


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 and Testing
 - API Configuration
 - 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 Binary Release
 - Grouper - Loader
 - GrouperShell (gsh)
 - Grouper UI Components
 - Grouper UI Development Environment
 - Grouper Use Cases
 - Grouper Web Services
 - Authentication for Grouper Web Services
 - Grouper Web Services FAQ
 - Import-Export
 - Initializing Administration of Privileges
 - Intro FAQ
 - License
 - Media Properties
 - Nav
 - Prerequisites
 - Resources
 - Software Download
 - Database Conversion v1.2.1 - v1.3.0
 - Entity Relationship Diagram For Grouper 1.3.0

- Grouper change log v1.3
 - v1.3-Release Notes
- Specs sheet
- Technical FAQ
- UI Building and Configuration
- UI Customization Guide
- Unresolvable Subject Deletion Utility (USDU)
- 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

Welcome to the Grouper Group Management Toolkit Wiki

Grouper Product	Grouper Project
<p><i>Open viewing. Editing restricted to the Grouper Developers.</i></p> <p>Grouper Product Documentation</p> <p>The Grouper Product space, with the latest software and documentation.</p> <ul style="list-style-type: none">• New! - Grouper v1.4.0 RC1 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://grouper.internet2.edu: Grouper's official website• Grouper Infosheet  <p>(PDF)</p>	<p><i>Open viewing. Editing restricted to Grouper Working Group members.</i></p> <p>The Grouper Working Group</p> <p>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!

Request Editing Permissions

For editing access within the Grouper space, you will need to first obtain a registered Confluence username/password:

1. Please [sign up](#) with your email as your username, and
2. Send an access request to Steve Olshansky <steveo AT internet2 DOT edu>.

Background

As a result of initial investigations by the MACE-Dir-Groups, Grouper was developed as an open source toolkit to address the needs of managing groups. Grouper is designed to function as the core element of a common infrastructure for managing group information across integrated applications and repositories. Grouper combines multiple sources of group information, both automated and manual, in managing memberships and other group information in a Groups Registry, a central information asset complementary to a site's Person Registry.

A few of the benefits of a groups management service, such as Grouper, include:

- the same groups are made available to many applications
- distributed authorities are able to directly manage access information
- sophisticated group management capabilities, such as subgroups and composite groups, to support many access management needs
- common user, web services, command line, and java interfaces for managing groups

Grouper supports several modes of distributed group management:

- per-group assignment of membership update privilege
- per-group assignment of administration of all forms of access
- per-naming stem assignment of entitlement to create groups within each namespace
- per-group opt-in or opt-out entitlement

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 [Contact](#) page.

Related Internet2 Middleware projects

- [Signet](#)
- [Shibboleth](#)

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 us](#).

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

Comments

Getting another ID here is a hassle. I already have a Shib ID from ProtectNetwork. Why cant you guys support ProtectNetwork Shib ID or InCommon or SDSS on this site? Thanks.

Posted by at Sep 18, 2006.

.....

HI, I would not have thought it should be that hard - I had no problems

James
<http://www.software-dungeon.co.uk>

Posted by at Mar 27, 2007.

.....

About Grouper

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

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

Why is Grouper Open Source?

The Grouper product is an open-source project, and is a result of the efforts by the Groups subgroup of [MACE-Dir](#). Grouper aims to satisfy the need for need a collaborative groups management tool, integrating and enabling authenticated access to groups data from applications and repositories across an institution.

Grouper has been under development since 2001, under the guidance of the MACE-Dir-Group and efforts of the development team and working group members.

Grouper is now at a stage fit for production-level use, though the project will continue to evolve and benefit from contributions of the users. The Working Group welcomes all ideas towards the design aspects of functionality and deployment. Any changes and improvements will be a direct result of collaborations between the user-base and the developers. Your continued support of Grouper will further enhance the efforts to date.

- [Licensed](#) under the Apache 2.0 license.
- [Grouper Credits](#)

Internet2 Contributor Software License Agreements: http://members.internet2.edu/intellectualproperty.html#appendix_c

Thanks!

The Grouper team would like to thank all who have contributed to the development of Grouper; in particular, the MACE-Dir-Groups Working Group members, Tom Barton (MACE-Dir-Groups Working Group chair) and Blair Christensen of the University of Chicago, and Gary Brown of the University of Bristol, who have contributed their time, expertise, and enthusiasm for this project: also Jessica Bibbee, Nate Klingenstein, Liene Karels, Renee Frost, Ann West, and Steve Olshansky from Internet2.

Development of this software was supported with funding from [Internet2](#), [University of Chicago](#), [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 us](#).

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



Credits

This page last changed on Aug 12, 2007 by steveo@internet2.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 Nov 22, 2008 by tbarton@uchicago.edu.

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

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

NEW!! [Grouper v1.4.0 Release Candidate 1](#) is now available!

Welcome to the Grouper v1.3.1 Product Documentation Space

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.

Systems Administration

Installation & Configuration

Prerequisites - Establishing the environment in which Grouper will be built and run.
API Building & Testing - Step-by-step instructions to build the Grouper API.
API Configuration - Configuring the API and integrating with existing identity stores.
UI Building & Configuration - Configuring, building, and deploying the UI.
Initializing Administration of Privileges - The very first naming stem and group you should create.
Installing Grouper Web Services - Instructions to build, secure, and deploy Grouper Web Services.

