/
public void setChildIntegerArray(int[] _childIntegerArray) {
    this.childIntegerArray = _childIntegerArray;
}
}

```

and for this code:

```

BeanChild beanChild = new BeanChild("va<l1", "val2", new String[]{"a"}, new int[]{1, 2});
    BeanParent beanParent = new BeanParent("qwe", "rtyu", new String[]{"uio", "cv"}, 45,
        null, beanChild, null, new BeanChild[]{beanChild});

    BeanGrandparent beanGrandparent = new BeanGrandparent("xv", "",
        new String[]{null}, beanParent, new BeanParent[]{beanParent, beanParent} );

```

It will generate this XHTML:

```

<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>the title</title>
  </head>
  <body>
    <div title="BeanGrandparent">
      <p class="field1">xv</p>
      <p class="field2" />
      <ul class="stringArray">
        <li />
      </ul>
      <div class="beanParent" title="BeanParent">
        <p class="parentField1">qwe</p>
        <p class="parentField2">rtyu</p>
        <ul class="parentStringArray">
          <li>uio</li>
          <li>cv</li>
        </ul>
        <p class="intField">45</p>
        <div class="beanChild" title="BeanChild" />
        <div class="beanChild2" title="BeanChild">
          <p class="childField1">va&lt;l1</p>
          <p class="childField2">val2</p>
          <ul class="childStringArray">
            <li>a</li>
          </ul>
          <ul class="childIntegerArray">
            <li>1</li>
            <li>2</li>
          </ul>
        </div>
        <ul class="beanArray2">
          <li title="BeanChild">
            <p class="childField1">va&lt;l1</p>
            <p class="childField2">val2</p>
            <ul class="childStringArray">
              <li>a</li>
            </ul>
            <ul class="childIntegerArray">
              <li>1</li>
              <li>2</li>
            </ul>
          </li>
        </ul>
      </div>
    </div>
  </body>
</html>

```

```

<ul class="beanParents">
  <li title="BeanParent">
    <p class="parentField1">qwe</p>
    <p class="parentField2">rtyu</p>
    <ul class="parentStringArray">
      <li>uio</li>
      <li>cv</li>
    </ul>
    <p class="intField">45</p>
    <div class="beanChild" title="BeanChild" />
    <div class="beanChild2" title="BeanChild">
      <p class="childField1">va&lt;l1</p>
      <p class="childField2">val2</p>
      <ul class="childStringArray">
        <li>a</li>
      </ul>
      <ul class="childIntegerArray">
        <li>1</li>
        <li>2</li>
      </ul>
    </div>
    <ul class="beanArray2">
      <li title="BeanChild">
        <p class="childField1">va&lt;l1</p>
        <p class="childField2">val2</p>
        <ul class="childStringArray">
          <li>a</li>
        </ul>
        <ul class="childIntegerArray">
          <li>1</li>
          <li>2</li>
        </ul>
      </li>
    </ul>
  </li>
  <li title="BeanParent">
    <p class="parentField1">qwe</p>
    <p class="parentField2">rtyu</p>
    <ul class="parentStringArray">
      <li>uio</li>
      <li>cv</li>
    </ul>
    <p class="intField">45</p>
    <div class="beanChild" title="BeanChild" />
    <div class="beanChild2" title="BeanChild">
      <p class="childField1">va&lt;l1</p>
      <p class="childField2">val2</p>
      <ul class="childStringArray">
        <li>a</li>
      </ul>
      <ul class="childIntegerArray">
        <li>1</li>
        <li>2</li>
      </ul>
    </div>
    <ul class="beanArray2">
      <li title="BeanChild">
        <p class="childField1">va&lt;l1</p>
        <p class="childField2">val2</p>
        <ul class="childStringArray">
          <li>a</li>
        </ul>
        <ul class="childIntegerArray">
          <li>1</li>
          <li>2</li>
        </ul>
      </li>
    </ul>
  </li>

```

```

        </li>
    </ul>
</li>
</ul>
</div>
</body>
</html>

```

It generates this XML:

```

<BeanGrandparent>
  <field1>xv</field1>
  <stringArray>
    <null />
  </stringArray>
  <beanParent>
    <parentField1>qwe</parentField1>
    <parentField2>rtyu</parentField2>
    <parentStringArray>
      <string>uio</string>
      <string>cv</string>
    </parentStringArray>
    <intField>45</intField>
    <beanChild2>
      <childField1>v&quot;a&lt;l{1}</childField1>
      <childField2>val2</childField2>
      <childStringArray>
        <string>a</string>
      </childStringArray>
      <childIntegerArray>
        <int>1</int>
        <int>2</int>
      </childIntegerArray>
    </beanChild2>
  </beanParent>
  <beanArray2>
    <BeanChild>
      <childField1>v&quot;a&lt;l{1}</childField1>
      <childField2>val2</childField2>
      <childStringArray>
        <string>a</string>
      </childStringArray>
      <childIntegerArray>
        <int>1</int>
        <int>2</int>
      </childIntegerArray>
    </BeanChild>
  </beanArray2>
</beanParent>
<beanParents>
  <BeanParent>
    <parentField1>qwe</parentField1>
    <parentField2>rtyu</parentField2>
    <parentStringArray>
      <string>uio</string>
      <string>cv</string>
    </parentStringArray>
    <intField>45</intField>
    <beanChild2>
      <childField1>v&quot;a&lt;l{1}</childField1>
      <childField2>val2</childField2>
      <childStringArray>
        <string>a</string>
      </childStringArray>
      <childIntegerArray>
        <int>1</int>
        <int>2</int>
      </childIntegerArray>
    </beanChild2>
  </BeanParent>
</beanParents>

```

```

        </childIntegerArray>
    </beanChild2>
<beanArray2>
  <BeanChild>
    <childField1>v&quot;a&lt;l{1}</childField1>
    <childField2>val2</childField2>
    <childStringArray>
      <string>a</string>
    </childStringArray>
    <childIntegerArray>
      <int>1</int>
      <int>2</int>
    </childIntegerArray>
  </BeanChild>
</beanArray2>
</BeanParent>
<BeanParent>
  <parentField1>qwe</parentField1>
  <parentField2>rtyu</parentField2>
  <parentStringArray>
    <string>uio</string>
    <string>cv</string>
  </parentStringArray>
  <intField>45</intField>
  <beanChild2>
    <childField1>v&quot;a&lt;l{1}</childField1>
    <childField2>val2</childField2>
    <childStringArray>
      <string>a</string>
    </childStringArray>
    <childIntegerArray>
      <int>1</int>
      <int>2</int>
    </childIntegerArray>
  </beanChild2>
<beanArray2>
  <BeanChild>
    <childField1>v&quot;a&lt;l{1}</childField1>
    <childField2>val2</childField2>
    <childStringArray>
      <string>a</string>
    </childStringArray>
    <childIntegerArray>
      <int>1</int>
      <int>2</int>
    </childIntegerArray>
  </BeanChild>
</beanArray2>
</BeanParent>
</beanParents>
</BeanGrandparent>

```

And it generates this JSON:

```

{
  "BeanGrandparent":{
    "field1":"xv",
    "stringArray":{
      "null":""
    },
    "beanParent":{
      "parentField1":"qwe",
      "parentField2":"rtyu",
      "parentStringArray":{
        "string":[
          "uio",

```

```

        "cv"
    ]
},
"intField":45,
"beanChild2":{
    "childField1":"v\"a<l{1}",
    "childField2":"val2",
    "childStringArray":{
        "string":"a"
    },
    "childIntegerArray":{
        "int":[
            1,
            2
        ]
    }
},
"beanArray2":{
    "BeanChild":{
        "childField1":"v\"a<l{1}",
        "childField2":"val2",
        "childStringArray":{
            "string":"a"
        },
        "childIntegerArray":{
            "int":[
                1,
                2
            ]
        }
    }
},
"beanParents":{
    "BeanParent":[
        {
            "parentField1":"qwe",
            "parentField2":"rtyu",
            "parentStringArray":{
                "string":[
                    "uio",
                    "cv"
                ]
            },
            "intField":45,
            "beanChild2":{
                "childField1":"v\"a<l{1}",
                "childField2":"val2",
                "childStringArray":{
                    "string":"a"
                },
                "childIntegerArray":{
                    "int":[
                        1,
                        2
                    ]
                }
            },
            "beanArray2":{
                "BeanChild":{
                    "childField1":"v\"a<l{1}",
                    "childField2":"val2",
                    "childStringArray":{
                        "string":"a"
                    },
                    "childIntegerArray":{

```



```

        "int":[
            1,
            2
        ]
    }
}
},
{
    "parentField1":"qwe",
    "parentField2":"rtyu",
    "parentStringArray":{
        "string":[
            "uio",
            "cv"
        ]
    },
    "intField":45,
    "beanChild2":{
        "childField1": "\a<l{1}",
        "childField2": "val2",
        "childStringArray":{
            "string": "a"
        },
        "childIntegerArray":{
            "int":[
                1,
                2
            ]
        }
    },
    "beanArray2":{
        "BeanChild":{
            "childField1": "\a<l{1}",
            "childField2": "val2",
            "childStringArray":{
                "string": "a"
            },
            "childIntegerArray":{
                "int":[
                    1,
                    2
                ]
            }
        }
    }
}
]
}
}
}

```

Warnings example

Here is INVALID MARKUP, (extra attribute, and extra element) but it is successfully parsed and here is the warning that the web service will produce in the response back to the client (though the request will still succeed if everything else is ok)

```
<div title=" BeanChild " id=" something">
  <span>something</span>
  <p class="childField1">va&lt;l1</p>
  <p class="childField2">val2</p>
  <ul class="childStringArray">
    <li>a</li>
  </ul>
  <ul class="childIntegerArray">
```

```
<li>1</li>
<li>2</li>
</ul>
</div>
```

Warnings:

Element 'div' is not expecting attribute: 'id'
Element 'div' is not expecting child element: 'span'

Group Save

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Group save will insert or update a group's uuid, extension, display name, or description (with restrictions). Note: the group displayName and extension are not used in a groupSave. That information is used from the group name, and displayExtension.

Features

- Can pass SaveMode which is INSERT, UPDATE, or INSERT_OR_UPDATE (default)
- If the stem doesnt exist, the call will fail
- Lookup group to edit by group lookup (by name or uuid)
- Returns group, can be detailed or not
- Can actAs another user

Group save Lite service

- Accepts one group to save
- Documentation: [SOAP](#) (click on groupSaveLite), [REST](#) (click on groupSaveLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Group save service

- Accepts multiple groups to save
- This will persist (insert/update/delete) types, attributes, composites from detail
- Documentation: [SOAP](#) (click on groupSave), [REST](#) (click on groupSave)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/groups
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

FAQ

- **How can I make a group which has a manual membership list and requires users to be faculty student or staff?**

First off, you need permission to view the facultyStudentStaff group, if it is not public. Note, the composite arguments shouldnt be necessary, but until it is fixed, use them and it will work. This makes a group, a system of record group (where the manual entries go), and the overall group is a composite intersection of the manual group and the facultyStudentStaff group. What does that look like in a soap request? (note, fields which arent used need to be there, due to axis bug. Note you need to enable "requireGroups" in your grouper.properties

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
  <soapenv:Body>
    <ns1:groupSave xmlns:ns1="http://soap.ws.grouper.middleware.internet2.edu/xsd">
      <ns1:clientVersion>v1_4_002</ns1:clientVersion>
      <ns1:wsGroupToSaves>
        <ns1:wsGroup>
          <ns1:description>
            test group with requiring active facultyStudentStaff
          </ns1:description>
          <ns1:detail>
            <ns1:attributeNames>requireAlsoInGroups</ns1:attributeNames>
            <ns1:attributeValues>penn:community:facultyStudentStaff</ns1:attributeValues>
          </ns1:detail>
        </ns1:wsGroup>
      </ns1:wsGroupToSaves>
    </ns1:groupSave>
  </soapenv:Body>
</soapenv:Envelope>
```

```

<ns1:compositeType>intersection</ns1:compositeType>
<ns1:hasComposite>T</ns1:hasComposite>
<ns1:leftGroup>
  <ns1:description></ns1:description>
  <ns1:displayExtension></ns1:displayExtension>
  <ns1:displayName></ns1:displayName>
  <ns1:extension></ns1:extension>
  <ns1:name>penn:community:facultyStudentStaff</ns1:name>
  <ns1:uuid></ns1:uuid>
</ns1:leftGroup>
<ns1:rightGroup>
  <ns1:description></ns1:description>
  <ns1:displayExtension></ns1:displayExtension>
  <ns1:displayName></ns1:displayName>
  <ns1:extension></ns1:extension>
  <ns1:name>test:isc:astt:chris:myGroup_systemOfRecord</ns1:name>
  <ns1:uuid></ns1:uuid>
</ns1:rightGroup>
<ns1:typeNameNames>requireInGroups</ns1:typeNameNames>
</ns1:detail>
<ns1:displayExtension>My test group</ns1:displayExtension>
<ns1:extension>myGroup</ns1:extension>
<ns1:name>test:isc:astt:chris:myGroup</ns1:name>
</ns1:wsGroup>
<ns1:wsGroupLookup>
  <ns1:groupName>test:isc:astt:chris:myGroup</ns1:groupName>
</ns1:wsGroupLookup>
</ns1:wsGroupToSaves>
<ns1:actAsSubjectLookup>
  <ns1:subjectId></ns1:subjectId>
</ns1:actAsSubjectLookup>
<ns1:txType></ns1:txType>
<ns1:includeGroupDetail>T</ns1:includeGroupDetail>
</ns1:groupSave>
</soapenv:Body>
</soapenv:Envelope>

```

Has Member

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Has member will see if a group contains a subject as a member

Features

- Can base member list based on memberfilter (e.g. All, Immediate, Effective)
- Can pass in Field name to query based on Field (e.g. admins, optouts, optins, etc from Field table in DB)
- Lookup subjects by subject lookup (by id, source, identifier, etc)
- Lookup groups by group lookup (by name or uuid)
- Returns group / subject information, can be detailed or not
- Can actAs another user

Has member Lite service

- Accepts one group and one subject to see if member
- Documentation: [SOAP](#) (click on hasMemberLite), [REST](#) (click on hasMemberLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): GET /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members/10021368
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Has member service

- Accepts a group and multiple subjects to see if members
- Documentation: [SOAP](#) (click on hasMember), [REST](#) (click on hasMember)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/groups/aStem%3AaGroup/members
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Member change subject

This page last changed on Oct 13, 2009 by mchzyer@idp.protectnetwork.org.

Description

"[Member change subject](#)" will change the subject that a member refers to. You would want to do this when a person or entity changes their id, or if they were loaded wrong in the system. If the new subject does not have a member associated with it, this is a simple case, where the subject data is put in the member object. If the new subject does have a member object, then all data in all tables that referred to the old member object, will now refer to the new member object. The old member is deleted from the member table by default, though this is an option. Generally you will want it removed, unless there is a foreign key problem where you need to do as much work as possible. In GSH you can get a dry-run report of what will be done.

The operation is potentially time consuming only when two formerly separate Subjects are being merged into one, and that the time required is to replace the memberships (and audit fields e.g. modifiedBy) of the formerly separate Subject that is being retired with new ones associated with the other Subject.

Features

- Will not fail if the new subject is the same as the old subject
- Lookup subjects/members by subject lookup (by id, source, identifier, etc)
- Returns subject information of the new subject
- Can actAs another user

Member change subject Lite service

- Accepts one subject (new) and one member (old) to change the subject information
- Documentation: [SOAP](#) (click on memberChangeSubject), [REST](#) (click on memberChangeSubject)
- For REST, a request body is required (probably via POST)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_4_000/members/10021368
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Member change subject service

- Accepts a list of subjects to change, and subjects to change to (and if the old should be deleted), to change the subjects of members
- Can operate in one transaction, or can let each change occur in its own separate unit
- Documentation: [SOAP](#) (click on memberChangeSubject), [REST](#) (click on memberChangeSubject)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_4_000/members
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each change
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Stem Delete

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Stem delete will insert or update a stem's uuid, extension, display name, or description (with restrictions)

Features

- If stem does not exist, the call will not fail (special result code)
- Lookup stem to delete by stem lookup (by name or uuid)
- Returns stem, can be detailed or not
- Can actAs another user

Stem delete Lite service

- Accepts one stem to delete
- Documentation: [SOAP](#) (click on stemDeleteLite), [REST](#) (click on stemDeleteLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): DELETE /grouper-ws/servicesRest/v1_3_000/stems/aStem%3AaStem2
 - Note: if passing data in request body e.g. actAs, use a POST
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Stem delete service

- Accepts multiple stems to delete
- Documentation: [SOAP](#) (click on stemDelete), [REST](#) (click on stemDelete)
- REST request (colon is escaped to %3A): POST /grouper-ws/servicesRest/v1_3_000/stems
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Stem Save

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

Description

Stem save will insert or update a stem's uuid, extension, display name, or description (with restrictions)

Features

- Can pass SaveMode which is INSERT, UPDATE, or INSERT_OR_UPDATE (default)
- If the parent stem doesn't exist, the call will fail
- Lookup stem to edit by stem lookup (by name or uuid)
- Returns stem
- Can actAs another user

Stem save Lite service

- Accepts one stem to save
- Documentation: [SOAP](#) (click on stemSaveLite), [REST](#) (click on stemSaveLite)
- For REST, the request can put data in query string (in URL or request body)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/stems/aStem%3AaStem2
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes](#)
- [Samples](#) (all files with "Lite" in them, click on "download" to see file)

Stem save service

- Accepts multiple stems to save
- Documentation: [SOAP](#) (click on stemSave), [REST](#) (click on stemSave)
- REST request (colon is escaped to %3A): PUT /grouper-ws/servicesRest/v1_3_000/stems
- (see documentation above for details): [Request object](#), [response object](#)
- [Response codes overall](#), [response codes for each assignment](#)
- Returns an overall status, and a status for each assignment
- [Samples](#) (all files without "Lite" in them, click on "download" to see files)

Web Services FAQ

This page last changed on Apr 03, 2008 by [mchzyer](#).

- Can I run grouper web services against any version of grouper?

No, you should use the version of grouper bundled in the web services. This makes it a little complicated if you have a UI / GSH / etc running against your grouper DB. You will need to keep them in sync...

- Why are there three flavors of web services (SOAP, XML-HTTP, REST-Lite)?

SOAP is supported since many schools required it. REST-Lite is supported because many schools required it. XML-HTTP is supported because Axis2 gives it for free when building a SOAP web service. This way the same SOAP interface can be accessed by clients who only want to talk HTTP and XML and not SOAP.

- Why is the rampart .aar file named GrouperServiceWssec.aar, but the URL is still /services/GrouperService?

The URL for rampart or not is the same. Inside the services.xml in the .aar files, it configures the app name. grouper-ws will not work if you change this (unless to do other build activities also)

- Why is there a "simple" operation for every non simple operation in SOAP and XML-HTTP?

The non-simple operations are batchable if the client wants to do one operation multiple times with one request. The simple operations are for if the client only needs to do one thing (e.g. assign a member to a group and not assign multiple members to a group). Also, there is a valuable side effect that for the XML-HTTP, if the operation only has scalar input params (and not complex types or arrays), that the input does not need XML, it can be in the query string for GET or in the http form param pairs in the body for POST.

It is confusing that there are so many different ways to call grouper via web service. The documentation will be improved to make it easier to find the best strategy.

- Why element named "return" (by axis)

This is an unfortunate "feature" by axis, the default element is named "return" which is a keyword in many programming languages, so if the language automatically converts the xml to an object graph, then it will be broken. Chris will followup with Axis to see if there is a fix for this

- Why returning error codes and messages and not just use SOAP faults?

For two of three of the flavors of web services SOAP faults are not an option since they are not SOAP. Also, for SOAP batched, the status of each line item needs to be returned for the client to process, and a SOAP fault would preclude that. Also, the fewer SOAP specific features that are used, the more widespread the compatibility will be. All three flavors of web service use the same underlying logic, so the more consistent the better.

- Can we add a service for subject search?

Yes, this will be added

WS - Proof of Concept 2007-10-2007

This page last changed on Jan 02, 2008 by jbibbee@internet2.edu.

[GROUPER:](#) [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

Summary

12/10/2007 A proof of concept of Axis2 web services was completed. It is a service that takes a query for a subject search and returns the subjects. It uses the same API that the UI uses to search for subjects. The Axis2 web service supports SOAP and REST. The proof of concept is not an actual service that we will use, the inputs and outputs and design of which services we need will be more thoroughly thought out, the intent is to get Axis up and running. Also, I cleansed all the data, so if you notice that it is not consistent, it is due to data cleansing.

The Service

The service is a method which takes a query for subjects, and returns an array of subjects (at some point I had lists working with Axis, then it stopped working, so I am left with an array)

```
package edu.internet2.middleware.grouper.webservices;

import java.util.Set;

import edu.internet2.middleware.grouper.SubjectFinder;
import edu.internet2.middleware.subject.provider.JDBCSubject;

public class GrouperService {

    /**
     * web service wrapper for find all subjects based on query
     * @param query
     * @return
     */
    @SuppressWarnings("unchecked")
    public WsSubject[] findAll(String query) {
        Set<JDBCSubject> subjectSet = SubjectFinder.findAll(query);
        if (subjectSet == null || subjectSet.size() == 0) {
            return null;
        }
        //convert the set to a list
        WsSubject[] results = new WsSubject[subjectSet.size()];
        int i=0;
        for (JDBCSubject jdbcSubject : subjectSet) {
            WsSubject wsSubject = new WsSubject(jdbcSubject);
            results[i++] = wsSubject;
        }
        return results;
    }
}
```

Note that the return type is WsSubject (detailed below). I am assuming that the beans used for the web service will be new, and used solely for the web services. The WSDL is based (automatically) on the GrouperService class and the beans it uses. Here is the bean:

```
package edu.internet2.middleware.grouper.webservices;

import edu.internet2.middleware.subject.provider.JDBCSubject;

/**
 * subject bean for web services
 * @author mchyzer
 */
```

```

*/
public class WsSubject {

    public WsSubject(){}

    public WsSubject(JDBCSubject jdbcSubject) {
        this.id = jdbcSubject.getId();
        this.description = jdbcSubject.getDescription();
        this.name = jdbcSubject.getName();
    }

    private String id;
    private String name;
    private String description;
    /**
     * @return the id
     */
    public String getId() {
        return id;
    }
    /**
     * @param id the id to set
     */
    public void setId(String id) {
        this.id = id;
    }
    /**
     * @return the name
     */
    public String getName() {
        return name;
    }
    /**
     * @param name the name to set
     */
    public void setName(String name) {
        this.name = name;
    }
    /**
     * @return the description
     */
    public String getDescription() {
        return description;
    }
    /**
     * @param description the description to set
     */
    public void setDescription(String description) {
        this.description = description;
    }
}

```

REST Results

When I go to this url: <http://localhost:8090/grouper/services/GrouperService/findAll?query=100abcd>, I see this (well, after adding some whitespace):

```

<ns:findAllResponse>
  <ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
    <ns:description>JUSTIN SMITH</ns:description>
    <ns:id>100abcda</ns:id>
    <ns:name>JUSTIN SMITH</ns:name>
  </ns:return>
  <ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
    <ns:description>MICHAEL SMITH</ns:description>

```

```

<ns:id>100abddb</ns:id>
<ns:name>MICHAEL SMITH</ns:name>
</ns:return>
<ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
  <ns:description>MARLENE SMITH</ns:description>
  <ns:id>100abdc</ns:id>
  <ns:name>MARLENE SMITH</ns:name>
</ns:return>
</ns:findAllResponse>

```

Here is a Java REST client which does not use any Java data structures that the SOAP client uses:

```

package edu.internet2.middleware.grouper.webservicesClient;

import java.io.Reader;
import java.io.StringReader;
import java.util.List;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.GetMethod;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.input.SAXBuilder;

public class RunGrouperServiceNonAxis {

    /**
     * @param args
     */
    @SuppressWarnings("unchecked")
    public static void main(String[] args) throws Exception {
        HttpClient httpClient = new HttpClient();
        GetMethod getMethod = new GetMethod(
            "http://localhost:8090/grouper/services/GrouperService/findAll?query=1002136");
        httpClient.executeMethod(getMethod);
        int statusCode = getMethod.getStatusCode();
        // see if request worked or not
        if (statusCode != 200) {
            throw new RuntimeException("Bad response from web service: "
                + statusCode);
        }
        String response = getMethod.getResponseBodyAsString();
        Reader xmlReader = new StringReader(response);
        try {
            // process xml
            Document document = new SAXBuilder().build(xmlReader);
            Element findAllResponse = document.getRootElement();
            assertTrue("findAllResponse".equals(findAllResponse.getName()),
                "root not findAllResponse: " + findAllResponse.getName());

            // process each child
            List<Element> returns = findAllResponse.getChildren();
            for (Element theReturn : returns) {

                // make sure they have the right element name and type
                assertTrue("return".equals(theReturn.getName()),
                    "element not 'return': " + theReturn.getName());
                assertTrue(
                    "edu.internet2.middleware.grouper.webservices.WsSubject"
                        .equals(theReturn.getAttributeValue("type")),
                    "type is: " + theReturn.getAttributeValue("type"));

                String description = theReturn.getChild("description",
                    findAllResponse.getNamespace()).getValue();
                String id = theReturn.getChild("id",
                    findAllResponse.getNamespace()).getValue();
            }
        }
    }
}

```

```

        String name = theReturn.getChild("name",
            findAllResponse.getNamespace()).getValue();

        System.out.println(id + " - " + name + " - " + description);
    }
} finally {
    try {
        xmlReader.close();
    } catch (Exception e) {
    }
}
}

/**
 * assert like java 1.4 assert
 *
 * @param isTrue
 * @param reason
 */
private static void assertTrue(boolean isTrue, String reason) {
    if (!isTrue) {
        throw new RuntimeException(reason);
    }
}
}
}

```

SOAP Results

First need to make a SOAP client. Before we can do that we need the WSDL from here (Axis automatically generated this WSDL): <http://localhost:8090/grouper/services/GrouperService?wsdl>

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:axis2="http://
webservices.grouper.middleware.internet2.edu/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/
mime/" xmlns:ns0="http://webservices.grouper.middleware.internet2.edu/xsd" xmlns:soap12="http://
schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd" xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://webservices.grouper.middleware.internet2.edu/">
  <wsdl:documentation>GrouperService</wsdl:documentation>
  <wsdl:types>
    <xs:schema xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd"
      attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://
webservices.grouper.middleware.internet2.edu/xsd">
      <xs:element name="findAll">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="query" nillable="true" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="findAllResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="return" nillable="true"
type="ns:WsSubject"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:complexType name="WsSubject">
        <xs:sequence>
          <xs:element minOccurs="0" name="description" nillable="true" type="xs:string"/>
          <xs:element minOccurs="0" name="id" nillable="true" type="xs:string"/>

```

```

        <xs:element minOccurs="0" name="name" nillable="true" type="xs:string"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>
</wsdl:types>
<wsdl:message name="findAllRequest">
    <wsdl:part name="parameters" element="ns0:findAll"/>
</wsdl:message>
<wsdl:message name="findAllResponse">
    <wsdl:part name="parameters" element="ns0:findAllResponse"/>
</wsdl:message>
<wsdl:portType name="GrouperServicePortType">
    <wsdl:operation name="findAll">
        <wsdl:input message="axis2:findAllRequest" wsaw:Action="urn:findAll"/>
        <wsdl:output message="axis2:findAllResponse" wsaw:Action="urn:findAllResponse"/>
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="GrouperServiceSOAP11Binding" type="axis2:GrouperServicePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <wsdl:operation name="findAll">
        <soap:operation soapAction="urn:findAll" style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GrouperServiceSOAP12Binding" type="axis2:GrouperServicePortType">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <wsdl:operation name="findAll">
        <soap12:operation soapAction="urn:findAll" style="document"/>
        <wsdl:input>
            <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GrouperServiceHttpBinding" type="axis2:GrouperServicePortType">
    <http:binding verb="POST"/>
    <wsdl:operation name="findAll">
        <http:operation location="GrouperService/findAll"/>
        <wsdl:input>
            <mime:content type="text/xml" part="findAll"/>
        </wsdl:input>
        <wsdl:output>
            <mime:content type="text/xml" part="findAll"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="GrouperService">
    <wsdl:port name="GrouperServiceSOAP11port_http" binding="axis2:GrouperServiceSOAP11Binding">
        <soap:address location="http://localhost:8090/grouper/services/GrouperService"/>
    </wsdl:port>
    <wsdl:port name="GrouperServiceSOAP12port_http" binding="axis2:GrouperServiceSOAP12Binding">
        <soap12:address location="http://localhost:8090/grouper/services/GrouperService"/>
    </wsdl:port>
    <wsdl:port name="GrouperServiceHttpport" binding="axis2:GrouperServiceHttpBinding">
        <http:address location="http://localhost:8090/grouper/services/GrouperService"/>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Then we can use Axis to make a Java SOAP client:

```
C:\mchzyer\jsc\dev\grouper\axisJar>wsdl2java -p edu.internet2.middleware.grouper.webservicesClient -t -uri
GrouperService.wsdl
```

Then we just make a call with very simple Java code (the underlying classes were generated by Axis with the above command):

```
/**
 *
 */
package edu.internet2.middleware.grouper.webservicesClient;

import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.FindAll;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.FindAllResponse;
import edu.internet2.middleware.grouper.webservicesClient.GrouperServiceStub.WsSubject;

/**
 * @author mchzyer
 */
public class RunGrouperService {

    /**
     * @param args
     */
    public static void main(String[] args) throws Exception {
        GrouperServiceStub stub = new GrouperServiceStub(
            "http://localhost:8091/grouper/services/GrouperService");

        FindAll findAll = new FindAll();
        findAll.setQuery("1002136");
        FindAllResponse findAllResponse = stub.findAll(findAll);
        WsSubject[] subjects = findAllResponse.get_return();
        if (subjects == null) {
            System.out.println("No results");
        } else {
            for (WsSubject wsSubject : subjects) {
                System.out.println(wsSubject.getId() + " - "
                    + wsSubject.getName() + " - "
                    + wsSubject.getDescription());
            }
        }
    }
}
```

Here are the results:

```
100abcda - MICHAEL SMITH - MICHAEL SMITH
100abcdb - JUSTIN SMITH - JUSTIN SMITH
100abcdc - MARLENE D SMITH - MARLENE D SMITH
```

Here is the SOAP message that was transmitted:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
<soapenv:Body>
  <ns1:findAll xmlns:ns1="http://webservices.grouper.middleware.internet2.edu/xsd">
    <ns1:query>100abcd</ns1:query>
  </ns1:findAll>
</soapenv:Body>
</soapenv:Envelope>
```

Here is the SOAP response:

```
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
<soapenv:Body>
<ns:findAllResponse xmlns:ns="http://webservices.grouper.middleware.internet2.edu/xsd">
<ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
<ns:description>MICHAEL SMITH</ns:description>
<ns:id>100abcda</ns:id>
<ns:name>MICHAEL SMITH</ns:name>
</ns:return>
<ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
<ns:description>JUSTIN SMITH</ns:description>
<ns:id>100abddb</ns:id>
<ns:name>JUSTIN SMITH</ns:name>
</ns:return>
<ns:return type="edu.internet2.middleware.grouper.webservices.WsSubject">
<ns:description>MARLENE D SMITH</ns:description>
<ns:id>100abdc</ns:id>
<ns:name>MARLENE D SMITH</ns:name>
</ns:return>
</ns:findAllResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Details

1. I downloaded the latest stable Axis2: v1.3
2. I made a directory with the contents, and set an env variable: AXIS2_HOME
3. I copied all the jars (~40) to grouper/lib, and removed the dupes (I think the only one was xml-apis.jar)
4. Copy the axis quick start directories: "services", "modules", "conf" to grouper-ui/webapp/WEB-INF
5. Create an axis archive (.aar) file. Based this on the Axis quickstart one:



```
C:\temp\grouperaar>find
.
.\edu
.\edu\internet2
.\edu\internet2\middleware
.\edu\internet2\middleware\grouper
.\edu\internet2\middleware\grouper\webservices
.\edu\internet2\middleware\grouper\webservices\GrouperService.class
.\edu\internet2\middleware\grouper\webservices\WsSubject.class
.\GrouperService.aar
.\META-INF
.\META-INF\MANIFEST.MF
.\META-INF\services.xml
```

Each time the service classes are changed (these are also in the grouper/src), the classfiles need to be put in the GrouperService.aar file. The services.xml file has the following contents:

```
<service name="GrouperService" scope="application" targetNamespace="http://
webservices.grouper.middleware.internet2.edu/">
<description>
Grouper Service
</description>
<messageReceivers>
<messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
<messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
</messageReceivers>
<schema schemaNamespace="http://webservices.grouper.middleware.internet2.edu/xsd"/>
<parameter name="ServiceClass">edu.internet2.middleware.grouper.webservices.GrouperService</
parameter>
</service>
```

Zip that up in GrouperSystem.aar, and put that in the WEB-INF/services dir

1. Do a "dist" ui build, start the app server

 Questions or comments?  [Contact us](#).

GROUPER: [FAQ](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#)

.NET Clients for Grouper WS

This page last changed on Sep 05, 2008 by sanjay.vivek@ncl.ac.uk.

Development of the .NET client was done on Visual Studio 2005. A working knowledge of developing projects on Visual Studio is required before proceeding with this guide. The .NET client for the Grouper WS is merely proof of concept and as such, the UI is very simplistic. All relevant files can be found in the Attachment section.

Creating a client to consume a Basic Auth enabled Grouper WS

Once you have a client application (project) in place, you will need to add a proxy class to the project that allows the client to access the Grouper WS. The following example describes how to create a Web service client and generate a proxy class that allows the client to access the Grouper WS.

You will begin by creating a project and adding a Web reference to it. When you add the Web reference, Visual C# 2005 will generate the appropriate proxy class. You will then create an instance of the proxy class and use it to call the Web service's methods. First, create a Windows application in Visual C# 2005, then perform the following steps:

Step 1: Opening the Add Web Reference Dialog

Right click the project name in the **Solution Explorer** and select **Add Web Reference**

Step 2: Locating Web Services on Your Computer

In the **Add Web Reference** dialog that appears, enter the URL of the Basic Auth enabled Grouper WS in the URL field.

Step 3: Adding the Web Reference

Add the Web reference by clicking the **Add Reference** button. A Web reference is similar to a package name in Java. I named my Web Reference as **BasicAuthGrouper**.

Step 4: Viewing the Web Reference in the Solution Explorer

The **Solution Explorer** should now contain a **Web References** folder with a node named **BasicAuthGrouper**. This node should contain **GrouperService.discomap** and **GrouperService.wsdl** files.

When we reference class GrouperService in the client application, we will do so through the BasicAuthGrouper namespace.

We now have a client application with a proxy class that calls the Grouper WS methods. Our next step is write C# code to call the Grouper WS methods via the proxy class. Firstly, you will create a Web Form and then invoke the Grouper WS by performing the following steps:

Step 5: Create a Web Form

Right click the Project name and select **Add New Item**. The select **Web Form** and untick the **Place code in separate file** box. You should now have a new Web form in the .aspx format.

Step 6: Write the C# code to call Grouper WS

Copy the code below onto the Web Form:

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Net" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%
    BasicAuthGrouper.WsGroup[] wsGroupArray = null;

    try
```

```

{

    /***/

    /*** Grouper ***/
    * "Grouper" policy requires the wse3 extensions, to create the policy
    * you'll need the wse3 plugin for visual studio, then right click the
    * project and select "wse3 settings" then use a user/pass based authentication
    * method, specify the password and don't use the ws security extensions.
    */

    BasicAuthGrouper.GrouperService gws = new BasicAuthGrouper.GrouperService();

    //setting up Basic Auth stuff
    NetworkCredential netCredential = new NetworkCredential("Username", "Password");
    Uri uri = new Uri(gws.Url);
    ICredentials credentials = netCredential.GetCredential(uri, "Basic");
    gws.Credentials = credentials;
    gws.PreAuthenticate = true;

    BasicAuthGrouper.WsGetGroupsLiteResult wsgglr = gws.getGroupsLite("v1_3_000", "ncf17@ncl.ac.uk",
    "", "", "All", "", "", "", "", "", "", "", "", "");
    wsGroupArray = wsgglr.wsGroups;

    /***/
}
catch(Exception e)
{
%>
    <%= e.ToString() %>
<%
}
%>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Untitled Page</title>
</head>
<body>
    <div>

Grouper says:<br />
<%
    try
    {
        for(int i=0;i<wsGroupArray.Length;i++)
        {
%>
            <br />
            /***WsGroup <%= i+1 %>*****/<br />
            Description: <%= (wsGroupArray[i]).description %><br />
            Detail: <%= (wsGroupArray[i]).detail %><br />
            Display Extension: <%= (wsGroupArray[i]).displayExtension %><br />
            Display Name: <%= (wsGroupArray[i]).displayName %><br />
            Extension: <%= (wsGroupArray[i]).extension %><br />
            Name: <%= (wsGroupArray[i]).name %><br />
            Uuid: <%= (wsGroupArray[i]).uuid %><br />
<%
        }
    }
    catch(Exception e)
    {
%>
        Error: <%= e.ToString() %>
<%
    }
%>

```

```
</div>
</body>
</html>
```

It should be noted that the BasicAuthGrouper from the line of code below is the name given to the Web Reference in Step 4. Packages in Java are named in a similar fashion.