Tools & Topics for On-Going Administration

Import/Export Tool - Documentation for the XML Import/Export tool.
GrouperShell - Documentation for the *gsh* command line utility.
Custom Group Types & Fields - What they are and how to create and delete them.
Unresolvable Subject Deletion Utility - Documentation for the command line *usdu* tool.
Bad Membership Finder Utility - Documentation for the *findbadmemberships* command line script.

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](#)

[API & UI v1.3.1 Javadoc](#)
[WS v1.3.1 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_3_1 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_UI_1_3_1
grouper-ui
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_WS_1_3_1
grouper-ws

```

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

Internet2 Middleware Initiative Commons

This area will house documents and other items that are shared between the projects of the Internet2 Middleware Initiative (I2MI). See also [I2MI-Common in CVS](#).


Subject API- integrates with existing sources of identities whose memberships or privileges are to be managed.

Ldappc - automates the reflection of group, membership, and permission information contained in the Groups and Privileges Registries into a site's LDAP directory service.

Roadmap for I2MI Common Products - notable future enhancements to the I2MI-Common products.

Combined Roadmap for I2MI Common, Grouper, and Signet products.

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: About FAQ Software Documentation Archives Contribute WG Contact
--

[Comments](#)

Those looking to get the latest cvs head for grouper the suitbale cvs invocation seems to be:

```
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 HEAD grouper
```

Posted by [calebracey](#) at Oct 10, 2007.

Step-by-Step Instructions to Build and Test the Grouper API as of v1.3.0

These instructions assume that you have a shell open on the grouper directory. Execute the commands (in bold) in the sequence shown below.

1. **ant build** - This will compile Grouper and place the class files under the grouper/build directory. It also places test instances of three configuration files in the grouper/conf directory. To facilitate testing, these should not be modified at this point in the deployment process.
2. **ant schemaexport** - This will generate the DDL appropriate for the database configured in the grouper/conf/grouper.hibernate.properties file and install the Groups Registry tables. The default database, designed for testing, is located in grouper/dist/run/.
3. **ant db.init** - This 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.

Note: In some environments, this step may fail because the default database was not properly closed during the previous step. We're not yet sure why that happens. However, you can delete the file grouper/dist/run/grouper.lock and repeat this step, which should conclude successfully.

4. **ant test** - This compiles and runs the Grouper test suite to exercise the API and ensure that Grouper is properly configured.

Note: In some environments, this step may fail because the default database was not properly closed during the previous step. We're not yet sure why that happens. However, you can delete the file grouper/dist/run/grouper.lock and repeat this step, which should conclude successfully.

Note: Running the test suite is destructive and will delete all groups and memberships in the Groups Registry. **Do not run the test suite against a production database.**

The Grouper API is now built and tested. Assuming there were no errors, this phase of installation is complete.

One further optional step will ease your use of the API in several contexts:

6. **ant dist** - Builds a grouper.jar file in the grouper/dist/lib directory incorporating all of the Grouper API classes.

And for convenience the following ant target is also provided:

7. **ant dist-lib** - Builds a grouper-lib.jar file in the grouper/dist/lib directory incorporating all of the 3rd party jar's that the Grouper API depends on.

Building the Grouper UI

It is now necessary to configure the API following the instructions in the [API Configuration](#) section.

Only once that has been done can you compile and deploy the UI together with the API jarfile(s), which is done in a coordinated process documented in the [UI Building & Configuring](#) section.

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

API Configuration

This page last changed on Sep 22, 2008 by tbarton@uchicago.edu.

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

Configuring the Grouper API as of v1.3.1

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.

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, changing the display name for internal subjects
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.

Database Driver Location

Place the jar file containing the JDBC driver for your database in the grouper/lib/ folder. The Grouper v1.3.1 package includes the JDBC driver for HSQLDB v1.7.2.11 in this folder. You should replace it if you will be using a different version of HSQLDB.

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 and Grouper API v1.2.1 and earlier - Prior to v1.3.0 Grouper used Apache DBCP for JDBC connection pooling and enabled prepared statement pooling by default. Prepared statement pooling must be disabled for Oracle 9i with the setting:

```
hibernate.dbcp.ps.maxIdle = 0
```

Misc Settings

The operations in ant (or straight from Java) which affect the DB negatively (e.g. dropping and recreating the schema/data during unit tests) will generate a prompt to the user showing the database which will be affected, and confirming that it is ok. This prompt might get annoying or might not work correctly, so in the grouper.properties you can whitelist or blacklist db url/schema combinations. Here is documentation from grouper.example.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

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 [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 v1.3.0 relies on v0.3.1 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.

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 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 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.grouper	The group name of the group whose members are to be considered security-equivalent to GrouperSystem.

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

As of version 1.3.0 the Grouper UI started referring to GrouperAll as *EveryEntity* and GrouperSystem as *GrouperSysAdmin*. 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

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\|Contact Information\|Contact Information|Contact Information|Contact Information].

Bad Membership Finder Utility

This page last changed on Sep 03, 2008 by [shilen](#).

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

Bad Membership Finder Utility

This document is released alongside Grouper v1.3.1.

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

Without any arguments, this utility prints its usage

```
$ ./bin/findbadmemberships.sh
```

```
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.

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

Bad memberships for invalid groups and stems

If you have a bad memberships that's associated with a nonexistent group or stem, you can simply delete the membership using the following SQL. **Be sure to substitute "UUID" for the UUID of the bad membership.**

```
delete from grouper_memberships where membership_uuid='UUID';
commit;
```

Bad memberships for valid groups and stems

If you have a bad membership that's associated with a valid group or stem, you will need to use both SQL and xml-export/xml-import. Please go through the following steps for each group or stem with a bad membership.

1. Export the memberships to a file. **Be sure to substitute "UUID" for the UUID of the group or stem.**

```
ant xml-export -Dcmd="GrouperSystem -id UUID file"
```

2. Remove all memberships for the group or stem using SQL. **Be sure to substitute "UUID" for the UUID of the group or stem.**


```
delete from grouper_memberships where owner_id='UUID';
commit;
```

3. Import the correct memberships.

```
ant xml-import -Dcmd="GrouperSystem -list file"
```

After you have fixed all of your groups and stems with bad memberships, you should re-run the utility to verify that the memberships are now correct.

```
./bin/findbadmemberships.sh -all
```

 Questions or comments?  Contact us.

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

Contact Information

This page last changed on Jul 18, 2007 by jbibbee@internet2.edu.

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

E-Mail Us - If you have questions or comments about the wiki, documentation, and/or the Grouper project, please email us: [grouper-info AT internet2 DOT edu](mailto:grouper-info@internet2.edu) .

Support & Resources

Grouper Working Group

If you are interested in or have questions regarding the development of Grouper, please visit the [Grouper Working Group](#) web or [GrouperWG](#) wiki.

Grouper Mailing Lists

Below is a description of the mailing lists you can join to assist your deployment of Grouper.

To subscribe to any of these lists, send email to [sympa AT internet2 DOT edu](mailto:sympa@internet2.edu) with the following in the *subject line*:

subscribe <list name> <your name>

For example:

subscribe grouper-dev Jane Doe

Grouper-Developers Mailing List: Sites wishing to further participate in the development of Grouper and contribute work towards these efforts are encouraged to join the <grouper-dev AT internet2 DOT edu> mailing list. A bi-weekly Grouper (dev) Working Group conference call is held in discussion of development efforts.

subscribe grouper-users Jane Doe

Questions and comments regarding the implementation of Grouper should be directed to this list. Messages will be archived, and can be accessed for reference. It is anticipated that subsequent discussion will be supported by both the Grouper developers *and* implementing sites. Your contributions here - comments, questions, and concerns - will be of great value to the general Grouper community. This should include, but not be limited to, questions about the documentation, undocumented problems, installation or operational issues, and other product issues that arise.

subscribe mw-announce Jane Doe

Periodic news and announcements about Grouper and other items of broad interest, such as pointers to new standards or discussion lists, major updates to widely-used middleware tools, and information about white papers and best-practices documents. Posting to `mw-announce` is done by Internet2 staff; to have an item considered for posting, send it to `mw-announce-submit AT internet2 DOT edu`

To **unsubscribe** from any of these lists, send email to [sympa AT internet2 DOT edu](mailto:sympa@internet2.edu) with the subject:

unsubscribe grouper-dev
unsubscribe grouper-users
unsubscribe grouper-announce

Archive: [grouper-dev list](#)

Archive: [grouper-users list](#)

Archive: [mw-announce list](#)

Archive: [i2mi-ui list](#) - Internet2 Middleware Initiative SW User Interface Development list

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

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

Custom Group Types, Fields, Attributes, Lists

This page last changed on Apr 25, 2008 by [mchzyer](#).

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

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.

```
<registry>
  <metadata>
    <groupTypesMetaDef>
      <groupTypeDef name='teaching'>
        <field name='academic-staff' required='false' type='list' readPriv='read' writePriv='update' />
      </groupTypeDef>
      <groupTypeDef name='clerical-staff' required='false' type='list' readPriv='read' writePriv='update' />
      <groupTypeDef name='faculty_code' required='false' type='attribute' readPriv='read' writePriv='admin' />
    </groupTypesMetaDef>
  </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: About FAQ Software Documentation Archives Contribute WG Contact

Customising the Grouper UI

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

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

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"/>
</plug-in>
```

```
<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.UoBUIGroupPrivilegeResolv
```

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 implementation from **resources/grouper/media.properties** using the key **repository.browseer.<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,

#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: About FAQ Software Documentation Archives Contribute WG Contact

Grouper UIs

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



Welcome Fiona Windsor [Log out](#) [Act as admin](#) [Change](#)

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/modify privileges.


☐ Keith Benson is a direct member

[Remove selected members](#) [Remove all members](#)

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

 University of
BRISTOL

GroupsManager



All UoB Groups Search Group workspace Entity workspace Assistance

Welcome Fiona Windsor [Log out](#) [Act as admin](#) [Change](#)

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](#)

[feedback](#)

University of Bristol, Senate House,
Tyndall Avenue, Bristol BS8 1TH, UK • Tel: +44 (0)117 928 9000

© 2002-2004 University of Bristol
Updated 01/12/2004 by the Public Relations Office

Developer's Guide to the Grouper API


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

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

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: About FAQ Software Documentation Archives Contribute WG Contact

Glossary

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

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

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>	Privileges that determine what a Subject can do with a Group . They are: <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: <ul style="list-style-type: none">• OPTIN - can add self to the membership, and• OPTOUT - can remove self from membership.	Subject is a UI "entity"
<i>Attribute</i>	A single-valued string associated with a Group or a Naming Stem . By default, Grouper supports six attributes (one of two kinds of Field): <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.• 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	<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"

	<p><i>name</i> attribute is: <i><parent stem>:<extension></i>.</p> <ul style="list-style-type: none"> • 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, 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	<p>A Subject 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</p>

Subject is a UI "entity"

	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> <p>naming stem is a UI "folder"</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	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 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 .
Member	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 .
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.