```
BasicAuthGrouper.GrouperService gws = new BasicAuthGrouper.GrouperService();
```

Step 7: Call the Grouper WS with the C# client

Go to the **Debug** menu on top and click on **Start without Debugging**. This means that Visual Studio executes its built-in Web server and you don't have to install the IIS server. A Web browser should open with all the relevant result set. You might have to run this twice because the Basic Auth headers might not go through in the first SOAP request message. This is a peculiarity of .NET.

A pre-authentication mode has to be set the authorization header on every request. The recommended way to get preauthentication is to set the credentials on the request, set PreAuthenticate=true (in Step 6), and send the request. Once the first authentication handshake happens, and credentials are accepted by the server, preauthentication will happen behind the scenes for every request to the same Uri-Prefix.

Creating a client to consume a Rampart enabled Grouper WS

Web Services Enhancements 3.0 (WSE3) for Microsoft® .NET implements WS-Security standards on a .NET framework and has to be installed before it can interoperate with a Rampart enabled service. Rampart is the open source implementation of WS-Security standards and a .NET client should be able to invoke a Rampart enabled service as they both support WS-Security standards.

WSE3 can be **downloaded** from <http://www.microsoft.com/downloads/details.aspx?familyid=018A09FD-3A74-43C5-8EC1-8D789091255D&displaylang=en>

Once WSE3 has been downloaded and installed, restart Visual Studio 2005. Right click on any project and you should now see a new tab right at the bottom stating **WSE Settings 3.0....** The appearance of this tab means that WSE3 has been correctly installed.

Make sure that you install WSE3 and not WSE2 because upgrading from WSE2 to WSE3 is a real hassle and causes lots of interoperability problems.

Once you have a client application (project) in place, you will need to add a proxy class to the project that allows the client to access the Grouper WS. The following example describes how to create a Web service client and generate a proxy class that allows the client to access the Grouper WS.

You will begin by creating a project and adding a Web reference to it. When you add the Web reference, Visual C# 2005 will generate the appropriate proxy class. There's an additional step when it comes to invoking a Rampart enabled Grouper WS whereby you need to define a policy file. You will then create an instance of the proxy class and use it to call the Web service's methods. First, create a Windows application in Visual C# 2005, then perform the following steps:

Step 1: Opening the Add Web Reference Dialog

Right click the project name in the **Solution Explorer** and select **Add Web Reference**

Step 2: Locating Web Services on Your Computer

In the **Add Web Reference** dialog that appears, enter the URL of the Rampart enabled Grouper WS in the URL field.

Step 3: Adding the Web Reference

Add the Web reference by clicking the **Add Reference** button. A Web reference is similar to a package name in Java. I named my Web Reference as **RampartGrouper**.

Step 4: Viewing the Web Reference in the Solution Explorer

The **Solution Explorer** should now contain a **Web References** folder with a node named **RampartGrouper**. This node should contain **GrouperService.discomap** and **GrouperService.wsdl** files.

When we reference class GrouperService in the client application, we will do so through the RampartGrouper namespace.

We now have a client application with a proxy class that calls the Grouper WS methods.

Step 5: Set up a policy file

WS-Security Policy specification defines a standard way to define how to secure messages exchanged between Web services and clients. A Policy file can be used to describe the security constraints of a service and is used by both WSE3 and Rampart. The Policy file for the Rampart enabled Grouper WS requires UsernameToken authentication and any client that wishes to invoke the GrouperWS has to be UsernameToken authentication headers in the SOAP request message. As such, a Policy file has to be set up on the client side that contains the right Username/Password combo.

Right click on the Project and select **Add New Item**. Then select **Web Configuration File** and name it **WSE3Policy** and untick the **Place code in separate file** box.. You should now have a **WSE3Policy.config** file in your project domain. Open the WSE3Policy.config file and copy the policy below into WSE3Policy.config:

```
<policies xmlns="http://schemas.microsoft.com/wse/2005/06/policy">
  <extensions>
    <extension name="usernameForCertificateSecurity"
type="Microsoft.Web.Services3.Design.UsernameForCertificateAssertion," +
      " Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
    <extension name="x509" type="Microsoft.Web.Services3.Design.X509TokenProvider,
Microsoft.Web.Services3, " +
      "Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <extension name="requireActionHeader"
type="Microsoft.Web.Services3.Design.RequireActionHeaderAssertion, " +
      "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
    <extension name="usernameOverTransportSecurity"
type="Microsoft.Web.Services3.Design.UsernameOverTransportAssertion, " +
      "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
    <extension name="username" type="Microsoft.Web.Services3.Design.UsernameTokenProvider, " +
      "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
    <extension name="kerberosSecurity" type="Microsoft.Web.Services3.Design.KerberosAssertion, " +
      "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
    <extension name="kerberos" type="Microsoft.Web.Services3.Design.KerberosTokenProvider, " +
      "Microsoft.Web.Services3, Version=3.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" /
    >
  </extensions>
  <policy name="Grouper">
    <usernameOverTransportSecurity>
      <clientToken>
        <username username="Username" password="Password" />
      </clientToken>
    </usernameOverTransportSecurity>
    <requireActionHeader />
  </policy>
</policies>
```

The policy above is named **Grouper** and has a username/password combo. Insert the right username/password combo in the username element.

Now right click on the Project again and select **WSE3 Settings 3.0 at the bottom**. A multi-tabbed popup will now appear. Click on the **General** tab at the top and tick both **Enable this project for Web Service Enhancements** and **Enable Microsoft Web Services Enhancement Soap Protocol Factory**.

Now click on the **Policy** tab and tick **Enable Policy**. Click on **Browse** and choose the **WSE3Policy.config** file you defined earlier. You should now see the **Grouper** policy file in the **Edit Application Policy** column and click **OK**.

Step 6: Create a Web Form

Right click the Project name and select **Add New Item**. Then select **Web Form** and untick the **Place code in separate file** box. You should now have a new Web form in the .aspx format.

Step 7: Write the C# code to call Grouper WS

Copy the code below onto the Web Form:

```
<%@ Page Language="C#" %>
<%@ Import Namespace="Microsoft.Web.Services3" %>
<%@ Import Namespace="Microsoft.Web.Services3.Security.Tokens" %>
<%@ Import Namespace="System.Net" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<%

    RampartGrouper.WsGroup[] wsGroupArray = null;

    try
    {

        /*****

        /***** Grouper *****/
        * "Grouper" policy requires the wse3 extensions, to create the policy
        * you'll need the wse3 plugin for visual studio, then right click the
        * project and select "wse3 settings" then use a user/pass based authentication
        * method, specify the password and don't use the ws security extensions.
        */

        RampartGrouper.GrouperServiceWse gws = new RampartGrouper.GrouperServiceWse();
        gws.SetPolicy("Grouper");
        RampartGrouper.WsGetGroupsLiteResult wsgglr = gws.getGroupsLite("v1_3_000", "nsv12@ncl.ac.uk", "",
        "", "All", "", "", "", "", "", "", "", "", "", "", "");
        wsGroupArray = wsgglr.wsGroups;

        /*****

    }
    catch(Exception e)
    {
%>
        <%= e.ToString() %>
    }
%>
<%
    <html xmlns="http://www.w3.org/1999/xhtml" >
    <head>
        <title>Untitled Page</title>
    </head>
    <body>
        <div>
            RampartGrouper says: <br />
        <%
            try
            {
                for(int i=0;i<wsGroupArray.Length;i++)
                {
%>
                    <br />
                    /*****WsGroup <%= i+1 %>*****/<br />
                    Description: <%= (wsGroupArray[i]).description %><br />
                    Detail: <%= (wsGroupArray[i]).detail %><br />
                }
            }
            catch { }
        }
    }
%>
```

```

        Display Extension: <%= (wsGroupArray[i]).displayExtension %><br />
        Display Name: <%= (wsGroupArray[i]).displayName %><br />
        Extension: <%= (wsGroupArray[i]).extension %><br />
        Name: <%= (wsGroupArray[i]).name %><br />
        Uuid: <%= (wsGroupArray[i]).uuid %><br />
    <%
    }
    }
    catch(Exception e)
    {
    %>
        Error: <%= e.ToString() %>
    <%
    }
    %>
    </div>
</body>
</html>

```

The important item from the above code the Grouper policy is used in the following line of code:

```
gws.SetPolicy("Grouper");
```

Step 7: Call the Grouper WS with the C# client

Go the **Debug menu** on top and click on **Start without Debugging**. This means that Visual Studio executes its built-in Web server and you don't have to install the IIS server. A Web browser should open with all the relevant result set.

PHP Clients for Grouper Web Service

This page last changed on Sep 02, 2008 by sanjay.vivek@ncl.ac.uk.

This page looks at developing PHP clients for both HTTP Basic Auth and Rampart enabled Grouper WS. One of the core requirements for Grouper WS security is to enable clients to authenticate both against Basic Auth and Rampart enabled Grouper WS. We used the WSO2 Web Services Framework for PHP (WSO2 WSF/PHP) to develop the PHP Clients. WSF/PHP is an open source framework that supports a wide range of WS-* specification implementations. It can be downloaded at <http://wso2.org/downloads/wsf/php/>

WSF/PHP with HTTP Basic Authentication

HTTP Basic Authentication is only supported by **WSF/PHP versions 1.3.0** and above so make sure that the latest version of WSF/PHP is installed.

The code below can be used to invoke the getGroupsLite method on a Basic Auth enabled Grouper WS:

```
<?php

// Request payload string and call the getGroupsLite method
$reqPayloadString = <<<XML
<ns1:getGroupsLite xmlns:ns1="http://soap.ws.grouper.middleware.internet2.edu/
xsd"><ns1:param0>v1_3_000</ns1:param0>

                                <ns1:param1>$_GET[user]</ns1:param1>
                                <ns1:param2></ns1:param2>
                                <ns1:param3></ns1:param3>
                                <ns1:param4>All</ns1:param4>
                                <ns1:param5></ns1:param5>
                                <ns1:param6></ns1:param6>
                                <ns1:param7></ns1:param7>
                                <ns1:param8></ns1:param8>
                                <ns1:param9></ns1:param9>
                                <ns1:param10></ns1:param10>

XML;

try {
    // Create message with request payload and options
    $reqMessage = new WSMessage($reqPayloadString,
        array("to" => "http://XXXXX/basicauthgrouper-ws/services/GrouperService",
            "action" => "http://php.rampart.apache.org"));

    // Create client with options
    $client = new WSClient(array("useWSA" => FALSE,
        "httpAuthUsername" => "Username",
        "httpAuthPassword" => "Password",
        "httpAuthType" => "Basic"));

    // Send request and capture response
    $resMessage = $client->request($reqMessage);
    header("Content-Type: text/xml");
    print_r($resMessage->str);

} catch (Exception $e) {

    if ($e instanceof WSFault) {
        printf("Soap Fault: %s\n", $e->Reason);
    } else {
        printf("Message = %s\n", $e->getMessage());
    }
}
```



```
}
}
?>
```

WSF/PHP with Rampart

WSO2/PHP doesn't contain a WSDL2JAVA tool so we can't a stub to make use of the policies defined in the WSDL. As such, the policy file that defines the security requirements of the service has to be on classpath of the client. The **policy.xml** used in this instance is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="UTOverTransport"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic256/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Lax/>
            </wsp:Policy>
          </sp:Layout>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:UsernameToken sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/
securitypolicy/IncludeToken/AlwaysToRecipient" />
        </wsp:Policy>
      </sp:SignedSupportingTokens>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

A client that calls the `getGroupsLite` method on the Rampart enabled Grouper WS with the exact set of parameters as the Java client is given below:

```
<?php

// Request payload string
$reqPayloadString = <<<XML
<ns1:getGroupsLite xmlns:ns1="http://webservicesRampart.grouper.middleware.internet2.edu/
xsd"><ns1:param0>v1_3_000</ns1:param0>

<ns1:param1>$_GET[user]</ns1:param1>
<ns1:param2></ns1:param2>
<ns1:param3></ns1:param3>
<ns1:param4>All</ns1:param4>
<ns1:param5></ns1:param5>
<ns1:param6></ns1:param6>
<ns1:param7></ns1:param7>
<ns1:param8></ns1:param8>
<ns1:param9></ns1:param9>

XML;
```

```

try {
    // Create message with request payload and options
    $reqMessage = new WSMMessage($reqPayloadString,
        array("to" => "http://XXXXX/grouper-ws/services/GrouperService",
            "action" => "http://php.rampart.apache.org"));

    // Set up security options
    $policy_xml = file_get_contents("policy.xml");
    $policy = new WSPolicy($policy_xml);
    $security_token = new WSSecurityToken(array("user" => "Username",
        "password" => "Password",
        "passwordType" => "Digest"));

    // Create client with options
    $client = new WSClient(array("useWSA" => FALSE,
        "policy" => $policy,
        "securityToken" => $security_token));

    // Send request and capture response
    $resMessage = $client->request($reqMessage);
    header("Content-Type: text/xml");
    print_r($resMessage->str);

} catch (Exception $e) {

    if ($e instanceof WSFault) {
        printf("Soap Fault: %s\n", $e->Reason);
    } else {
        printf("Message = %s\n", $e->getMessage());
    }
}
?>

```

The important part of the above code is:

```

$policy_xml = file_get_contents("policy.xml");
$policy = new WSPolicy($policy_xml);
$security_token = new WSSecurityToken(array("user" => "Username",
    "password" => "Password",
    "passwordType" => "Digest"));

```

This indicates the location of policy.xml. It also provides the username/password in the form of a digest.

Better UI design with XML pagination

XML pagination is used to display the list of users/groups in a more consistent and presentable manner. For example, listgroups.php invokes the getGroupsLite method, which in turn returns an XML array of groups. The XML array of groups is then presented in a more presentable manner using XML pagination techniques. A code snippet is given below:

```

//call getGroupsLite method and return an array of groups
$xml = simplexml_load_file("https://community.ncl.ac.uk/rampart/live/getgroups.php?user=$user");

//Total number of Groups
$totalGroups = $xml->xpath("//ns:name");

```

Both the listgroups.php and the pagination.class.php are provided in the Attachments section.

Getting started with hooks

This page last changed on Feb 24, 2009 by [mchzyer](#).

Instructions to get started with grouper and hooks.

- Install postgres, add a grouper user with a pass.
- Download the latest grouper 1.4 branch
 - `cvs -d:pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi login`
 - `cvs -d:pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi export -r GROUPER_1_4_BRANCH grouper`
 - `cvs -d:pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi export -r GROUPER_UI_1_4_BRANCH grouper-ui`
- In grouper dir, run: `ant dist`
- Change these settings in `conf/grouper.hibernate.properties`
 - `hibernate.dialect = org.hibernate.dialect.PostgreSQLDialect`
 - `hibernate.connection.driver_class = org.postgresql.Driver`
 - `hibernate.connection.url = jdbc:postgresql://localhost:5433/grouper`
 - `hibernate.connection.username = grouper`
 - `hibernate.connection.password = whateverYouSet`
- Copy the `grouper_home/lib/jdbcSamples/postgresql.jar` to `grouper_home/lib/custom`
 - `cp lib/jdbcSamples/postgresql.jar lib/custom`
- Change this in `conf/grouper.properties` so ddl can be done:
`db.change.allow.user.1=grouper`
`db.change.allow.url.1=jdbc:postgresql://localhost:5432/grouper`
- Add tables: `bin/gsh.sh -registry -runscript`
 - You should now be able to browse the DB with pgAdmin or whatever tool you use to admin the db
- Check tables: `bin/gsh.sh -registry -check`
 - Should output: NOTE: database table/object structure (ddl) is up to date
- Start gsh and add a subject: `bin/gsh.sh`
`gsh 0% addSubject("mchzyer", "person", "Chris Hyzer")`
`gsh 1% exit`
- In `grouper.properties`, I will change/add these settings:
`groups.wheel.use = true`
`groups.wheel.group = etc:sysadmingroup`
`configuration.autocreate.group.name.0 = etc:sysadmingroup`
`configuration.autocreate.group.description.0 = super users`
`configuration.autocreate.group.subjects.0 = mchzyer`
- Start gsh again: `bin/gsh.sh` see if the user is in the groups. Make a stem. Add a type, and an attribute
`gsh 0% grouperSession = GrouperSession.startRootSession();`
`edu.internet2.middleware.grouper.GrouperSession: f802d876-b876-4315-b76e-0586bcc561b1,'GrouperSystem','application'`
`gsh 1% subject = findSubject("mchzyer");`
`subject: id='mchzyer' type='person' source='jdbc' name='Chris Hyzer'`
`gsh 2% member = MemberFinder.findBySubject(grouperSession, subject);`
`member: id='mchzyer' type='person' source='jdbc' uuid='1324c75e-9435-4c45-97e9-af40f2b71046'`
`gsh 3% member.getGroups();`
`group: name='etc:sysadmingroup' displayName='etc:sysadmingroup'`
`uuid='e8cf5974-97ea-4865-9ac9-719fe3a13134'`
`gsh 4% addRootStem("aStem", "aStem");`
`stem: name='aStem' displayName='aStem' uuid='3e1c5e6e-6dd4-43f0-8b6c-20cb39f01ac8'`
`gsh 5% typeAdd("fubGroup");`
`type: 'fubGroup'`
`gsh 5% typeAddAttr("fubGroup", "gid", AccessPrivilege.READ, AccessPrivilege.ADMIN, false);`
`attribute: 'gid'`
- Lets add the hook and try in gsh. Just add in grouper source tree for simplicity sake

```
package test;
```

```
import edu.internet2.middleware.grouper.Group;  
import edu.internet2.middleware.grouper.GroupType;
```

```

import edu.internet2.middleware.grouper.GroupTypeFinder;
import edu.internet2.middleware.grouper.hooks.beans.HooksContext;
import edu.internet2.middleware.grouper.hooks.beans.HooksGroupBean;

/**
 * add a type after a group insert
 */
public class GroupAddFubHook extends
    edu.internet2.middleware.grouper.hooks.GroupHooks {

    /**
     *
     * @see
     edu.internet2.middleware.grouper.hooks.GroupHooks#groupPostInsert(edu.internet2.middleware.grouper.hooks.beans.HooksCo
     edu.internet2.middleware.grouper.hooks.beans.HooksGroupBean)
     */
    @SuppressWarnings("unchecked")
    @Override
    public void groupPostInsert(HooksContext hooksContext,
        HooksGroupBean postInsertBean) {

        super.groupPostInsert(hooksContext, postInsertBean);

        try {
            Group group = postInsertBean.getGroup();
            GroupType fubGroup = GroupTypeFinder.find("fubGroup");
            group.addType(fubGroup);
            group.setAttribute("gid", "2");
            group.store();

        } catch (Exception e) {
            throw new RuntimeException(e.getMessage(), e);
        }
    }
}

```

[mchyzer@flash2 grouper]\$ ant dist

- Add this line to the hooks section in conf/grouper.properties
hooks.group.class=test.GroupAddFubHook
- See hook work, see new type and new attribute: bin\gsh.sh