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. 	Naming privileges are now referred to as Creation privileges and the two types are Create Group (replaces CREATE) and Create Folder (replaces STEM)
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>	Stem is a UI "folder"
Stem	A synonym for a Naming Stem .	Stem is a UI "folder"
Subgroup	A Group that is a Direct Member of another group.	
Subject	<p>An abstraction of any object whose Memberships are to be managed by Grouper. Most Grouper deployments will manage subjects 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 subjects to Grouper by use of the Subject API.</p>	Subject is a UI "Entity"
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: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

UI Terminology

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

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

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 Subject	Entities With Privileges 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 is a indirect privilegee	has direct privileges has indirect privileges	as a member of the group 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
			a fundamental unit (container) of the hierarchy that can have a parent (folder or 'root') or children (folders or groups)
Hierarchy	stem [conceptual]	Folder	
	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	

			<p>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.</p> <p>This does not provide access to manage the group (add membership or edit attributes)</p> <p>a hierarchy Is made up of folders. The folder subfolder relationship define the path through the hierarchy</p> <p>a session specific area where you can store groups that you will need to create compound groups, etc</p> <p>a session specific area where you can store groups that you will need to create compound groups, etc</p>
	Stem privilege	Creation Privileges	
Navigation	saved subjects	Entity Workspace	
	Saved groups	Group Workspace	
Administrative	Search subjects grouperAll	Search EveryEntity	
	GrouperSystem	GrouperSysAdmin	
	WheelGroup	SysadminGroup	
Group Priv	Admin	Admin	<p>Default group privileges that are inherited upon group creation</p> <p>the highest level administrative user of the system</p> <p>all people in this group have full system admin privileges</p> <p>Entity (typically group or person) may modify the membership of this group, delete the group or assign privileges for the group</p> <p>Any entity (typically group or person) that is a part of this group</p> <p>Entity (typically group or person) may choose to join this group</p> <p>Entity (typically group or person) may choose to leave this group</p> <p>Entity (typically group or person) may see the membership list for this group</p> <p>Entity (typically group or person) may modify the membership of this group</p> <p>Entity (typically group or person) may see that this group exists</p>
	Member	Member	
	Optin	Optin	
	Optout	Optout	
	Read	Read	
	Update	Update	
	View	View	

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</i>, <i>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</i>, <i>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 Memberships 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 U 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 # 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	An entity that is listed in the Membership list of a Group has a direct membership in the group. Also see Indirect Membership .
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 entity references, and an 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	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. ...see Examples below.
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	There are two distinct uses for this term in Grouper. <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
ID	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
<i>name</i>	Biological Sciences Division
<i>ID path</i>	uofc:bsd
<i>path</i>	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
<i>ID</i>	eis_staff
<i>name</i>	Enterprise Information Systems staff
<i>ID path</i>	uofc:bsd:eis_staff
<i>path</i>	The University Of Chicago:Biological Sciences Division:Enterprise Information Systems staff

 Questions or comments?  Contact us.

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

Grouper Binary Release

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

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

Grouper - Loader

This page last changed on Dec 09, 2008 by [mchzyer](#).

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 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.

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).

INTERNET®

Groupe

Welcome MICHAEL CHRISTOPHER HYZER Log out Act as self Change

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

Name: aGroup2
ID: aGroup2
Description: some description

Assign privileges to everyone
☐ admin
☐ update
☒ read
☒ view
☐ optin
☐ optout

Select group types
☐ dynamic
☒ grouperLoader

Save
Back to group summary

Grouper is sponsored by

enn
CITY OF PENNSYLVANIA

Gr

Welcome Hyzer, Chris : mchyzer, Staff, ISC Administrative Systems Tools and Technologies, PROGRAMMER ANALYST SR Log out Act as admin

EXPLORE
Edit attributes

Current location is:
Root: a stem: aGroup2

Group type	Attribute	Value
grouperLoader	grouperLoaderDbName	grouper
	grouperLoaderIntervalSeconds	30
	grouperLoaderPriority	
	grouperLoaderQuartzCron	
	grouperLoaderQuery	select SUBJECT_ID from agroup2_v
	grouperLoaderScheduleType	START_TO_START_INTERVAL
	grouperLoaderType	SOL_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.

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 doesn't 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 isn't 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

Comments

Below I'll contrast this approach with the one I've been using - and modifications I would expect to make going forward. This is not to say my approach is better or would be more generally applicable:

Custom attributes

Custom attributes are expensive. I have used a few for loading specifically and lots more for course codes etc. Going forward, where practical, it might make sense to either use one or two attributes which encode multiple values - the fewer rows returned the better.

Alternatively, having a single marker attribute and using a separate table to hold additional configuration may make sense. This would assume you wouldn't generally search on this additional configuration, but a loader would be able to retrieve it. Looking at the example it seems likely that many groups would share this information i.e. loader maintained groups might fall into a 'small' number of categories i.e. student records, personnel ..., so you could load the config info once for each category.

Number of queries

For each category of groups I have a single query which returns `subject_id` and `group_name` for all groups. I build a nested Map of these results. I also have one query which retrieves all (non-group) immediate memberships for all groups. I also turn this into a nested Map and compute the differences between current and desired memberships so I only add / remove members as needed.

Going forward, rather than query all immediate memberships I would aim to get all memberships for a given category.

Currently I query the Grouper database directly, however, it would be possible to extend the API to do these types of queries.

I don't use transactions currently but would expect to keep all changes to a single group in a transaction.