```

gsh 0% group = addGroup("aStem", "aGroup5", "aGroup5");
group: name='aStem:aGroup5' displayName='aStem:aGroup5' uuid='b5552545-2ad2-462c-
b5df-67586c987992'
gsh 1% group.getTypes();
type: 'base'
type: 'fubGroup'
gsh 2% group.getAttributes();
java.util.HashMap: {extension=aGroup5, displayExtension=aGroup5, gid=2, name=aStem:aGroup5,
  displayName=aStem:aGroup5}
gsh 3%

```

*

Grouper UI

- Download or unzip grouper-ui
- Run ant - exit
- Edit the build.properties,
 - set the grouper.folder if not ../grouper
- Run ant - dist
- Edit your tomcat_home/conf/server.xml, add a context for the UI
<Engine defaultHost="localhost" name="Catalina">

```

    <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
    <Host appBase="webapps" autoDeploy="true" name="localhost" unpackWARs="true"
xmlNamespaceAware="false" xmlValidation="false">
        <Context docBase="/home/mchyzer/grouper_v1_4/grouper-ui/dist/grouper" path="/grouper"
reloadable="false"/>
    </Host>
</Engine>

```

- Add a user to your tomcat-users.xml file:

```

<tomcat-users>
    <role rolename="grouper_user"/>
    <user username="mchyzer" password="whateveryouwant" roles="grouper_user"/>
</tomcat-users>

```

- If you are using mod_jk, hook up the url with the tomcat:
JkMount /grouper/* tomcat_mchyzer
- Stop and start apache and tomcat:
[mchyzer@flash2 grouper]\$ /home/mchyzer/apache2_0/bin/apachectl stop
[mchyzer@flash2 grouper]\$ /home/mchyzer/apache2_0/bin/apachectl start
[mchyzer@flash2 grouper]\$ /home/mchyzer/tomcat/bin/shutdown.sh
Using CATALINA_BASE: /home/mchyzer/tomcat
Using CATALINA_HOME: /home/mchyzer/tomcat
Using CATALINA_TMPDIR: /home/mchyzer/tomcat/temp
Using JRE_HOME: /opt/appserv/java6
[mchyzer@flash2 grouper]\$ /home/mchyzer/tomcat/bin/startup.sh
Using CATALINA_BASE: /home/mchyzer/tomcat
Using CATALINA_HOME: /home/mchyzer/tomcat
Using CATALINA_TMPDIR: /home/mchyzer/tomcat/temp
Using JRE_HOME: /opt/appserv/java6
[mchyzer@flash2 grouper]\$
- Add a group with or without the fubGroup type, and see the type and attribute when done

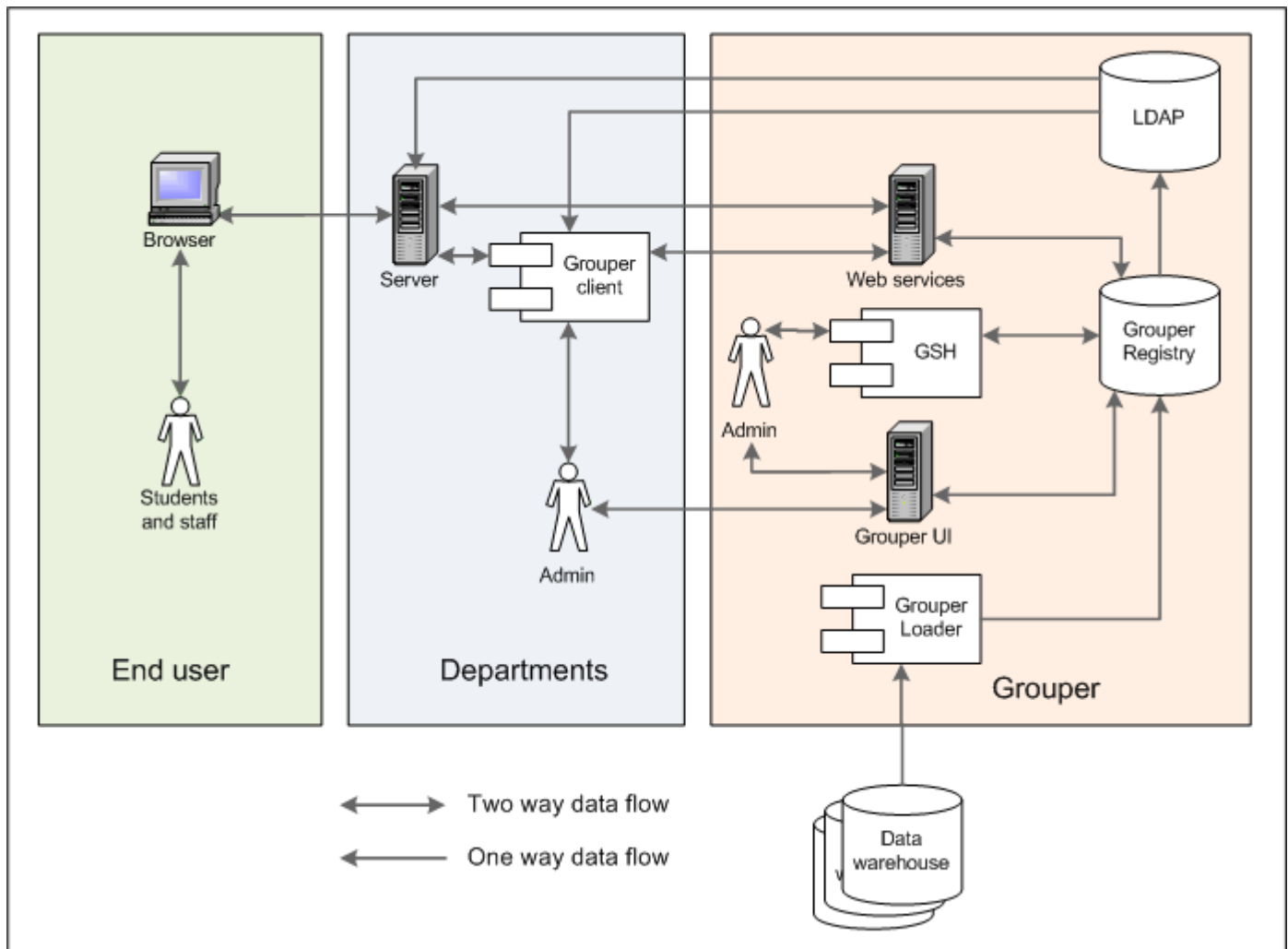
Grouper Client

This page last changed on Sep 15, 2009 by [mchzyer](#).

Grouper Client 1.4.0 is a client for Grouper LDAP and Web Services.

- It is a command line client and Java API
- It is self contained in one jar which will not conflict with any other jars you are using
- It is an example of how to use Grouper, a way to quickly try out proof of concepts, and can be used in production
- Grouper client is aimed at the department user, probably who did not install grouper, but who is using an installation at their institution
- If LDAP and web services are too low level an API for you at some point in your development cycle, grouper client is for you
- Another advantage is the web services client (REST) is forwards compatible, so if Grouper Web Services passes back new XML elements going forward, it will still work with this client.
- You can use this to easily see the HTTP/XML requests and responses to grouper-ws. Note, the headers are not detected, they are assumed to be correct...

Here is a sample grouper architecture diagram which includes the grouper client:



FAQ

- **How can I return pennnames/pennkeys from the web service? (note: pennname/pennkey is the subjectIdentifier at Penn, this is a text based login, not the numeric subjectId)**

First your web service's sources.xml needs to return the subject attribute. I had to make a column in my jdbc source of PENNNNAME (Oracle jdbc metadata makes this always UPPER)

You can specify to return pennnames as the subject attributes, and you can use them in your output template:

```
C:\temp>java -jar grouperClient.jar --operation=getMembersWs --groupNames=test:testGroup --
subjectAttributeNames=PENNNNAME --outputTemplate=${wsSubject.attributeValues[0]}$newline$
bwh
mchalyzer
```

BTW: you can show this on the UI in the custom/media.properties entry:
subject.attributes.order.pennperson=name,description,subjectType,id,PENNNNAME

- **How can I query based on pennkey from the web service? (again pennkey is Penn's subject identifier)**

First you need to understand that in the grouper.client.properties file you can mask "subjectId" and "subjectIdentifier" with terms used at your institution. So you have custom commands. For Penn we can do:

You can use the built in pennkey support in Penn's grouper client (needs custom configuration over the generic Grouper download):

```
C:\temp>java -jar grouperClient.jar --operation=hasMemberWs --groupName=test:testGroup --
pennKeys=mchalyzer,bwh
Index 0: success: T: code: IS_MEMBER: 10099999: true
Index 1: success: T: code: IS_MEMBER: 10099998: true
```

This is due to our grouper.client.properties settings:

```
#note: you will see documentation in the grouper.client.example.properties
grouperClient.alias.subjectIds = pennIds
grouperClient.alias.subjectIdentifiers = pennKeys
grouperClient.alias.subjectId = pennId
grouperClient.alias.subjectIdentifier = pennKey
grouperClient.alias.SubjectId = PennId
grouperClient.alias.SubjectIdentifier = PennKey
```

If you didnt have this customization, you can simply look by subject identifier:

```
C:\temp>java -jar grouperClient.jar --operation=hasMemberWs --groupName=test:testGroup --
subjectIdentifiers=mchalyzer,bwh
Index 0: success: T: code: IS_MEMBER: 10099999: true
Index 1: success: T: code: IS_MEMBER: 10099998: true
```

- **How can I make a group which has a manual membership list and requires users to be faculty student or staff?**

First off, you need permission to view the facultyStudentStaff group, if it is not public. Note, the composite arguments shouldnt be necessary, but until it is fixed, use them and it will work. This makes a group, a system of record group (where the manual entries go), and the overall group is a composite intersection of the manual group and the facultyStudentStaff group. Note you need to enable "requireGroups" in your grouper.properties

```
C:\temp>java -jar grouperClient.jar --operation=groupSaveWs --name=test:isc:astt:chris:myGroup
--includeGroupDetail=true --description="test group with requiring active facultyStudentStaff"
--displayExtension="My test group" --attributeName0=requireAlsoInGroups --
attributeValue0=penn:somewhere:facultyStudentStaff --typeNames=requireInGroups --
compositeType=INTERSECTION --leftGroupName=test:isc:astt:chris:myGroup_systemOfRecord --
rightGroupName=penn:somewhere:facultyStudentStaff
Success: T: code: SUCCESS_INSERTED: test:isc:astt:chris:myGroup
```

Using (from end-user's perspective)

To use grouper client, you need java 1.5+, the grouperClient.jar, and the grouper.client.properties file (either in your classpath, or in the same directory as grouperClient.jar)

To use command line, just type this to see usage:

```
java -jar grouperClient.jar
```

The usage will be specific to your institution... but here is a sample usage:

Grouper Client USAGE:

This program runs queries against grouper ldap and web services

The system exit code will be 0 for success, and not 0 for failure

Output data is printed to stdout, error messages are printed to stderr or logs (configured in grouper.client.properties)

Grouper client webpage: <https://wiki.internet2.edu/confluence/display/GrouperWG/Grouper+Client>

Arguments are in the format: --argName=argValue

Example argument: --operation=encryptPassword

Example argument(OS dependent): --operation="value with whitespace"

Optional arguments below are in [brackets]

```
#####  
## Misc operations
```

Encrypt passwords for storing passwords in external encrypted files:

```
java -jar grouperClient.jar --operation=encryptPassword [--dontMask=true|false]
```

Usage (this message):

```
java -jar grouperClient.jar
```

Send file to web service:

```
java -jar grouperClient.jar --operation=sendFile --urlSuffix=groups/aStem:aGroup/members  
[fileName=theFileName] [fileContents=theFileContents] [--contentType=text/xml] [--  
labelForLog=addMember] [--indentOutput=false] [--saveResultsToFile=fileName] [--debug=true] [--  
clientVersion=someVersion]  
e.g. java -jar grouperClient.jar --operation=sendFile --fileName="C:/addMember.xml" --urlSuffix=groups/  
aStem:aGroup/members
```

```
#####  
## LDAP operations
```

NOTE: CHANGE THIS OR REMOVE IN grouper.client.usage.txt FOR YOUR SCHOOOL'S LDAP QUERIES
pennname to pennid usage:

```
java -jar grouperClient.jar --operation=pennnameToPennid --pennnameToDecode=pennname [--  
saveResultsToFile=fileName] [--outputTemplate=somePattern] [--debug=true]  
e.g.: java -jar grouperClient --operation=pennnameToPennid --pennnameToDecode=jsmith  
output: pennid: 12341234
```

NOTE: CHANGE THIS OR REMOVE IN grouper.client.usage.txt FOR YOUR SCHOOOL'S LDAP QUERIES
pennid to pennname usage:

```
java -jar grouperClient.jar --operation=pennidToPennkey --pennidToDecode=pennid [--  
saveResultsToFile=fileName] [--outputTemplate=somePattern] [--debug=true]  
e.g.: java -jar grouperClient --operation=pennidToPennkey --pennidToDecode=12341234  
output: pennname: jsmith
```

NOTE: CHANGE THIS OR REMOVE IN grouper.client.usage.txt FOR YOUR SCHOOOL'S LDAP QUERIES
hasMember ldap usage:

```
java -jar grouperClient.jar --operation=hasMemberLdap --groupName=a:b:c --pennnameToCheck=pennkey  
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--debug=true]  
e.g.: java -jar grouperClient --operation=hasMemberLdap --groupName=penn:myfolder:mygroup --  
pennnameToCheck=jsmith  
output: hasMemberLdap: true
```

NOTE: CHANGE THIS OR REMOVE IN grouper.client.usage.txt FOR YOUR SCHOOOL'S LDAP QUERIES
getMembers ldap usage:

```
java -jar grouperClient.jar --operation=getMembersLdap --groupName=a:b:c [--saveResultsToFile=fileName]  
[--outputTemplate=somePattern] [--debug=true]  
e.g.: java -jar grouperClient --operation=getMembersLdap --groupName=penn:myfolder:mygroup
```


output: groupList: jsmith, tsmith, msmith
note: extremely large group lists might not display fully (e.g. over 1000 members)

```
#####  
## Web Service operations
```

addMemberWs web service usage (note: you can replace all members of a group also):

```
java -jar grouperClient.jar --operation=addMemberWs --groupName=a:b:c [--subjectIds=subjId0,subjId1]  
[--subjectIdentifiers=subjIdent0,subjIdent1] [--subjectSources=source0,source1] [--subjectIdsFile=fileName]  
[--subjectIdentifiersFile=fileName] [--subjectSourcesFile=fileName] [--defaultSubjectSource=subjectSourceId]  
[--fieldName=fieldNameToAdd] [--txType=GcTransactionType] [--includeGroupDetail=true|false] [--  
includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--replaceAllExisting=true|  
false] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]  
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--  
paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--  
clientVersion=someVersion]  
e.g.: java -jar grouperClient.jar --operation=addMemberWs --groupName=aStem:aGroup --  
subjectIds=12345,23456  
output line: Index 0: success: T: code: SUCCESS: 12345
```

note: if you want to manage group privileges, you can a param: privName to be:
readers,updaters,admins,optins,optouts,viewers. e.g. --paramName0=privName --paramValue0=readers

getMembersWs web service usage:

```
java -jar grouperClient.jar --operation=getMembersWs --groupNames=a:b:c,a:b:d [--  
fieldName=fieldNameToAdd] [--memberFilter=GcMemberFilter] [--includeGroupDetail=true|  
false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--  
actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]  
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--  
paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--  
clientVersion=someVersion]  
e.g.: java -jar grouperClient.jar --operation=getMembersWs --groupNames=aStem:aGroup,aStem:aGroup2  
output line: GroupIndex 0: success: T: code: SUCCESS: group: aStem:aGroup: subjectIndex: 0: 12345
```

note: if you want to manage group privileges, you can a param: privName to be:
readers,updaters,admins,optins,optouts,viewers. e.g. --paramName0=privName --paramValue0=readers

deleteMemberWs web service usage:

```
java -jar grouperClient.jar --operation=deleteMemberWs --groupName=a:b:c [--subjectIds=subjId0,subjId1]  
[--subjectIdentifiers=subjIdent0,subjIdent1] [--subjectSources=source0,source1] [--subjectIdsFile=fileName]  
[--subjectIdentifiersFile=fileName] [--subjectSourcesFile=fileName] [--defaultSubjectSource=subjectSourceId]  
[--fieldName=fieldNameToAdd] [--txType=GcTransactionType] [--includeGroupDetail=true|  
false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--  
actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]  
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--  
paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--  
clientVersion=someVersion]  
e.g.: java -jar grouperClient.jar --operation=deleteMemberWs --groupName=aStem:aGroup --  
subjectIds=12345,23456  
output line: Index 0: success: T: code: SUCCESS: 12345
```

note: if you want to manage group privileges, you can a param: privName to be:
readers,updaters,admins,optins,optouts,viewers. e.g. --paramName0=privName --paramValue0=readers

hasMemberWs web service usage:

```
java -jar grouperClient.jar --operation=hasMemberWs --groupName=a:b:c [--subjectIds=subjId0,subjId1] [--  
subjectIdentifiers=subjIdent0,subjIdent1] [--subjectSources=source0,source1] [--subjectIdsFile=fileName] [--  
subjectIdentifiersFile=fileName] [--subjectSourcesFile=fileName] [--defaultSubjectSource=subjectSourceId]  
[--fieldName=fieldNameToAdd] [--memberFilter=GcMemberFilter] [--includeGroupDetail=true|  
false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--  
actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]  
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--  
paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--  
clientVersion=someVersion]  
e.g.: java -jar grouperClient.jar --operation=hasMemberWs --groupName=aStem:aGroup --  
subjectIds=12345,23456
```

output line: Index 0: success: T: code: IS_MEMBER: 12345: true

getGroupsWs web service usage:

```
java -jar grouperClient.jar --operation=getGroupsWs [--subjectIds=subjId0,subjId1] [--subjectIdentifiers=subjIdent0,subjIdent1] [--subjectSources=source0,source1] [--subjectIdsFile=fileName] [--subjectIdentifiersFile=fileName] [--subjectSourcesFile=fileName] [--defaultSubjectSource=subjectSourceId] [--memberFilter=GcMemberFilter] [--includeGroupDetail=true|false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=getGroupsWs --subjectIds=12345,23456
output line: SubjectIndex 0: success: T: code: SUCCESS: subject: 12345: groupIndex: 0: aStem:aGroup2
```

groupSaveWs web service usage:

```
java -jar grouperClient.jar --operation=groupSaveWs --name=a:b:c [--includeGroupDetail=true] [--txType=transactionType] [--saveMode=SaveMode] [--groupLookupName=a:b:c] [--groupLookupUuid=sd87f-dsf87-sdf89-df78f] [--description=theDescription] [--displayExtension=theDisplayExtension] [--attributeName0=someName] [--attributeValue0=someValue] [--attributeNameX=xthName] [--attributeValueX=xthValue] [--compositeType=COMPLEMENT|INTERSECTION|UNION] [--leftGroupName=compositeLeft] [--rightGroupName=compositeRight] [--groupDetailParamName0=paramName] [--groupDetailParamValue0=paramValue] [--groupDetailParamNameX=xthName] [--groupDetailParamValueX=xthValue] [--typeNameNames=namesOfGroupTypes] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=groupSaveWs --name=aStem:aGroup
output: Success: T: code: SUCCESS_INSERTED: aStem:aGroup
```

stemSaveWs web service usage:

```
java -jar grouperClient.jar --operation=stemSaveWs --name=groupName [--txType=transactionType] [--saveMode=SaveMode] [--stemLookupName=theName] [--stemLookupUuid=theUuid] [--description=theDescription] [--displayExtension=theDisplayExtension] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=stemSaveWs --name=aStem:someStem
output: Success: T: code: SUCCESS_INSERTED: aStem:someStem
```

groupDeleteWs web service usage:

```
java -jar grouperClient.jar --operation=groupDeleteWs --groupNames=groupName0,groupName1 [--txType=GcTransactionType] [--includeGroupDetail=true|false] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=groupDeleteWs --groupNames=aStem:aGroup0,aStem:aGroup1
output line: Index 0: success: T: code: SUCCESS: aStem:aGroup0
```

stemDeleteWs web service usage:

```
java -jar grouperClient.jar --operation=stemDeleteWs --stemNames=a:b,a:c [--txType=GcTransactionType] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=stemDeleteWs --groupNames=aStem:aStem0,aStem:aStem1
output line: Index 0: success: T: code: SUCCESS: aStem:aStem0
```

getGrouperPrivilegesLiteWs web service usage

```
java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs [--groupName=a:b:c] [--stemName=a:b] [--privilegeName=admin|view|read|optin|optout|update|stem|create|etc] [--privilegeType=access|naming|etc] [--subjectId=subjId0] [--subjectIdentifier=subjIdent0] [--subjectSource=source0] [--includeGroupDetail=true|false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]
```

```
[--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=getGrouperPrivilegesLiteWs --groupName=aStem:aGroup --subjectId=test.subject.0
output line: Index 0: success: T: code: SUCCESS: group: aStem:aGroup: subject: test.subject.0: access: admin
```

assignGrouperPrivilegesLiteWs web service usage

```
java -jar grouperClient.jar --operation=assignGrouperPrivilegesLiteWs --privilegeName=admin|view|read|optin|optout|update|stem|create|etc --allowed=true|false [--groupName=a:b:c] [--stemName=a:b] [--privilegeType=access|naming|etc] [--subjectId=subjId0] [--subjectIdentifier=subjIdent0] [--subjectSource=source0] [--includeGroupDetail=true|false] [--includeSubjectDetail=true|false] [--subjectAttributeNames=name0,name1] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--saveResultsToFile=fileName] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=assignGrouperPrivilegesLiteWs --groupName=aStem:aGroup --subjectId=test.subject.0 --privilegeName=admin --allowed=true
output: Success: T: code: SUCCESS: group: aStem:aGroup: subject: test.subject.0: access: admin
```

findGroupsWs web service usage

```
java -jar grouperClient.jar --operation=findGroupsWs --queryFilterType=AND|MINUS|OR|FIND_BY_APPROXIMATE_ATTRIBUTE|FIND_BY_EXACT_ATTRIBUTE|FIND_BY_GROUP_NAME_APPROXIMATE|FIND_BY_GROUP_NAME_EXACT|FIND_BY_GROUP_UUID|FIND_BY_STEM_NAME|FIND_BY_TYPE|etc [--groupName=a:b:c] [--groupUuid=12as-1234gjth] [--stemName=aStem:someStem] [--stemUuid=sfds-sds234] [--stemNameScope=ONE_LEVEL|ALL_IN_SUBTREE] [--groupTypeName=someName] [--groupAttributeName=someName] [--groupAttributeValue=someValue] [--includeGroupDetail=true|false] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=findGroupsWs --queryFilterType=FIND_BY_GROUP_NAME_APPROXIMATE --groupName=aStem:aGroup
output: Index 0: name: aStem:aGroup, displayName: A stem:A Group
```

Note: to specify group math, use queryFilterType of AND|OR|MINUS, and then specify attribute for the left group with a 0 after attribute name, and 1 for the right group.

```
e.g.: java -jar grouperClient.jar --operation=findGroupsWs --queryFilterType=OR --queryFilterType0=OR --queryFilterType00=FIND_BY_GROUP_NAME_APPROXIMATE --groupName00=aStem:aGroup --queryFilterType01=FIND_BY_GROUP_NAME_APPROXIMATE --groupName01=aStem:aGroup --queryFilterType1=FIND_BY_GROUP_NAME_APPROXIMATE --groupName1=aStem:aGroup
```

Note: it is not clear which attributes go with which filter types, the rules are in the Java class: WsQueryFilterType
or use trial and error

findStemsWs web service usage

```
java -jar grouperClient.jar --operation=findStemsWs --queryFilterType=AND|MINUS|OR|FIND_BY_APPROXIMATE_ATTRIBUTE|FIND_BY_PARENT_STEM_NAME|FIND_BY_STEM_NAME|FIND_BY_STEM_NAME_APPROXIMATE|FIND_BY_STEM_UUID|etc [--stemName=a:b:c] [--stemUuid=12as-1234gjth] [--parentStemName=aStem:someStem] [--parentStemNameScope=ONE_LEVEL|ALL_IN_SUBTREE] [--stemAttributeName=someName] [--stemAttributeValue=someValue] [--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source] [--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1] [--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--clientVersion=someVersion]
e.g.: java -jar grouperClient.jar --operation=findStemsWs --stemQueryFilterType=FIND_BY_STEM_NAME_APPROXIMATE --stemName=aStem:aGroup
output: Index 0: name: aStem:aStem0, displayName: A stem:A Stem 0
```

Note: to specify group math, use stemQueryFilterType of AND|OR|MINUS, and then specify attribute for the left stem with a 0 after attribute name, and 1 for the right stem.

```
e.g.: java -jar grouperClient.jar --operation=findStemsWs --stemQueryFilterType=OR --stemQueryFilterType0=OR --stemQueryFilterType00=FIND_BY_STEM_NAME --
```

```
stemName00=aStem --stemQueryFilterType01=FIND_BY_STEM_NAME --stemName01=aStem --
stemQueryFilterType1=FIND_BY_STEM_NAME --stemName1=aStem
```

Note: it is not clear which attributes go with which filter types, the rules are in the Java class:
WsStemQueryFilterType
or use trial and error

memberChangeSubjectWs web service usage (note: you need to be in the sysAdminGroup or actAs someone who is)

```
java -jar grouperClient.jar --operation=memberChangeSubjectWs [--oldSubjectId=oldId] [--
oldSubjectIdentifier=oldIdent] [--oldSubjectSource=oldSourceId] [--newSubjectId=newId] [--
newSubjectIdentifier=newIdent] [--newSubjectSource=newSourceId] [--deleteOldMember=false]
[--actAsSubjectId=subjId] [--actAsSubjectIdentifier=subjIdent] [--actAsSubjectSource=source]
[--outputTemplate=somePattern] [--paramName0=name0] [--paramValue0=value1]
[--paramNameX=xthParamName] [--paramValueX=xthParamValue] [--debug=true] [--
clientVersion=someVersion]
e.g.: --operation=memberChangeSubjectWs --oldSubjectId=test.subject.0 --newSubjectId=test.subject.1 --
actAsSubjectId=GrouperSystem
output: Success: T: code: SUCCESS: oldSubject: test.subject.0, newSubject: test.subject.1
```

```
#####
```

Common options:

```
--outputTemplate=${index}: ${wsGroup.name}
the output template allow the caller to customize what is displayed in the output from the XML
anything in ${} will be evaluated, and there are different variables available for various operations.
if you pass in --debug=true, it will tell you the xml and the variables you can use. You can drill down
in the variables, e.g. ${wsGroupDeleteResult.wsGroup.name}, you can do operations, e.g. ${index+1},
you can do simple string utilities from GrouperClientUtils or GrouperClientCommonUtils, e.g.
${grouperClientUtils.trimToEmpty(wsGroup.name)}

--debug=true
this will display debug information including the request and response to stderr

--saveResultsToFile=/tmp/somefile.txt
you can save the stdout to a file if you like

--actAsSubjectId=subjId --actAsSubjectIdentifier=subjIdent --actAsSubjectSource=source
if you want to run the operation as a different user than the user who is authenticating
to the web service, then specify the actAsSubjectId or actAsSubjectIdentifier (and optionally
the actAsSubjectSource). You would do this e.g. to run a command as admin, or as a user who
is using the end layer application. Note you need permissions to do this in grouper.

--paramName0=name0 --paramValue0=value1 --paramNameX=xthParamName --
paramValueX=xthParamValue
you can specify params in name/value pairs if the operation supports it (see grouper
web service documentation for details)

--clientVersion=someVersion
generally this does not need to be changed. This is the version label sent to the web service
which might affect the output from the web service. Not it does not affect the request to the
web service (besides the label), it only affect the response from the web service.

--txType=GcTransactionType
affects how batched operations are executed on the server (e.g. adding multiple subjects to a group)
generally the only values which make sense are to use a large transaction or not: READ_WRITE_NEW, NONE

--includeGroupDetail=true
if applicable, this option will return not only the group's name, but more information such as the
attributes, types, composite members, etc.

--subjectAttributeNames=a,b,c
if applicable, subjects will be returned from the server with these attributes in a string array
```

To use grouperClient as a Java API, just add the grouperClient.jar to your classpath (e.g. in your WEB-INF/lib directory for a web app), add grouper.client.properties to your classpath, then use the classes generally in the edu.internet2.middleware.grouperClient.api package. e.g.

```
WsAddMemberResults wsAddMemberResults = new
GcAddMember().assignGroupName("aStem:aGroup").addSubjectId("12345").execute();
```

Here is an example of finding a stem

```
/*
 * @author mchzyer
 * $Id$
 */
package edu.internet2.middleware.grouperClient.poc;

import edu.internet2.middleware.grouperClient.api.GcFindStems;
import edu.internet2.middleware.grouperClient.ws.beans.WsFindStemsResults;
import edu.internet2.middleware.grouperClient.ws.beans.WsResultMeta;
import edu.internet2.middleware.grouperClient.ws.beans.WsStem;
import edu.internet2.middleware.grouperClient.ws.beans.WsStemQueryFilter;

/**
 *
 */
public class FindStem {

    /**
     * @param args
     */
    public static void main(String[] args) {

        GcFindStems gcFindStems = new GcFindStems();

        WsStemQueryFilter wsStemQueryFilter = new WsStemQueryFilter();
        wsStemQueryFilter.setStemName("penn");
        wsStemQueryFilter.setStemQueryFilterType("FIND_BY_STEM_NAME_APPROXIMATE");

        gcFindStems.assignStemQueryFilter(wsStemQueryFilter);

        WsFindStemsResults wsFindStemsResults = gcFindStems.execute();

        WsResultMeta resultMetadata = wsFindStemsResults.getResultMetadata();

        if (!"T".equals(resultMetadata.getSuccess())) {
            throw new RuntimeException("Error finding stems: " + resultMetadata.getSuccess()
                + ", " + resultMetadata.getResultCode()
                + ", " + resultMetadata.getResultMessage());
        }

        WsStem[] wsStems = wsFindStemsResults.getStemResults();

        if (wsStems != null) {
            for (WsStem wsStem : wsStems) {
                System.out.println(wsStem.getName());
            }
        }
    }
}
```

Note: you can use method chaining for compact usage, or put each parameter in its own statement.

Use cases currently implemented

1. Connect to LDAP SSL from windows freely and easily
2. Run LDAP lookups that might not even be grouper related (e.g. we want to convert a pennid to a pennkey, in our grouper ldap)
3. Run LDAP queries for hasMember, getMembers
4. Get the members of a group from LDAP, and store in a file, with each subjectId on a line
5. Get the members of a group from web services, and store a certain subject attribute in a file, with each one on its own line
6. Take a file of subjectIdentifiers, and replace the members of a group with it
7. All of the web service calls (non-Lite where applicable)
8. Send an XML file to web services, receive an XML file
9. Encrypt passwords

Using (from Grouper deployer's perspective)

Get the binary release:

1. Unzip, configure the grouper.client.properties and grouper.client.usage.txt, and use

Get the source release, unzip, cd to the dir

```
ant
##### NOW CUSTOMIZE THE conf/grouper.client.properties, conf/grouper.client.usage.txt, misc/README.txt
ant
##### OUTPUT is in dist dir: grouperClient.jar, or grouperClient.institution-1.4.0.tar.gz
```

Checkout grouper client:

```
cvs -d:pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi login
cvs -d:pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co grouper-misc/grouperClient
cd grouper-misc\grouperClient
ant
##### NOW CUSTOMIZE THE conf/grouper.client.properties, conf/grouper.client.usage.txt, misc/README.txt
ant
##### OUTPUT is in dist dir: grouperClient.jar, or grouperClient.institution-1.4.0.tar.gz
```

Now you can zip up the grouperClient.jar, grouper.client.properties, and a README.txt and post on a website for your department users to download. Of course no credentials should be in the zip, the users can fill these in when they are authorized.

Customizing the grouper.client.properties

Here is the example grouper.client.properties

```
#
# Grouper client configuration
# $Id: grouper.client.example.properties,v 1.3 2008/12/01 07:40:28 mchlyzer Exp $
#

#####
## LDAP connection settings
#####

# url of directory, including the base DN (distinguished name)
# e.g. ldap://server.school.edu/dc=school,dc=edu
# e.g. ldaps://server.school.edu/dc=school,dc=edu
grouperClient.ldap.url =

# kerberos principal used to connect to ldap
grouperClient.ldap.kerberosPrincipal =

# password for shared secret authentication to ldap
# or you can put a filename with an encrypted password
grouperClient.ldap.password =
```

The above section is generally for the user, though the url can be filled in when distributing to users

```
#####  
## Web service Connection settings  
#####  
  
# url of web service, should include everything up to the first resource to access  
# e.g. http://groups.school.edu:8090/grouperWs/servicesRest  
# e.g. https://groups.school.edu/grouperWs/servicesRest  
grouperClient.webService.url =  
  
# kerberos principal used to connect to web service  
grouperClient.webService.kerberosPrincipal =  
  
# password for shared secret authentication to web service  
# or you can put a filename with an encrypted password  
grouperClient.webService.password =
```

The above section is generally for users, though the url can be filled in before distributing to users

```
#####  
## Encrypted password settings  
#####  
  
# Put a random alphanumeric string (Case sensitive) for the password encryption. e.g. fh43IRJ4Nf5  
# or put a filename where the random alphanumeric string is.  
# e.g. c:/whatever/key.txt  
# e.g. sdfklj24lkj34lk34  
encrypt.key =  
  
# set this to true if you have slashes in your passwords and dont want to look in external files or unencrypt  
encrypt.disableExternalFileLookup = false
```

grouperClient contains a version of i2mi morphString to keep passwords encrypted in external files from the config file (e.g. so the config file can be more safely distributed, or stored in version control)

```
#####  
## Logging  
#####  
  
# For java.util.logging, only for the grouperClient package (not below)  
# from java java.util.logging.Level class: ALL, CONFIG, FINE, FINER, FINEST, OFF, SEVERE, WARNING  
grouperClient.logging.grouperClientOnly.logLevel = WARNING  
  
# If you are not using log4j (will use java.util.logging, you can turn logging on which will go to stderr  
# (if no file specified below). This is default log level  
# from java java.util.logging.Level class: ALL, CONFIG, FINE, FINER, FINEST, OFF, SEVERE, WARNING  
grouperClient.logging.logLevel = WARNING  
  
# If you dont want the logging to go to stderr, then put a lot file location here: e.g. f:/temp/grouperClient.log  
grouperClient.logging.logFile =  
  
# if you want ws requests and responses being logged to files, put the directory here.  
# The grouper client will create subdirs  
grouperClient.logging.webService.documentDir =  
  
# try to indent the xml. If this fails for some reason, or you want the raw xml,  
# set to false
```

```
grouperClient.logging.webService.indent = true
```

grouperClient contains a version of commons logging. So if used by itself, very basic logging will be used, either to stderr, or to a log file is specified above. There is a lot of debug logging, so if you are having issues, set the grouperClient.logging.grouperClientOnly.logLevel to ALL. You can segregate the grouperClient log level, from everything else. Also, you can store all web service files (e.g. for debugging purposes). Also, these can be indented for easy reading. Note: you can pass in --debug=true to see debug logs and web service request/responses in stderr.

```
#####
#####
#### Institutional and advanced settings
#####
#####

#####
## output templates
#####

webService.addMember.output = Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsSubject.id}$newline$
webService.getMembers.output = GroupIndex ${groupIndex}: success: ${resultMetadata.success}:
code: ${resultMetadata.resultCode}: group: ${wsGroup.name}: subjectIndex: ${subjectIndex}:
${wsSubject.id}$newline$
webService.deleteMember.output = Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsSubject.id}$newline$
webService.hasMember.output = Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsSubject.id}: ${hasMember}$newline$
webService.getGroups.output = SubjectIndex ${subjectIndex}: success: ${resultMetadata.success}:
code: ${resultMetadata.resultCode}: subject: ${wsSubject.id}: groupIndex: ${groupIndex}:
${wsGroup.name}$newline$
webService.groupSave.output = Success: ${resultMetadata.success}: code: ${resultMetadata.resultCode}:
${wsGroup.name}$newline$
webService.stemSave.output = Success: ${resultMetadata.success}: code: ${resultMetadata.resultCode}:
${wsStem.name}$newline$
webService.groupDelete.output = Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsGroup.name}$newline$
webService.stemDelete.output = Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsStem.name}$newline$
webService.getGrouperPrivilegesLite.output = Index ${index}: success: ${resultMetadata.success}:
code: ${resultMetadata.resultCode}: ${objectType}: ${objectName}: subject: ${wsSubject.id}:
${wsGrouperPrivilegeResult.privilegeType}: ${wsGrouperPrivilegeResult.privilegeName}$newline$
webService.assignGrouperPrivilegesLite.output = Success: ${resultMetadata.success}:
code: ${resultMetadata.resultCode}: ${objectType}: ${objectName}: subject:
${wsSubject.id}: ${wsAssignGrouperPrivilegesLiteResult.privilegeType}:
${wsAssignGrouperPrivilegesLiteResult.privilegeName}$newline$
webService.findGroups.output = Index ${index}: name: ${wsGroup.name}, displayName:
${wsGroup.displayName}$newline$
webService.findStems.output = Index ${index}: name: ${wsStem.name}, displayName:
${wsStem.displayName}$newline$
webService.memberChangeSubject.output = Success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: oldSubject: ${wsSubjectOld.id}, newSubject: ${wsSubjectNew.id}$newline$
```

Note that the settings the end user is likely to need to change are up top in the config file. Output templates are central to grouper client, so that the command line output can be parsed easily by clients, or used in other programs. The syntax is Java EL, and uses a version of the jakarta library jexl. In different circumstances different objects are in scope, this needs more documentation and examples, but the point is that you can customize the output to suit your needs, and make sure it will not change with upgrades.