Creating groups and assigning privileges

My loader comes in two parts - the membership loader described above and one which creates/updates stems/groups along with loader maintained types / attributes and also lists of privileged subjects - generally groups themselves. I use some custom attributes and a custom `UIGroupPrivilegeResolver` implementation so that the UI only allows users (including `GrouperSystem`) to maintain allowed attributes i.e. ones which would be overwritten with the next scheduled load.

Due to the lack of transactions I created Pseudo stems/groups and members so I could attempt to assemble parallel objects - detecting any errors before invoking API 'write' methods. I drive the loading with parameterised queries which will provide a row for each group of a category, or an individual group or a range of groups. For each row I build the pseudo objects (custom coded for each category type) and 'write' them.

I chose this approach because if there is an error part way through it is possible to start again where the error occurred, or even to run an individual group and debug a problem. Previously I had used a recursive approach which would start building faculties, departments and courses. It was difficult to pick up from where you left off if you got an out of memory error.

Going forward I don't think it will be necessary to use pseudo objects as I should be able to wrap write operations with transactions. In addition I would like to come up with a more generic means of defining what work needs to be done for each category of groups i.e. for each course I make admin / staff / student groups and set up privileges and attributes. I would like to have an XML configuration which acts as a template and drives the process.

Scheduling

I don't at the moment - and it is some time since I did a load, so using Quartz might help.

Posted by gary.brown@bristol.ac.uk at Apr 28, 2008.

GrouperShell (gsh)

This page last changed on Nov 14, 2008 by [mchyzer](#).

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

GrouperShell (gsh)

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 compatible with Grouper v1.2.0 or later. Version 0.1.1 was moved from <http://code.google.com/p/blair/wiki/gsh> and will become version 0.2.0 in the I2MI CVS repository.

Build

```
cd $GROUPER_HOME/ext
unzip gsh.zip
cd ..
ant build
ant dist
```

This will install gsh in \$GROUPER_HOME/ext/bin/gsh.sh

Source

CVS source code is available via

```
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi login
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co grouper-ext/gsh
```

Usage

Run gsh as an interactive shell:

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

Read gsh commands from STDIN:

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

Read gsh commands from a script file:

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

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
<code>addGroup(parent stem name, extension, displayExtension)</code>	Add group to registry
<code>delGroup(name)</code>	Delete group from registry
<code>getGroupAttr(group name, attr)</code>	Get value of group attribute
<code>getGroups(name)</code>	Find all groups with a matching naming attribute value, returns a Set of groups
<code>setGroupAttr(group name, attr, value)</code>	Set value of group attribute
<code>GroupFinder.findByName(grouperSession, name)</code>	Find one group by name
<code>GroupFinder.findById(grouperSession, name)</code>	Find one group by uuid
<code>group.manageIncludesExcludes(grouperSession, isIncludeExclude, calculate a group to have the other</code>	Understand how to calculate a group to have the other
<code>group.manageIncludesExcludes(grouperSession, isIncludeExclude, group that Members Must Also Be In)</code>	Understand how to calculate a group that Members Must Also Be In
<code>group.manageIncludesExcludes(grouperSession, isIncludeExclude, setOfGroupsThatMembersMustAlsoBeIn)</code>	Understand how to calculate a group that Members Must Also Be In

Group Types

Command	Description
<code>groupAddType(group name, type name)</code>	Add type to group
<code>groupDelType(group name, type name)</code>	Delete type from group
<code>groupGetTypes(group name)</code>	Get group's types
<code>groupHasType(group name, type name)</code>	Check whether group had type
<code>typeAdd(type name)</code>	Create custom group type
<code>typeAddAttr(type name, attr name, read, write, required)</code>	Create custom group attribute. <i>read</i> and <i>write</i> must be an <code>AccessPrivilege</code> (e.g. <code>AccessPrivilege.ADMIN</code>)
<code>typeAddList(type name, attr name, read, write)</code>	Create a custom list. <i>read</i> and <i>write</i> must be an <code>AccessPrivilege</code> (e.g. <code>AccessPrivilege.ADMIN</code>).
<code>typeDel(type name)</code>	Delete group type
<code>typeDelField(type name, field name)</code>	Delete custom field from group type
<code>typeFind(type name)</code>	Find the group
<code>typeGetFields(type name)</code>	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
<code>member.changeSubject(newSubject);</code>	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
<code>member.changeSubject(newSubject,! Member.DELETE_OLD_MEMBER);</code>	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
<code>member.changeSubjectReport(newSubject,Member.DELETE_OLD_MEMBER);</code>	Do this if the way which deletes the old member doesnt 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. To add a group to another group, use the group uuid to add as the subject id, e.g. addMember(groupNameToAddTo, groupUuidToAdd)
addMember(group name, subject id, field)	Add member to the specified list for the group. To add a group to another group, use the group uuid to add as the subject id, e.g. addMember(groupNameToAddTo, groupUuidToAdd)
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. AccessPrivilege.ADMIN)
grantPriv(stem name, subject id, privilege)	Grant privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. NamingPrivilege.STEM)
hasPriv(group name, subject id, privilege)	Check whether subject has privilege on group. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. AccessPrivilege.ADMIN)
hasPriv(stem name, subject id, privilege)	Check whether subject has privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. NamingPrivilege.STEM)
revokePriv(group name, subject id, privilege)	Revoke privilege on group. <i>privilege</i> must be an <i>AccessPrivilege</i> (e.g. AccessPrivilege.ADMIN)
revokePriv(stem name, subject id, privilege)	Revoke privilege on stem. <i>privilege</i> must be a <i>NamingPrivilege</i> (e.g. NamingPrivilege.STEM)

Registry

Note: gsh -registry is probably better than any of these commands

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

registryInstall()

If the default Grouper data is not there, it will be added (e.g. root stem, default fields, etc). Note: gsh -registry is probably better

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

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
GrouperReport.report(isRunUnresolvableSubjectReport, isRunBadMembershipFinder)	Run the Grouper "daily report" which gives some stats about recent activity, shows the loader progress and errors, runs unresolvable subject utility (optional), and show unresolvable subjects if any, run the find bad membership finder (optional) and show bad memberships if any

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.