```
#####
## ldap queries
#####

# operation name
ldapSearchAttribute.operationName.0 = pennnameToPennid
ldapSearchAttribute.ldapName.0 = ou=pennnames
ldapSearchAttribute.matchingAttributes.0 = pennname
ldapSearchAttribute.matchingAttributeLabels.0 = pennnameToDecode
ldapSearchAttribute.returningAttributes.0 = pennid
ldapSearchAttribute.outputTemplate.0 = pennid: ${pennid}
ldapSearchAttribute.resultType.0 = STRING

ldapSearchAttribute.operationName.1 = pennidToPennname
ldapSearchAttribute.ldapName.1 = ou=pennnames
ldapSearchAttribute.matchingAttributes.1 = pennid
ldapSearchAttribute.matchingAttributeLabels.1 = pennidToDecode
ldapSearchAttribute.returningAttributes.1 = pennname
ldapSearchAttribute.outputTemplate.1 = pennname: ${pennname}
ldapSearchAttribute.resultType.1 = STRING

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 = isInGroup: ${resultBoolean}
ldapSearchAttribute.resultType.2 = BOOLEAN

ldapSearchAttribute.operationName.3 = getMembersLdap
ldapSearchAttribute.ldapName.3 = ou=groups
ldapSearchAttribute.matchingAttributes.3 = cn
ldapSearchAttribute.matchingAttributeLabels.3 = groupName
ldapSearchAttribute.returningAttributes.3 = hasMember
ldapSearchAttribute.outputTemplate.3 = ${resultString}$newline$
ldapSearchAttribute.resultType.3 = STRING_LIST
```

The LDAP API is very generic. Right now simple attribute lookups are supported, checking to see if there is an attribute match, or listing a multi-valued attribute. More documentation is needed here, and perhaps more options, let us know what you need for ldap access.

```
#####
## Authentication settings
#####

# user prefix
grouperClient.ldap.user.prefix = uid=

# user suffix
grouperClient.ldap.user.suffix = ,ou=entities,dc=upenn,dc=edu

# config name for the ldap user name between prefix and suffix
grouperClient.ldap.user.label = kerberosPrincipal

# config name for the webService user name between prefix and suffix
grouperClient.webService.user.label = kerberosPrincipal

#version of the output, as we upgrade the client, we will maintain previous output versions
grouperClient.output.version = 1.4.0
```

To authenticate to LDAP the username might not need to be exposed to the end user, so you can put a prefix and suffix here. Also, the label used in the config file for the login id can be customized to make it easier to use.

```
#####
## Web service settings
#####

# web service client version
grouperClient.webService.client.version = v1_4_000

# socket timeout
grouperClient.webService.httpSocketTimeoutMillis = 90000

# connection manager timeout
grouperClient.webService.httpConnectionManagerTimeoutMillis = 90000

# ignore extraneous xml fields from server (e.g. on server upgrade, when the client isnt upgraded)
# if you dont ignore, and there is an extraneous field which is not omitted (below), then an exception
# will be thrown
grouperClient.webService.ignoreExtraneousXmlFields = true

# register fields to be ignored with xstream. this is useful if you are not
# ignoring extraneous fields (above), but know that there are a few to be ignored
# place them here with fully qualified classname dont property name, comma separated
# e.g. edu.internet2.middleware.grouperClient.ws.beans.WsResponseMeta.millis,
# edu.internet2.middleware.grouperClient.ws.beans.WsResponseMeta.millis2
grouper.webService.omitXmlProperties =
```

The timeouts and client version are stored here. If you want to ignore some XML that clients send that is not valid (e.g. if the service has changed, and there are old clients), then you can specify here. Also you can call out any specific properties in objects to ignore (inbound or outbound)

```
#####
## Misc
#####

# if there are extra command line args, should we fail or just log?
grouperClient.failOnExtraCommandLineArgs = true

# you can have aliases for subjectId and subjectIdentifier in command line args
# (though subjectId will still be allowed, but you cant pass both)
# if this value is pennIds, then e.g. for addMemberWs, you can use --pennIds=123,234
# instead of --subjectIds=123,345
grouperClient.alias.subjectIds =

# if this value is pennKeys, then e.g. for addMemberWs, you can use --pennKeys=abc,bcd
# instead of --subjectIdentifiers=abc,bcd
grouperClient.alias.subjectIdentifiers =

# if this value is pennId, then e.g. for getGrouperPrivilegesLite, you can use --pennId=123
# instead of --subjectId=123
grouperClient.alias.subjectId =

# if this value is pennKey, then e.g. for getGrouperPrivilegesLite, you can use --pennKey=abc
# instead of --subjectIdentifier=abc
grouperClient.alias.subjectIdentifier =

# if this value is PennId, then e.g. for addMemberWs, you can use --actAsPennId=123
# instead of --actAsSubjectId=abc,bcd
grouperClient.alias.SubjectId =

# if this value is PennKey, then e.g. for addMemberWs, you can use --actAsPennKey=abc
# instead of --actAsSubjectIdentifier=abc
grouperClient.alias.SubjectIdentifier =

# this should probably be changed to UTF-8 for international charsets... for US it can be: ISO-8859-1
```

```
grouperClient.default.fileEncoding = ISO-8859-1
```

If an invalid option is passed in, should it throw an error?

Also, you can put aliases on arguments that are for `subjectId` and `subjectIdentifier`. This means that this alternate argument name can be used instead of `subjectId` and `subjectIdentifier`. You can use either the alias or the original name, but not both at the same time. The examples above show what we are doing at Penn, where `subjectId` is `pennId`, and `subjectIdentifier` is `pennKey`.

Output templates

Per the usage readme:

```
--outputTemplate=${index}: ${wsGroup.name}
the output template allow the caller to customize what is displayed in the output from the XML
anything in ${} will be evaluated, and there are different variables available for various operations.
if you pass in --debug=true, it will tell you the xml and the variables you can use. You can drill down
in the variables, e.g. ${wsGroupDeleteResult.wsGroup.name}, you can do operations, e.g. ${index+1},
you can do simple string utilities from GrouperClientUtils or GrouperClientCommonUtils, e.g.
${grouperClientUtils.trimToEmpty(wsGroup.name)}
```

This uses the jakarta library [JEXL](#).

The easiest way to use this, is first to do a request with debug mode:

```
C:\dev_inst\eclipse\workspace_v33\grouperClient\dist>java -jar grouperClient.jar --operation=addMemberWs
--groupName=aStem:aGroup --subjectIds=test.subject.0 --debug=true
```

...

```
##### RESPONSE START (indented) #####
```

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=xxxxxxxxxxx; Path=/grouperWs
X-Grouper-resultCode: SUCCESS
X-Grouper-success: T
X-Grouper-resultCode2: NONE
Content-Type: text/xml
Date: Mon, 08 Dec 2008 05:43:36 GMT
Connection: close
```

```
<WsAddMemberResults>
<results>
  <WsAddMemberResult>
    <wsSubject>
      <resultCode>SUCCESS</resultCode>
      <success>T</success>
      <id>test.subject.0</id>
      <sourceId>jdbc</sourceId>
    </wsSubject>
    <resultMetadata>
      <resultCode>SUCCESS_ALREADY_EXISTED</resultCode>
      <success>T</success>
    </resultMetadata>
  </WsAddMemberResult>
</results>
<wsGroupAssigned>
  <extension>aGroup</extension>
  <displayExtension>aGroup</displayExtension>
  <displayName>aStem:aGroup</displayName>
  <name>aStem:aGroup</name>
  <uuid>01a1d70c-df7c-4ffa-b6ed-f90fa7c37f6b</uuid>
</wsGroupAssigned>
```

```

<resultMetadata>
  <resultCode>SUCCESS</resultCode>
  <resultMessage>Success for: clientVersion: v1_4_000, wsGroupLookup: WsGroupLookup[groupName=aStem:aGroup], subjectLookups: Array size: 1: [0]: WsSubjectLookup[subjectId=test.subject.0]
, replaceAllExisting: false, actAsSubject: null, fieldName: null, txType: NONE, includeGroupDetail:
false, includeSubjectDetail: false, subjectAttributeNames: null
, params: null</resultMessage>
  <success>T</success>
</resultMetadata>
<responseMetadata>
  <millis>453</millis>
  <serverVersion>v1_4_000</serverVersion>
</responseMetadata>
</WsAddMemberResults>
##### RESPONSE END #####
...
DEBUG: Output template: Index ${index}: success: ${resultMetadata.success}: code:
${resultMetadata.resultCode}: ${wsSubject.id}
, available variables: wsAddMemberResults, grouperClientUtils, wsGroupAssigned, index, wsAddMemberResult,
wsSubject, resultMetadata

```

You can see what the available variables are, and you can see the XML response.

So, if you want to print out some variables, you can use the objects available, and drill down by looking at what is available in the xml:

```

C:\grouper>java -jar grouperClient.jar --operation=addMemberWs --groupName=aStem:aGroup --
subjectIds=test.subject.0 --outputTemplate="sourceId: ${wsAddMemberResult.wsSubject.sourceId}, uuid:
${wsGroupAssigned.uuid}"
sourceId: jdbc, uuid: 01a1d70c-df7c-4ffa-b6ed-f90fa7c37f6b
C:\grouper>

```

If you evaluate an expression which returns null (which is what is returned if a variable doesn't exist), a warning will be displayed to stderr, but the stdout will still be intact. If this is intended, use `grouperClientUtils.defaultString()` and it will not return null.

Custom operations

If you want the grouper client to execute custom Java operations, then follow these instructions. For example, at Penn we will have a couple of operations that decode Cosign single-signon tokens.

- Put your code in `grouperClientHome/src/custom` (this is for the code which depends on `ext` or `extCustom` code, but nothing else)
- If you have external code, put that in `grouperClientHome/src/extCustom` (this is for typically 3rd party code which might depend on other jars at compile time, but not runtime)
- If you have jars for `extCustom`, but them in `grouperClientHome/lib/custom`
- The class which responds to an operation should implement the interface: `edu.internet2.middleware.grouperClient.ClientOperation`
- This has one method: `public String operate(OperationParams operationParams);`
- Register this in the `grouper.client.properties`:

```

#####
## Custom operations
## Implement the interface ClientOperation, put it in the jar
## Increment the int index for multiples (must be in order)
#####

```

```

customOperation.name.0 = cosignDecode
customOperation.class.0 = edu.upenn.isc.grouperClient.CosignDecodeOperation

```

* Implement the interface with the logic, and get params from the command line:

```

/**
 * @see
 edu.internet2.middleware.grouperClient.ClientOperation#operate(edu.internet2.middleware.grouperClient.OperationParams)
 */

```

```

public String operate(OperationParams operationParams) {

    Map<String, String> argMap = operationParams.getArgMap();
    Map<String, String> argMapNotUsed = operationParams.getArgMapNotUsed();

    //get params from command line
    String serviceName = GrouperClientUtils.argMapString(argMap, argMapNotUsed, "serviceName", true);
    String cookie = GrouperClientUtils.argMapString(argMap, argMapNotUsed, "cosignCookie", true);

    //get params from grouper.client.properties
    String keyStorePath = GrouperClientUtils.propertiesValue("cosign.keyStorePath", true);

    ... etc, execute the logic, and return the result which should be printed to screen or written to file
}

```

*** Build with: ant**

- Call the operation from the command line:

```

C:\grouperClient\dist\institution\grouperClient.institution-1.4.0>java -jar
grouperClient.jar --operation=cosignDecode --serviceName=cosign-isc-whatever-0 --
cosignCookie=0mmN5ZwyJukNxxxxxxxxxx
203-PennNet ID mchzyer
203-8-digit PennID 123456
203-Timestamp 1111111111
203 IP Address 1.2.3.4
C:\grouperClient\dist\institution\grouperClient.institution-1.4.0>

```

sdf

Misc

If you dont want to validate the SSL (e.g. self signed certificate) follow these instructions in grouperClient.properties

```
# to not require valid SSL, use: edu.internet2.middleware.grouperClient.ssl.EasySslSocketFactory
grouperClient.https.customSocketFactory =
```

```
# to not require valid SSL, use: edu.internet2.middleware.grouperClient.ssl.BlindSslSocketFactory
grouperClient.ldaps.customSocketFactory =
```

Todo

- Add uuids to the client operations

Hibernate and data layer updates

This page last changed on Feb 20, 2008 by [mchzyer](#).

Update to Hibernate 3 Instructions

This is now committed to grouper HEAD. Here are the steps to start using Hibernate3:

1. Sync and get the new jars: hibernate3.2.6, i2micommon, asms, cglib, c3p0, ehcache, backport-util-concurrent. Remove old jars (e.g. hibernate.3.2.5)

2. Change the grouper.properties to use hib3 dao factory

```
#dao.factory=edu.internet2.middleware.grouper.internal.dao.hibernate.HibernateDAOFactory
dao.factory=edu.internet2.middleware.grouper.internal.dao.hib3.Hib3DAOFactory
```

3. Change the grouper.hibernate.properties to use the hib3 ehcache, and hib3 dialect (in this example, mysql, but change this to whatever DB driver type you are using)

```
#hibernate.dialect          = net.sf.hibernate.dialect.MySQLDialect
hibernate.dialect          = org.hibernate.dialect.MySQL5Dialect
```

```
#hibernate.cache.provider_class = net.sf.hibernate.cache.EhCacheProvider
hibernate.cache.provider_class  = org.hibernate.cache.EhCacheProvider
```

4. You must start using c3p0 database pooling (this is the only one we unit test with grouper with). This means changing the grouper.hibernate.properties (feel free to set the c3p0 pool settings as you see fit. Below is a safe version which should perform fine, but you can tune it to err on the side of performance if you like:

```
# Use DBCP connection pooling
#hibernate.dbcp.maxActive      = 16
#hibernate.dbcp.maxIdle       = 16
#hibernate.dbcp.maxWait       = -1
#hibernate.dbcp.whenExhaustedAction = 1
```

```
# Use c3p0 connection pooling (since dbcp not supported in hibernate anymore)
# http://www.hibernate.org/214.html, http://www.hibernate.org/hib\_docs/reference/en/html/session-configuration.html
hibernate.c3p0.max_size 16
hibernate.c3p0.min_size 0
#seconds
hibernate.c3p0.timeout 100
hibernate.c3p0.max_statements 0
hibernate.c3p0.idle_test_period 100
hibernate.c3p0.acquire_increment 1
hibernate.c3p0.validate false
```

5. Check your log4j.properties, if you have TRACE log on hibernate, change to ERROR. If you have net.sf.hibernate, might want to change to org.hibernate. Otherwise ignore.

```
log4j.logger.org.hibernate          = ERROR, grouper_error
```

Proposed objectives

1. Upgrade to the latest version of hibernate, which is now 3.2.5
2. Add consistency to how hibernate and db resources are handled
3. Add support for transactions

Hibernate upgrade

- Blair has already added support for hibernate. When I ran the unit tests with hib3, and half of them failed, but maybe after a little work, many would be fixed
- In order to take advantage of hib3, the properties files needs to be changed regarding: dao.factory, hibernate.cache.provider_class, hibernate.dialect

- Hib2 will be experimental after the switch to hib3, but it is not recommended. It will not support transactions. (GrouperTransaction)
- Any add-ons which use hibernate might need to be tweaked

Hibernate / DB resources

- This goes hand in hand with the transaction support, so let's discuss this too
- Error handling in JDBC and hibernate is repetitive, tedious, and easy to get wrong. There are cases in grouper where the error handling is not done optimally or consistently
- Here is an example

```
public Map findAllAttributesByGroup(String uuid)
    throws GrouperDAOException
{
    Map attrs = new HashMap();
    try {
        Session hs = Hib3DAO.getSession();
        Query qry = hs.createQuery("from Hib3AttributeDAO as a where a.groupUuid = :uuid");
        qry.setCacheable(false);
        qry.setCacheRegion(KLASS + ".FindAllAttributesByGroup");
        qry.setString("uuid", uuid);
        Hib3AttributeDAO a;
        Iterator it = qry.iterate();
        while (it.hasNext()) {
            a = (Hib3AttributeDAO) it.next();
            attrs.put( a.getAttrName(), a.getValue() );
        }
        hs.close();
    }
    catch (HibernateException eH) {
        throw new GrouperDAOException( eH.getMessage(), eH );
    }
    return attrs;
}
```

- If there is an exception thrown before the `hs.close()`, then the session will not be closed, which could leak DB connections or leave resources locked on the database
- If we use inverse of control (like how Spring might do it), then we can handle this centrally. Here is another block we can change:

```
public void addType(GroupDTO _g, GroupTypeDTO _gt)
    throws GrouperDAOException
{
    try {
        Session hs = Hib3DAO.getSession();
        Transaction tx = hs.beginTransaction();
        try {
            hs.save( // new group-type tuple
                new Hib3GroupTypeTupleDAO()
                    .setGroupUuid( _g.getUuid() )
                    .setTypeUuid( _gt.getUuid() )
            );
            hs.saveOrUpdate( Rosetta.getDAO(_g) ); // modified group
            tx.commit();
        }
        catch (HibernateException eH) {
            tx.rollback();
            throw eH;
        }
        finally {
            hs.close();
        }
    }
    catch (HibernateException eH) {
        throw new GrouperDAOException( eH.getMessage(), eH );
    }
}
```

```
}
}
```

Here is how it would look with inverse of control

```
public void addType(final GroupDTO _g, final GroupTypeDTO _gt)
    throws GrouperDAOException {

    HibernateSession.callbackHibernateSession(HibernateTransactionType.READ_WRITE_OR_USE_EXISTING,
        new HibernateHandler() {

            public Object callback(HibernateSession hibernateSession) {
                Session hs = hibernateSession.getSession();
                hs.save( // new group-type tuple
                    new Hib3GroupTypeTupleDAO()
                        .setGroupUuid( _g.getUuid() )
                        .setTypeUuid( _gt.getUuid() )
                );
                hs.saveOrUpdate( Rosetta.getDAO(_g) ); // modified group
                //let HibernateSession commit or rollback depending on if problem or enclosing transaction
                return null;
            }

        });
}
```

- Note that the session does not need to be opened or closed, it is in a callback, which is provided in an anonymous inner class. So the proper exception handling and resource handling will be done everywhere
- Here is a readonly example:

```
public GroupDTO findByName(String name)
    throws GrouperDAOException,
        GroupNotFoundException
{
    try {
        Session hs = Hib3DAO.getSession();
        //TODO CH 20080209 Change this to be one query, not two to attribute then group
        Query qry = hs.createQuery("from Hib3AttributeDAO as a where a.attrName = 'name' and a.value = :value");
        qry.setCacheable(false);
        qry.setCacheRegion(KLASS + ".FindByName");
        qry.setString("value", name);
        Hib3AttributeDAO a = (Hib3AttributeDAO) qry.uniqueResult();
        hs.close();
        if (a == null) {
            throw new GroupNotFoundException("Cannot find group with name: " + name + "");
        }
        return this.findByName( a.getGroupUuid() );
    }
    catch (HibernateException eH) {
        throw new GrouperDAOException( eH.getMessage(), eH );
    }
}
```

Here is how the code would look with inverse of control (notice how data is passed in and out of the anonymous block (final params, and a return object)

```
public GroupDTO findByName(final String name)
    throws GrouperDAOException,
        GroupNotFoundException
{
    Hib3AttributeDAO hib3AttributeDAO = (Hib3AttributeDAO)HibernateSession.callbackHibernateSession(
        HibernateTransactionType.READONLY_OR_USE_EXISTING,
        new HibernateHandler() {

            public Object callback(HibernateSession hibernateSession) {
```



```

        Session hs = hibernateSession.getSession();
        //TODO CH 20080209 Change this to be one query, not two to attribute then group
        Query qry = hs.createQuery("from Hib3AttributeDAO as a where a.attrName = 'name' and a.value
= :value");
        qry.setCacheable(false);
        qry.setCacheRegion(KLASS + ".FindByName");
        qry.setString("value", name);
        Hib3AttributeDAO a = (Hib3AttributeDAO) qry.uniqueResult();
        return a;
    }
});
//this throws exception, keep out of
GroupDTO groupDTO = hib3AttributeDAO == null ? null :
Hib3GroupDAO.this.findByUuid( hib3AttributeDAO.getGroupUuid() );
//handle exceptions out of data access method...
if (groupDTO == null) {
    throw new GroupNotFoundException("Cannot find group with name: '" + name + "'");
}
return groupDTO;
}

```

For simple database actions (mainly single actions), there is a helper framework in HibernateSession to remove the need for inverse of control:

Here is the same block (note, transaction level can be configured, but there is an intelligent default):

```

public GroupDTO findByName(final String name)
    throws GrouperDAOException,
        GroupNotFoundException {

    Hib3AttributeDAO hib3AttributeDAO = HibernateSession.byHqlStatic()
        .createQuery("from Hib3AttributeDAO as a where a.attrName = 'name' and a.value = :value")
        .setCacheable(false)
        .setCacheRegion(KLASS + ".FindByName")
        .setString("value", name).uniqueResult(Hib3AttributeDAO.class);

    //this throws exception, keep out of
    GroupDTO groupDTO = hib3AttributeDAO == null ? null :
    Hib3GroupDAO.this.findByUuid( hib3AttributeDAO.getGroupUuid() );
    //handle exceptions out of data access method...
    if (groupDTO == null) {
        throw new GroupNotFoundException("Cannot find group with name: '" + name + "'");
    }
    return groupDTO;
}

```

Similarly for object based queries, there are helper classes/methods. Here is an example:

```

public String create(GrouperSessionDTO _s)
    throws GrouperDAOException
{
    try {
        Session hs = Hib3DAO.getSession();
        Transaction tx = hs.beginTransaction();
        Hib3DAO dao = (Hib3DAO) Rosetta.getDAO(_s);
        try {
            hs.save(dao);
            tx.commit();
        }
        catch (HibernateException eH) {
            tx.rollback();
            throw eH;
        }
        finally {
            hs.close();
        }
    }
    return dao.getId();
}

```

```

    }
    catch (HibernateException eH) {
        throw new GrouperDAOException( eH.getMessage(), eH );
    }
}

```

This code will look like this:

```

public String create(GrouperSessionDTO _s)
    throws GrouperDAOException {

    Hib3DAO dao = (Hib3DAO) Rosetta.getDAO(_s);
    HibernateSession.byObjectStatic().save(dao);
    return dao.getId();
}

```

- I suggest that this is the only way to get a Session object so that we ensure it is done correctly
- The inputs to the Hib3SessionHandler describe the session returned. If it is readonly the performance can be dramatically increased (Penn has seen this, I can measure to make sure Grouper will reap the benefits also)
- The syntax and tricks to anonymous inner classes can be new and different to programmers who have not used them, but I think with proper documentation and examples they are easy to get the hang of (all the Penn Java developers use them now just fine)

Add support for transactions

- Lets take the use case of web services in batch mode, where the caller wants to add a few new groups, and if one fails, they should all fail. This is not currently possible
- To support transactions we can use the inverse of control described above
- We can support four modes: GrouperTransactionType.READONLY_OR_USE_EXISTING, READONLY_NEW, READ_WRITE_OR_USE_EXISTING, READ_WRITE_NEW
 - READONLY_OR_USE_EXISTING: even if in the middle of a transaction, create a new read/write autonomous nested transaction. If this block is exited normally it will always commit. If exception is thrown, it will always rollback.
 - READONLY_NEW: even if in the middle of a transaction, create a new readonly autonomous nested transaction. Code in this state cannot commit or rollback.
 - READ_WRITE_OR_USE_EXISTING: use the current transaction if one exists. If there is a current transaction, it MUST be read/write or there will be an exception. If there isnt a transaction in scope, then create a new read/write one. If you do not commit at the end, and there is a normal return (no exception), then the transaction will be committed if new, and not if reusing an existing one. If there is an exception, and the tx is new, it will be rolledback. If there is an exception and the tx is reused, the tx will not be touched, and the exception will propagate.
 - READ_WRITE_NEW: even if in the middle of a transaction, create a new read/write autonomous nested transaction. If this block is exited normally it will always commit. If exception is thrown, it will always rollback.
- Note that you can nest transactions to any level (actually there is a hard stop at 15 since I cant picture actually wanting to nest that deep, and I want to try to detect if the ThreadLocals are acting up)
- When we know the ThreadLocals should be empty, we should clear them. e.g. at the start of an HTTPRequest in web services or UI: HibernateSession.resetAllThreadLocals()
- To code the above use case, it would look something like this (note there is an implicit complete if no exception, and rollback if there is an exception, but you can control this if you like with GrouperTransaction methods):

```

/**
 * run multiple logic together
 * @param grouperSession
 * @param groupName
 * @param groupName2
 * @param displayExtension
 * @param groupDescription
 */
public void runLogic(GrouperSession grouperSession, String groupName,
    String groupName2, String displayExtension, String groupDescription) {

```

```

try {

    //insert a group
    Group.saveGroup(grouperSession, groupDescription, displayExtension, groupName,
        null, SaveMode.INSERT, false);

    //insert another group
    Group.saveGroup(grouperSession, groupDescription, displayExtension, groupName2,
        null, SaveMode.INSERT, false);
} catch (StemNotFoundException e) {
    throw new RuntimeException("Stem wasnt found", e);
} catch (Exception e) {
    throw new RuntimeException(e);
}

}

/**
 * show simple transaction
 * @throws Exception if problem
 */
public void testTransaction() {

    final GrouperSession rootSession = SessionHelper.getRootSession();
    final String displayExtension = "testing123 display";
    final String groupDescription = "description";
    final String groupName = "i2:a:testing123";
    final String groupName2 = "i2:b:testing124";

    try {
        R.populateRegistry(2, 2, 0);

        GrouperTest.deleteGroupIfExists(rootSession, groupName);
        GrouperTest.deleteGroupIfExists(rootSession, groupName2);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }

    //demonstrate passing back data with final array
    final Integer[] someInt = new Integer[1];

    //you can pass back one object from return
    String anythingString = (String) GrouperTransaction
        .callbackGrouperTransaction(new GrouperTransactionHandler() {

            public Object callback(GrouperTransaction grouperTransaction)
                throws GrouperDAOException {

                //everything in here will run in one transaction, note how to access "this"
                TestGroup0.this.runLogic(rootSession, groupName, groupName2,
                    displayExtension, groupDescription);

                //pass data back from final array (if need more than just return value)
                someInt[0] = 5;

                //if return with no exception, then it will auto-commit.
                //if exception was thrown it will rollback
                //this can be controlled manually with grouperTransaction.commit()
                //but it would be rare to have to do that

                //pass data back from return value
                return "anything";
            }

        });

    System.out.println(anythingString + ", " + someInt[0]);

```

}

- In this case it is not readonly (defaults to `READ_WRITE_OR_USE_EXISTING`, but can be controlled), and a transaction wrapper is passed to the callback (`GrouperTransaction`). There is a method called "`GrouperTransaction.commit()`" (takes param which can be only if new or always) which will determine if the transaction originated in this callback, or is from a `ThreadLocal` from an outer transaction. If it originated here, it will commit. Else not.
- If you can picture the code in the `addGroup()` method, it will look similar... read/write transaction, with `USE_EXISTING`, and "`commitIfNewTx()`" which when called from here will not commit.
- Since we are supporting nested transactions, there can be a `ThreadLocal` stack of hibernate sessions which can be accessed from anywhere and not need to be passed around to all method signatures.
- All 87 places that use `Hib3DAO.getSession()` were refactored. It will be tedious but should be low impact and should improve the quality of the code
- The higher impact changes to the code relate to how hibernate is handled:
 1. Since the Session (from Hibernate) can only deal with one object by key at a time, each time an object is retrieved from hibernate it must be "evicted" from the session. The `ByObjectStatic` and `ByHqlStatic` helper methods will do this, and if "`Query.list()`" for example is called outside of the helpers, then the results must be evicted. The side effect is that no hibernate dirty checking will be available. This is no problem since that is more pain than it is worth
 2. If a transaction type is a NEW transaction type, then at the end of the `HibernateSession` callback, it will be committed or rolled back (and Session object is discarded, and state is synced with the DB). However, if it is a "USE_EXISTING" transaction type, then at the end of the `HibernateSession`, the Session (from Hibernate) will start bulking up. So `HibernateSession` will `flush()` (write all queries to the wire), and `clear()` (take all objects in the Session out). This will make the transactions work

Hibernate ID's and versioning

This page last changed on Jul 31, 2008 by [mchzyer](#).

How to

To upgrade, get a version of grouper newer than 1.4, and build 7/30/8. Then start grouper or do an: `ant schemaexport`. This will generate a script. You should review it before running it. It will create `hibernate_version_number` cols, and init them in all tables. Also it will create `old_id` and `old_uuid` cols, and copy data from `id` and `uuid` there. Then it will set the `id` to `col` to be the value of the `uuid`. Then it will drop the `uuid` col. When you are satisfied that everything is ok, you should set the `grouper.properties.ddlutils.dropBackupUuidCols` to `true`, and the next time you `schemaexport` it will drop the `old_id` and `old_uuid` cols. Here is a [sample oracle upgrade script](#) (might be out of date)

Troubleshooting

You need to pay attention to each error, and resume the script after fixing problems. You should not have grouper or ant run this script for you. Here is sample oracle troubleshooting:

ORA-01408: such column list already indexed

Details: `CREATE UNIQUE INDEX type_name_idx ON GROUPER_TYPES (NAME)`

>>> resolution, drop unique constraint on name, drop existing index, then run again

Details: `ALTER TABLE GROUPER_MEMBERSHIPS DROP COLUMN MEMBERSHIP_UUID`

ORA-12991: column is referenced in a multi-column constraint

>>> resolution, drop multi-column constraint, then run again

Details: `ALTER TABLE GROUPER_MEMBERSHIPS ADD CONSTRAINT fk_memberships_list_name_type FOREIGN KEY`

`(LIST_NAME, LIST_TYPE) REFERENCES GROUPER_FIELDS (NAME, TYPE)`

ORA-02270: no matching unique or primary key for this column-list

ORA-02429: cannot drop index used for enforcement of unique/primary key

>>> resolution, add a unique constraint on name and type in grouper fields

>>> `ALTER TABLE AUTHZADM.GROUPER_FIELDS ADD CONSTRAINT fields_name_type_unq`

>>> `UNIQUE (NAME, TYPE) ENABLE VALIDATE`

Details: `DROP INDEX SUBJECTATTRIBUTE_ID_NAME_IDX`

ORA-02429: cannot drop index used for enforcement of unique/primary key

>>> this worked when i tried again

Details: `ALTER TABLE SUBJECTATTRIBUTE ADD CONSTRAINT`

`fk_subjectattr_subjectid FOREIGN KEY (SUBJECTID) REFERENCES SUBJECT (SUBJECTID)`

ORA-02298: cannot validate (AUTHZADM.FK_SUBJECTATTR_SUBJECTID)

- parent keys not found

>>> this worked when I tried again

Introduction

In the grouper dev call on 3/5/8 we discussed removing the non-uuid id columns of some tables.

e.g. `grouper_groups` has a column named "id", and a column named "uuid". Both are identifiers for the row. The "id" col is not used in any foreign keys or in any api's and is just used for hibernate as a key, and the primary key of the table. The "uuid" is used for unenforced foreign keys. The following tables are candidates for removing the id cols: `grouper_composites`, `grouper_fields`, `grouper_groups`, `grouper_members`, `grouper_sessions`, `grouper_types`, `grouper_stems`. It seems like `grouper_memberships` is not a candidate since there can be multiple rows with the same `membership_uuid`...

Why remove?

It is confusing to have two id cols for a table. Also, it takes space to keep the index for this other row. I think the current design is not the original design, and we can clean it up.

Challenges

We cannot simply remove the id col, and keep the uuid column. The hibernate id col is used by hibernate to know if a sql change is an update or insert. If there is a key there, it is an update, if not, it is an insert. And the key gets created right when the record gets inserted. However, Grouper's logic design leverages the unenforced foreign keys based on uuid'd which are created before the row is inserted to setup a batch of sql's (complete with unenforced foreign key references) before a single insert occurs.

Solution

We should use the uuid col for the primary key, and for hibernate's identity. We should ditch the id col. To get around the fact that hibernate needs to know if something is an insert or update, we should add a new col which is a number which will be the hibernate version of the row. It starts at 0 and increments with every update.

http://www.hibernate.org/hib_docs/reference/en/html/mapping.html#mapping-declaration-version

However, this feature would throw exceptions if two grouper processes stepped on each other's toes by throwing "StaleState" exceptions. e.g. if the version that a process updates is not the same as when it was selected, that means some other process updated it, and a StaleState exception is thrown, which indicates that the process should refresh the data and make the edits again. This will not work with Grouper for two reasons: we cache data, and we make more updates than are absolutely necessary. e.g. we update the Stem object when a child group is created. So if two grouper processes created a group under the same stem around the same period of time, it would throw exception. Eventually we can think about using real hibernate versioning, but not right now. To use it, we should clean up our updates to only update what has changed (dirty checking), and have more intelligent cache refreshes. Until then we can use versioning that does not throw exceptions if stale. So the version column will not be exactly accurate, but it will be a good approximation. Also, we should remove support for hibernate2 since we are requiring hib3, and these changes make hib2 not compile...

I have made these changes locally, and they are not all that complex or risky. All unit tests pass.

Yes we still have the same number of columns, but now the column is not an id, it is just an int. It also gives good information (how many times updated), does not have an index on it, and greases the skids if we ever go to strict hibernate versioning.

The implementation is done with a normal hibernate column mapping, an interface on the DAOs which are versioned (Hib3HibernateVersioned), and a hibernate interceptor which sets / increments this val for us automatically. Also, before where the hibernate id was passed back from the DAO to the DTO since hibernate originated the data, now the version will. Not a lot of code changes.

Migration path

It is not that hard to migrate the DB of existing grouper deployments. It is a bit easier if the db col is still the "id" col, but we just move the uuid data in there. Then remove the uuid col. We can either see which databases need this and make a simple script for it. Or we could try to use DdlUtils to make an ant script that does this for any db. Before making the changes, a backup should be made, and maybe an export.

Hooks POC (Proof of concept)

This page last changed on Oct 13, 2009 by mchyzer@idp.protectnetwork.org.

This document is about a proof of concept on hooks for a group change and a membership change. Here is another [getting started guide](#).

The current progress is a working unit test for each, and a grouper UI example for a group change (member change is in progress).

Grouper UI group example

I added a hook to the grouper UI which is a veto hook which will not allow a group to be created which is not in the "penn" folder (which is a subfolder of root).

To implement this, it requires a group hook class:

```
/*
 * @author mchyzer
 * $Id: GroupHooksImplExample.java,v 1.1.2.1 2008/06/11 06:19:38 mchyzer Exp $
 */
package edu.internet2.middleware.grouper.ui.hooks;

import org.apache.commons.lang.StringUtils;

import edu.internet2.middleware.grouper.GrouperConfig;
import edu.internet2.middleware.grouper.hooks.GroupHooks;
import edu.internet2.middleware.grouper.hooks.beans.HooksGroupPreInsertBean;
import edu.internet2.middleware.grouper.hooks.HookVeto;
import edu.internet2.middleware.grouper.internal.dao.GroupDAO;

/**
 * test implementation of group hooks for test
 */
public class GroupHooksImplExample extends GroupHooks {

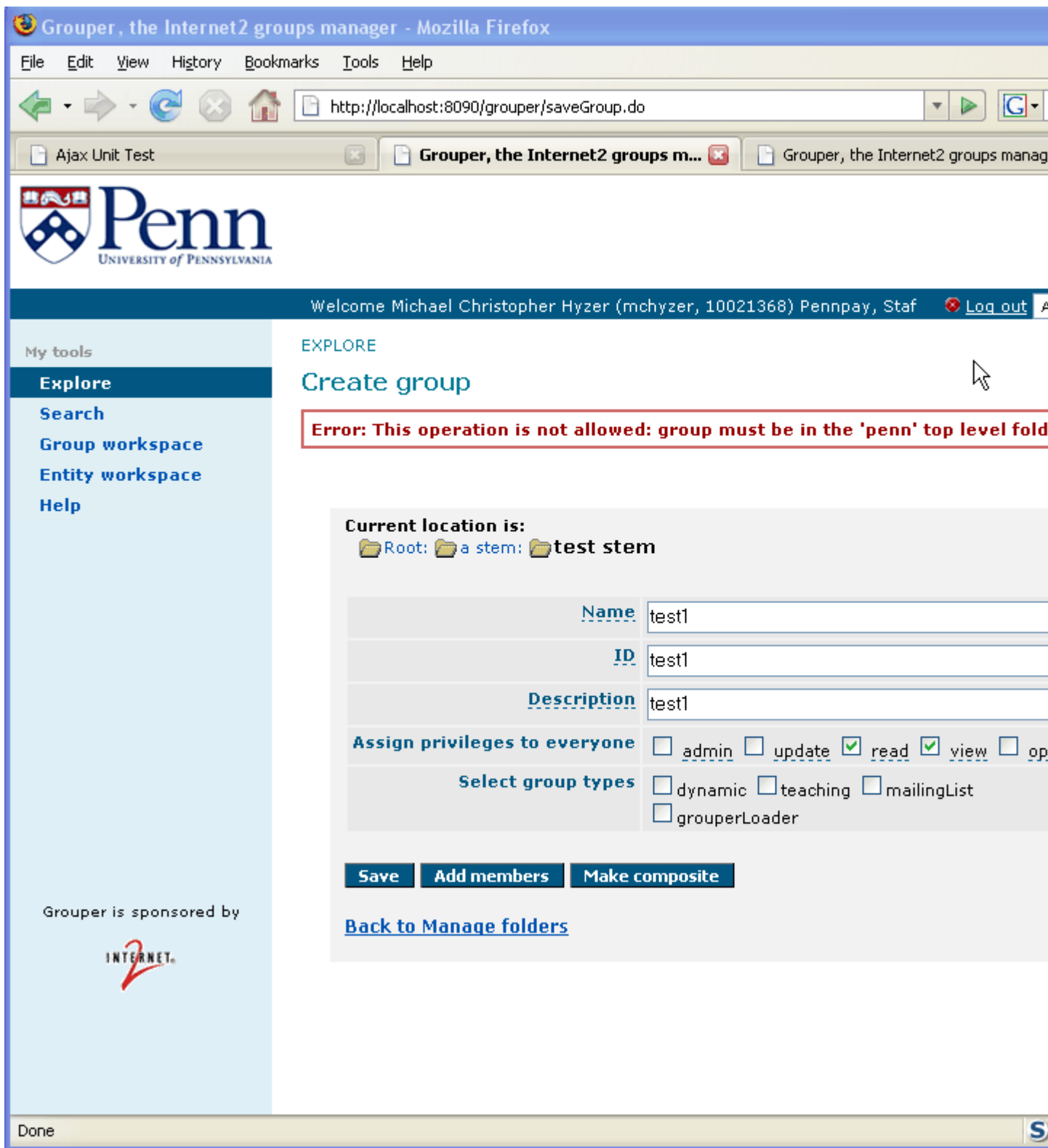
    /**
     * @see
     * edu.internet2.middleware.grouper.hooks.GroupHooks#groupPreInsert(edu.internet2.middleware.grouper.hooks.beans.HooksGroupPreInsertBean)
     */
    @Override
    public void groupPreInsert(HooksGroupPreInsertBean preInsertBean) {

        GroupDAO groupDAO = preInsertBean.getGroupDao();
        String name = StringUtils.defaultString((String)groupDAO.getAttributes().get(GrouperConfig.ATTR_NAME));
        if (!name.startsWith("penn:")) {
            throw new HookVeto("hook.veto.group.name.prefix", "group must be in the 'penn' top level folder");
        }
    }
}
```