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 membeships
subject=SubjectFinder.findById("GrouperSystem")	Work only with a specific member
session=GrouperSession.start(subject)	
memberSubject=SubjectFinder.findById("1234567")	
member=MemberFinder.findBySubject(session,memberSubject)	
usduByMember(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

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

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.userProperties(file)	Properties file for extra settings for import
XmlExport	Operate within a certain grouper session (defaults to root session)
xmlExport.grouperSession(grouperSession)	
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.properties")).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.

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)
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.properties")).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

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-84a7-7af609flad09'
```

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

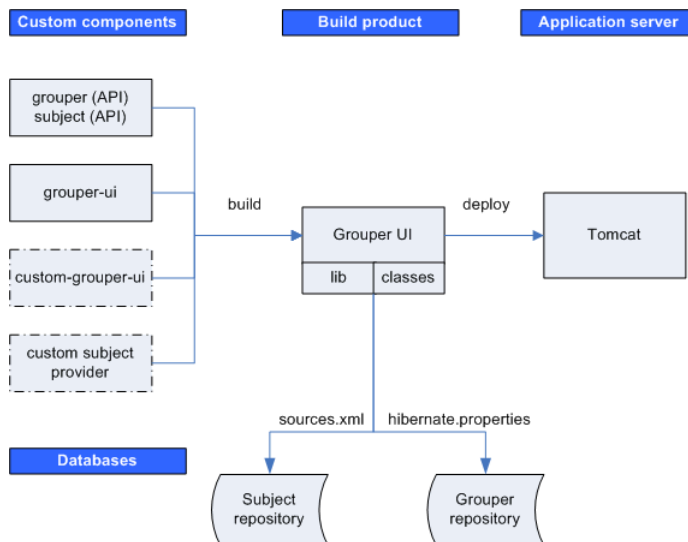
Grouper UI Components

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

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

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 (*optional*)

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 (*optional*)

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: About FAQ Software Documentation Archives Contribute WG Contact

Grouper UI Development Environment

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

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

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: About FAQ Software Documentation Archives Contribute WG Contact


Grouper Use Cases

This page last changed on Jul 18, 2007 by jbibbee@internet2.edu.

GROUPER: About FAQ Software Documentation Contribute WG Contact
--

This document will describe how Grouper addresses many of the current challenges in managing groups.

... more to come soon!

 Questions or comments?  Contact us.

GROUPER: About FAQ Software Documentation Contribute WG Contact
--

Grouper Web Services for v1.3.0

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](#).

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](#)

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)
- [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)
- View Or Edit Privileges: does not exist for v1.3. Does exist for 1.4
- [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)

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
\webservices\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 1.3.0)

1. investigate backwards compatibility with Axis... discuss options
2. persist group detail when saving groups (including attributes)
3. add find subject service
4. test more (unit test)
5. make some params to test stuff... (junit to throw exceptions in the middle of tx?)
6. come up with formatter and code style and remove all warnings
7. add logging filter
8. fix javadoc warnings
9. look into axis upgrade when param order error is fixed, see about enums
10. add metadata service
11. add getGroups with batched groupLookup input
12. add back in privileges service
13. add back in memberships service
14. privileges: Input param replaceAllExisting?
15. filter getMember by privileges (find member?)
16. in rest add GET starting points with links to resources
17. improve auto-toString methods in resultMessage
18. look at acegi
19. add ip source filtering to grouper

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

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.ramport
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

XML Import / Export for Grouper v1.2.0 - v1.3.0

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.*
- 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 args:	Prints this message

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. 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 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 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 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 args:	Prints this message 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. 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.

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

Initializing Administration of Privileges

This page last changed on May 21, 2008 by tbarton@uchicago.edu.

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

Initializing Administration of Grouper Privileges as of v1.3.0

Prior to v1.0, Grouper required on-going, though perhaps occasional, use of a root-like principal called GrouperSystem to manage the assignment of privileges in Grouper. As of v1.0, it is possible to configure a [wheel group](#) of externally authenticated subjects who can choose when to act with the privilege of GrouperSystem. Hence, it is necessary to use the GrouperSystem account only once, during installation, to bootstrap the designation of these individuals.

Using GrouperShell to Bootstrap the Wheel Group


If you've enabled the [wheel group](#), you must create the group named within the groups.wheel.group property in the grouper/conf/grouper.properties configuration file, and add some members to that group. Since GrouperShell acts as GrouperSystem, it can be used to create the necessary naming stem(s), group, and memberships.

To do so, build the gsh utility per the instructions in the [GrouperShell](#) documentation, then issue a series of gsh commands along the following lines. This particular sequence creates the group 'etc:sysadmin' and adds one member to it.