And it requires a configuration in the grouper.properties:

```
#implement edu.internet2.middleware.grouper.hooks.GroupHooks
hooks.group.class=edu.internet2.middleware.grouper.ui.hooks.GroupHooksImplExample
```

The result when trying to add a group not in that folder, is:



Grouper UI membership example

I added a hook to the Grouper UI (unfinished... was getting to fancy) for a membership veto that if the group type is grouperLoader, do not allow group memberships. However, if the user is a wheel group user, then allow it but put a warning on screen

Here is the starting point for the Java:


```

/*
 * @author mchzyer
 * $Id: MembershipHooksImplExample.java,v 1.1.2.1 2008/06/11 06:19:38 mchzyer Exp $
 */
package edu.internet2.middleware.grouper.ui.hooks;

import edu.internet2.middleware.grouper.Group;
import edu.internet2.middleware.grouper.GroupType;
import edu.internet2.middleware.grouper.GroupTypeFinder;
import edu.internet2.middleware.grouper.SchemaException;
import edu.internet2.middleware.grouper.hooks.MembershipHooks;
import edu.internet2.middleware.grouper.hooks.beans.GrouperBuiltinContextType;
import edu.internet2.middleware.grouper.hooks.beans.GrouperContextType;
import edu.internet2.middleware.grouper.hooks.beans.HooksContext;
import edu.internet2.middleware.grouper.hooks.beans.HooksMembershipPreAddMemberBean;
import edu.internet2.middleware.grouper.hooks.HookVeto;

/**
 * test implementation of group hooks for test
 */
public class MembershipHooksImplExample extends MembershipHooks {

    /**
     * @see
     edu.internet2.middleware.grouper.hooks.MembershipHooks#membershipPreAddMember(edu.internet2.middleware.grouper.hooks.MembershipHooksImplExample)
     */
    @Override
    public void membershipPreAddMember(HooksMembershipPreAddMemberBean preAddMemberBean) {
        HooksContext hooksContext = preAddMemberBean.getHooksContext();
        GrouperContextType grouperContextType = hooksContext.getGrouperContextType();

        //only care about this if not grouper loader
        if (!grouperContextType.equals(GrouperBuiltinContextType.GROUPER_LOADER)) {

            //if the act as user is in the wheel group, then just admonish
            if (hooksContext.isSubjectActAsInGroup("penn:etc:sysAdminGroup")) {

                //add warning to system

            } else {

                Group group = preAddMemberBean.getGroup();
                GroupType groupType = null;
                try {
                    groupType = GroupTypeFinder.find("grouperLoader");
                } catch (SchemaException se) {
                    throw new RuntimeException(se);
                }
                if (group.hasType(groupType)) {
                    throw new HookVeto("hook.veto.loader.membership", "the membership of this group is automatically managed and does not permit manual changes");
                }
            }
        }
    }
}

```

Here is the grouper.properties config:

```
#implement edu.internet2.middleware.grouper.hooks.MembershipHooks
```

hooks.membership.class=edu.internet2.middleware.grouper.ui.hooks.MembershipHooksImplExample

No screen shot yet

Issues

Everything went pretty smoothly, but...

1. To implement a hook, it will require some understanding about grouper. We should post a bunch of examples. The problem is we have business objects, data transfer objects, and data access objects. Sometimes logic goes through any of these paths. So I added hooks to the data access layer (layer on top of hibernate) so that all operations can be hooked. Sometimes there will be a reference to the business object (e.g. Group), but it doesn't really make sense in some cases (like on an insert, there is no group uuid yet, so the Group object is not created yet, and even if it were, most methods would be invalid).
2. On memberships, the DAO hook will probably not be the useful one. Information like group name or subject id is not even available, it would have to be queried in a lot of cases. In some cases it is known, but it is weird to have it there sometimes and not others.
3. I added a high level membership hook (addMember) which will give the information about what the group name is, subjectId, etc. This is most likely the one that can be used for veto operations. The low level one would most likely be used for auditing. The weird thing about the addMember hook is that some of the objects are business objects, and some are DTOs (see the BaseMemberOf object which encapsulates one member addition). I think people will figure it out... but for the record, I'm not completely bought in to the necessity of having so many data layers. 😊

Implementation of grouper code details

This is implemented in a 1.4 hooks branch in cvs.

- Here is the start of a [groups hook class](#) (you override this to add a hook)
- Here is the start of a [membership hook class](#) (notice high and low level hook methods)
- All DB calls have been refactored to go through grouper's HibernateSession API. The transaction implementation in 1.3 brought us more than halfway there, and this seals the deal. The HibernateSession API will allow events to be registered on each hibernate action in a safe way so that the transaction can still be used (differentiates from the built in hibernate interceptors and I think events though events are not documented well)

e.g. the delete and load methods are just wrappers around the ones in hibernate of the same name. ByObject just separates up the namespace a bit (there are also ByHql, and ByCriteria)

```
HibernateSession.callbackHibernateSession(GrouperTransactionType.READ_WRITE_OR_USE_EXISTING,
    new HibernateHandler() {

        public Object callback(HibernateSession hibernateSession) {
            ByObject byObject = hibernateSession.byObject();
            byObject.delete( byObject.load( Hib3GrouperSessionDAO.class, _s.getId() ) );
            return null;
        }

    });
```

- Each hook will have [its own bean](#) (these aren't finished by any means, but look at [add member](#) for decent example). This is to encapsulate the params passed to the hook (and to facilitate an easy way to get data back from a hook if needbe). This way if any params change, existing implementors will be less likely to have to change their code
- Notice the reference in the hook beans to the [hook context bean](#). This holds information about the current user, and gives utility methods (e.g. is the current user in a certain group). This bean also holds attributes which can be set by the current application. e.g. the UI and WS might give a reference to the HttpServletRequest. These attributes can be set as threadsafe or not, which means when the asynchronous callback is called, all the data will be handled correctly (e.g. in a new thread there is no HttpServletRequest object since that is a weak reference and the request will be over before thread is)
- For vetos, there is [one exception](#) for various types, and it indicates which type of veto it is (set automatically if not manually). This is going to require a little bit of work on the implementors (e.g.

WS, UI, etc) to handle the vetos gracefully. In the UI it was a matter of adding this code (the three lines starting with catch HookVeto). Note this will have to be done in all places where the API is used if it cant be done centrally (hopefully it can be added to filter or something...). 😊

```
try{
    group = parent.addChildGroup(extension,displayExtension );

} catch(HookVeto hookVeto) {

    //this action was vetoed, put explanation on screen, and go back
    Message.addVetoMessageToScreen(request, hookVeto);
    return mapping.findForward(FORWARD_CreateAgain);

} catch(GroupAddException e) {
    String name = parent.getName() + GrouperHelper.HIER_DELIM + extension;
    request.setAttribute("message", new Message(
        "groups.message.error.add-problem",new String[] {e.getMessage()}, true));
    return mapping.findForward(FORWARD_CreateAgain);
}
```

- There was an issue of setters affecting the API, and I think we can fix that. e.g. for group I added a save() method which needs to now be called after setting the description, name, etc. However, there are methods which arent affected (e.g. Grouper.addMember() does not require a save). Only javabeen setters.
- Note that in the UI you can specify a message in the nav.properties for each hook veto, or just use the standard one as a default to reduce the number of steps required to get working...
- Its pretty easy/lightweight to add a hook invocation to the grouper code (I will be adding these)... e.g. here is the one for the addMember:

```
//see if there is a hook class
MembershipHooks membershipHooks = (MembershipHooks)GrouperHookType.MEMBERSHIP.hooksInstance();

if (membershipHooks != null) {
    HooksMembershipPreAddMemberBean hooksMembershipPreUpdateHighLevelBean =
        new HooksMembershipPreAddMemberBean(new HooksContext(), mof);

    membershipHooks.membershipPreAddMember(hooksMembershipPreUpdateHighLevelBean);
}
```

- For the low level hooks, the implementation is similar but even easier (since it is central)... each DAO implements this [interface](#), so those methods just need to be implemented. e.g. in Hib3GroupDAO

```
/**
 * @see
 * edu.internet2.middleware.grouper.internal.dao.hib3.Hib3DAO#onPreSave(edu.internet2.middleware.grouper.hibernate.Hibernate
 */
@Override
public void onPreSave(HibernateSession hibernateSession) {
    super.onPreSave(hibernateSession);

    //see if there is a hook class
    GroupHooks groupHooks = (GroupHooks)GrouperHookType.GROUP.hooksInstance();

    if (groupHooks != null) {
        HooksGroupPreInsertBean hooksGroupPreInsertBean = new HooksGroupPreInsertBean(new
        HooksContext(), this);
        groupHooks.groupPreInsert(hooksGroupPreInsertBean);
    }
}
```

This will kick in wherever a group is saved

- Group / member hooks are [unit tested](#), and all existing unit tests still pass
- sdf

Useful sample Grouper scripts and queries

This page last changed on Oct 16, 2009 by mchyzer@idp.protectnetwork.org.

Note: Carefully review and test all scripts in a test environment before running in prod. You are liable for scripts that you run.

Assign group memberships and privileges like another user

If you want to add a user to all the groups another user is in, you can generate GSH scripts via SQL. Note, this is an oracle script, for mysql you will need to change the || to "concat()". Also, the pagesize and linesize settings are oracle specific. You can omit or replace for other dbs. This will work for 1.4 and 1.5

Group memberships (note, adjust "someExistingSubjectId", and "someNewSubjectId"):

```
set pagesize 10000
set linesize 1000

select 'addMember("'" || gmV.GROUP_NAME || "'", "someNewSubjectId", FieldFinder.find("'" || gmV.LIST_NAME ||
"''));" as command
from grouper_memberships_v gmV
where gmV.subject_id = 'someExistingSubjectId'
AND gmV.membership_TYPE = 'immediate'
and list_type = 'list';
```

Take the output of that, which looks like this:

```
addMember("test:stem:whatever:group", "someNewSubjectId", FieldFinder.find("members"));
addMember("school:hey:there:folder:group2", "someNewSubjectId", FieldFinder.find("members"));
```

Put that in a file called script.txt, and should put this at the top: GrouperSession.startRootSession();

Then run like this: gsh.sh script.txt

For group privileges, adjust for subjectIds and run this query (again, for mysql, use concat instead of ||):

```
select 'grantPriv("'" || gmV.GROUP_NAME || "'", "someNewSubjectId", ' | gmV.list_name || ');' as command
from grouper_memberships_v gmV
where gmV.subject_id = 'someExistingSubjectId'
AND gmV.membership_TYPE = 'immediate'
and list_type = 'access';
```

Put the output in a script file, and add these lines to the top:

```
GrouperSession.startRootSession();
readers = AccessPrivilege.READ;
updaters = AccessPrivilege.UPDATE;
admins = AccessPrivilege.ADMIN;
viewers = AccessPrivilege.VIEW;
optins = AccessPrivilege.OPTIN;
optouts = AccessPrivilege.OPTOUT;

grantPriv("test:stem:whatever:group1", "someNewSubjectId", viewers);
grantPriv("test:stem:whatever:group2", "someNewSubjectId", viewers);
grantPriv("test:stem:whatever:group3", "someNewSubjectId", readers);
```

```
grantPriv("test:stem:whatever:group4", "someNewSubjectId", viewers);
```

Run the script like this: gsh.sh script.txt

For stem privileges adjust for subjectIds and run this query (again, for mysql, use concat instead of ||):

```
select 'grantPriv("'" || gmvm.STEM_NAME || "'", "someNewSubjectId", ' || gmvm.list_name || ');' as command
from grouper_memberships_v gmvm
where gmvm.subject_id = 'someExistingSubjectId'
AND gmvm.membership_TYPE = 'immediate'
and list_type = 'naming';
```

Put the output in a script file, and add these lines to the top:

```
GrouperSession.startRootSession();

creators = NamingPrivilege.CREATE;
stemmers = NamingPrivilege.STEM;

grantPriv("test:stem:whatever:stem1", "someNewSubjectId", creators);
grantPriv("test:stem:whatever:stem2", "someNewSubjectId", creators);
grantPriv("test:stem:whatever:stem3", "someNewSubjectId", creators);
```

Run the script like this: gsh.sh script.txt

Delete an attribute by deleting all assignments of attribute

Note, the long for loop needs all text on one line. This was run on Grouper 1.4

```
C:\mchzyer\isc\dev\grouper_v1_4\grouper\bin>gsh
Using GROUPER_HOME:      C:\mchzyer\isc\dev\grouper_v1_4\grouper\bin\..
Using GROUPER_CONF:      C:\mchzyer\isc\dev\grouper_v1_4\grouper\bin\..\conf
Using JAVA:               "c:\dev_inst\java\bin\java"
using MEMORY:             64m-512m
...
Type help() for instructions
gsh 0% typeAdd("testType");
type: 'testType'
gsh 2% typeAddAttr("testType", "typeAttr", AccessPrivilege.READ, AccessPrivilege.ADMIN, false);
attribute: 'typeAttr'
gsh 3% groupAddType("test:test1", "testType")
true
gsh 5% setGroupAttr("test:test1", "typeAttr", "whatever")
true
gsh 7% groupAddType("test:test2", "testType")
true
gsh 9% setGroupAttr("test:test2", "typeAttr", "whatever2")
true
gsh 11% for(theGroup :
  GrouperDAOFactory.getFactory().getGroup().findAllByApproximateAttr("typeAttr", "%")) { System.out
.println(theGroup.getName()); theGroup.deleteAttribute("typeAttr"); }
test:test1
test:test2
gsh 12% typeDelField("testType", "typeAttr");
```

true

All groups that use groups as members

This will find all groups that use groups as members. Useful for when you are deleting a stem and subgroups. If the groups in the stem are used elsewhere, it will show the relationships (this works in grouper 1.4):

```
select group_parent.name parent_name, gf.NAME , group_member.NAME member_name
from grouper_groups_v group_parent, grouper_groups_v group_member, grouper_memberships gm,
  grouper_members gmember, grouper_fields gf
where group_parent.GROUP_ID = gm.OWNER_ID and gm.member_id = gmember.ID
and gmember.SUBJECT_ID = group_member.GROUP_ID
and group_member.NAME like 'penn:community:employee:orgs:%'
and gf.ID = gm.FIELD_ID;
```

Grouper UI custom authentication example

This page last changed on Oct 13, 2009 by mchzyer@idp.protectnetwork.org.

Gary already has an example of custom Grouper UI authentication, the [Yale CAS auth](#). I configured the grouper UI to work with [Penn's single signon](#), and I thought another example committed to grouper UI cvs would be useful for people (not because you would use Penn's SSO, but because you might integrate with the UI similarly). Here are the steps I used to get it to work:

1. Add a new eclipse project (mine is in the same as UI in contrib, but you will probably keep in your local source control), this will depend on the grouper-ui project. Here is what [my dir looks like](#)
2. There is a [build.xml](#) which will be linked from the grouper-ui build.xml. I kept things simple by not having any build config params, you might have a build.properties...
3. You need a custom web.something.xml which will merge into the web.core.xml. In my case, it is [web.penn.xml](#), and it again is simplified, I just protect all .do resources (even public ones, Im ok having everything protected I think), but the URL we publish will probably be a protected page (e.g. grouper/home.do), and not the public one (grouper/).
4. That [filter we added to the web.xml](#) will do two things, first it will redirect the user to our single signon login screen if there is no detected user. Second, it [wraps the HttpServletRequest object](#) so that any calls to getRemoteUser() (which is what grouper-ui uses to get the logged in user) will get the user from the token passed in from single signon.
5. The [request wrapper](#) caches the user identity, but makes sure the token from single signon doesnt change (e.g. if a user logs in after another user didnt log off). If there is a mismatch, it kills the session and cookies which should allow the user to login again. If the identity is in session cache, use it (since it is expensive to decode a token). If not, then decode the token (if there wasnt a token, the user will be redirected to the login page by the filter above). To decode the token, we use [our jar](#) which calls a command line program which has the security associated with it. Then cache the result.
6. If you dont have an additional-build.xml file for grouper-ui (used to link additional build steps without editing the build.xml directly), then you can copy and [example](#), and rename to additional-build.xml. Edit the contents to point to your build.xml for your auth mechanism (e.g. build.xml above). Note this build file must have a webapp and resources target.
7. In your build.properties for grouper-ui, specify where this additional-build.xml is, e.g.

```
#add an additional build file to incorporate site specific changes
additional.build=additional-build.xml
```

8. For configuration settings, you can either use params in the web.xml (like Gary's example), or a config file (I used media.properties since I dont expect to have to change my settings, but if so I want a way without compiling). Note that my use of media.properties happens before the local.media.properties is considered since the user isnt logged yet, so again, it might not be a good example, up to you. You can look in the [request wrapper](#) for my example.

9. Remove the simple auth in the web.core.xml

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Tomcat login</web-resource-name>
    <url-pattern>/login.do</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <!-- NOTE: This role is not present in the default users file -->
    <role-name>grouper_user</role-name>
  </auth-constraint>
</security-constraint>

<!-- Define the Login Configuration for this Application -->
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Grouper Application</realm-name>
</login-config>
```

```
<!-- Security roles referenced by this web application -->
<security-role>
  <description>
    The role that is required to log in to the Manager Application
  </description>
  <role-name>grouper_user</role-name>
</security-role>
```