```
% cd GROUPER_HOME
% ext/bin/gsh.sh
gsh-0.1.1 0% addRootStem("etc", "Grouper Administration")
stem: name='etc' displayName='Grouper Administration' uuid='f7687876-2c94-4635-997c-f2793fb8152d'
gsh-0.1.1 1% addGroup("etc", "sysadmin", "SysAdmin Group")
group: name='etc:sysadmin' displayName='Grouper Administration:SysAdmin Group' uuid='6f77fb36-b466-481a-84a7-7af609flad09'
gsh-0.1.1 2% addMember("etc:sysadmin", "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: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [Contribute](#) [WG](#) [Contact](#)

Intro FAQ

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

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

Intro FAQ about Grouper

Grouper is [Licensed](#) under the Apache 2.0 license. For technical FAQ, see the [Technical FAQ](#).

1. [What is Grouper?](#)
2. [Who needs Grouper?](#)
3. [How can Grouper help my institution?](#)
4. [How does Grouper differ from other solutions?](#)
5. [What do I need to use Grouper?](#)
6. [Does Grouper integrate well with others?](#)

1. What is Grouper?

...

2. Who needs Grouper?

...

3. How can Grouper help my institution?

...

4. How does Grouper differ from other solutions?



...

5. Who developed Grouper?

...

6. Does Grouper integrate well with others?

...

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


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

License

This page last changed on Jul 18, 2007 by jbibbee@internet2.edu.

GROUPER: About FAQ Software Documentation Contribute WG Contact
--

Note: Information regarding the **Grouper license** may be found here: <http://middleware.internet2.edu/dir/groups/grouper/license.html>

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

Media Properties

This page last changed on May 21, 2008 by tbarton@uchicago.edu.

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

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: About FAQ Software Documentation Archives Contribute WG Contact

Nav

This page last changed on Aug 30, 2007 by jbibbee@internet2.edu.

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

Prerequisites

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

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

Prerequisites as of v1.3.1

The essential prerequisite steps are:

1. [Install & Configure the Prerequisite Infrastructure](#),
2. [Establish a Database for Grouper](#), and
3. [Download the Grouper v1.3.1 Distribution](#).

The [Project layout](#) of the downloaded package is also described.

Install & Configure the Prerequisite Infrastructure

[Ant](#) - You will need ant v1.6.3 or later to build Grouper v1.3.0. 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.

[Java & Servlet Container](#)- You will need java v5.0.6+ and Tomcat v5.5+ to build and run Grouper v1.3.0. For Grouper v1.2.1 and earlier, you may use java v1.4.2+ to build & run Grouper. Java & Tomcat versions must be chosen to work together. Choose either:

- Java 5.0.6+ and Tomcat 5.5+ (**recommended**), OR
- Java 1.4.2 and Tomcat 4.1 or 5.0

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.

Establish a Database for Grouper

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. DDL for the Grouper schema is generated dynamically by Hibernate, according to the database configured in its properties file.

Download the Grouper Distribution

The [Download](#) page contains instructions for downloading the API and UI tarballs.

Note: The Grouper Quickstart distribution (also referenced there) is an integrated package with self-contained instructions, intended solely for getting a demo up and running quickly. This wiki page is concerned with installing the API and UI tarballs.

Project Layout

The Grouper API tarball and the Grouper UI tarball should be expanded in the same parent directory so that various automated installation tasks can succeed. The top-level 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
build/	Compiled code
contrib/	Contributed software
doc/	Grouper API documentation
lib/	Third party .jar files required by the Grouper API
libAnt/	Third party .jar files required by the Ant compiler to build Grouper
src/	Java source for Grouper API and API test suite
sql/	SQL scripts for creating, initializing and testing the Groups Registry
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.

Note: This layout assumes that the API and UI packages are unpacked in the same parent directory. The UI build process will attempt to create another directory peer to these. Hence, the parent directory must be writable.

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

Resources

This page last changed on Jan 04, 2008 by jbibbee@internet2.edu.

Resources - A Look at Ourselves

[CAMP: Building a Distributed Access Management Infrastructure](#) November 7-9, 2006

Use this parent page to add your scores for the self-assessment tool, [A Look at Ourselves](#).

1. **Add Page** > Select a page template to start from.
2. Select the "A Look at Ourselves" template > Create page from template.
3. Title the page as you wish, or leave your institution's name if privacy is not a concern.
4. Any results shared here will be compiled for your convenience!

Software Download

This page last changed on Dec 15, 2008 by tzeller@memphis.edu.

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

Download Grouper

Grouper v1.4.0 Release Candidate 1

Software release: [22-November-2008]

Release component	Description
Grouper QuickStart v1.4.0 RC1	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 v1.4.0 RC1	Binary release of the Grouper Application Program Interface and associated utilities.
Grouper API Source v1.4.0 RC1	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.4.0 RC1	Contains the full source for the Grouper User Interface.
Grouper WS v1.4.0 RC1	Contains the full source for the Grouper Web Services Interface.
Grouper Client v1.4.0 RC1	Experimental client for Grouper LDAP and Web Services.
Contributed Software	Provided by the Grouper community, in addition to the above core Grouper products.

[v1.4.0 Release Notes](#) - Highlights of the v1.4.0 release, upgrade instructions, and detailed log of changes.

[Documentation for this release](#) is in a set of wiki pages parallel to those for the latest stable release.

Grouper v1.3.1

Software release: [22-September-2008]

This is the most recent stable version.

Release component	Description
Grouper QuickStart v1.3.1 tarball	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 v1.3.1 tarball	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.3.1 tarball	Contains the full source for the Grouper User Interface.
Grouper WS v1.3.1 tarball	Contains the full source for the Grouper Web Services Interface.
Contributed Software	Provided by the Grouper community, in addition to the above core Grouper products.

[v1.3.1 Release Notes](#) - View the changes & fixes for the v1.3.1 release.

Older versions

[Archives](#) - For a complete listing of feature updates and changes to Grouper across previous releases, including archived documentation.

For an explanation of Grouper specific terms and definitions, see the [Glossary](#). For detailed information regarding the deployment of Grouper, refer to the System Administration section of the main [Grouper Product](#) Documentation page.

If You're Interested...

[Licensed](#) under the Apache 2.0 license.

[Specsheet](#) offers operational specifications for the development and deployment of Grouper.

[Grouper Working Group](#) information relating to the Grouper project and software development can be found here. Or go to the [GrouperWG](#) wiki.

[CVS](#) - 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_3_1 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_UI_1_3_1 grouper-ui
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_WS_1_3_1 grouper-ws
```

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

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

Contact

 If you have a question about the Grouper software,  [Contact us](#).

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

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.

This page last changed on Apr 16, 2008 by [shilen](#).

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:

-
- The diagram illustrates the database schema for the 'grouper' database, showing the relationships between various tables and their attributes.
- Tables and their attributes:**
- grouper_composites**
 - id (varchar(128))
 - uuid (varchar(128))
 - owner (varchar(128))
 - left_factor (varchar(128))
 - right_factor (varchar(128))
 - type (varchar(50))
 - creator_id (varchar(128))
 - create_time (bigint)
 - grouper_groups_types**
 - id (varchar(128))
 - group_uuid (varchar(128))
 - type_uuid (varchar(128))
 - grouper_attributes**
 - id (varchar(128))
 - group_id (varchar(128))
 - field_name (varchar(32))
 - value (text(5535))
 - grouper_memberships**
 - owner_id (varchar(128))
 - member_id (varchar(128))
 - list_name (varchar(32))
 - list_type (varchar(32))
 - mship_type (varchar(32))
 - via_id (varchar(128))
 - depth (int)
 - parent_membership (varchar(128))
 - membership_uuid (varchar(128))
 - creator_id (varchar(128))
 - create_time (bigint)
 - grouper_stems**
 - id (varchar(128))
 - uuid (varchar(128))
 - parent_stem (varchar(128))
 - name (varchar(255))
 - display_name (varchar(255))
 - creator_id (varchar(128))
 - create_time (bigint)
 - modifier_id (varchar(128))
 - modify_time (bigint)
 - display_extension (varchar(255))
 - extension (varchar(255))
 - description (text(5535))
 - create_source (varchar(255))
 - modify_source (varchar(255))
 - grouper_groups**
 - id (varchar(128))
 - uuid (varchar(128))
 - parent_stem (varchar(128))
 - creator_id (varchar(128))
 - create_time (bigint)
 - modifier_id (varchar(128))
 - modify_time (bigint)
 - create_source (varchar(255))
 - modify_source (varchar(255))
 - grouper_types**
 - id (varchar(128))
 - type_uuid (varchar(128))
 - name (varchar(255))
 - creator_uuid (varchar(128))
 - create_time (bigint)
 - is_assignable (bit(1))
 - is_internal (bit(1))
 - grouper_sessions**
 - id (varchar(128))
 - member_id (varchar(128))
 - start_time (bigint)
 - session_uuid (varchar(128))
 - grouper_members**
 - id (varchar(128))
 - member_uuid (varchar(128))
 - subject_id (varchar(255))
 - subject_source (varchar(255))
 - subject_type (varchar(255))
 - subjectAttribute**
 - subject_id (varchar(255))
 - attribute_name (varchar(255))
 - value (text(5535))
 - create_time (bigint)
 - Subject**
 - subjectid (varchar(255))
 - subjecttypeid (varchar(32))
 - name (varchar(255))
- Relationships:**
- grouper_composites** (1) to **grouper_groups** (many)
 - grouper_groups** (1) to **grouper_stems** (many)
 - grouper_stems** (1) to **grouper_groups** (many)
 - grouper_stems** (1) to **grouper_types** (many)
 - grouper_stems** (1) to **grouper_sessions** (many)
 - grouper_stems** (1) to **grouper_members** (many)
 - grouper_groups** (1) to **grouper_attributes** (many)
 - grouper_groups** (1) to **grouper_memberships** (many)
 - grouper_types** (1) to **grouper_memberships** (many)
 - grouper_memberships** (1) to **grouper_members** (many)
 - grouper_memberships** (1) to **subjectAttribute** (many)
 - subjectAttribute** (1) to **Subject** (many)

privileges.

Grouper change log v1.3

This page last changed on Sep 03, 2008 by [mchyzer](#).

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.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

v1.3-Release Notes

This page last changed on Oct 24, 2008 by tbarton@uchicago.edu.

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

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>Release v1.3.1 includes the following fixes and improvements:</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>Release v1.3.0 includes the following fixes and improvements:</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.	22-May-2008

- 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.
- 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 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.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: About FAQ Software Documentation Archives Contribute WG Contact

Specsheet

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

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

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.
Java	Contributions: Yale CAS authentication filter. JDK v5.0.6 or later.
Compiler	Ant v1.6.3 or later.
Browser Requirements	<ul style="list-style-type: none">XHTML 1.0CSS 2.1cookies must be enabledjavascript 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.
Java	Contributions: Yale CAS authentication filter. JDK v5.0.6 or later.
Compiler	Ant v1.6 or later.
Browser Requirements	<ul style="list-style-type: none">XHTML 1.0CSS 2.1cookies must be enabledno 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: About FAQ Software Documentation Archives Contribute WG Contact

Technical FAQ

This page last changed on Aug 07, 2008 by khuxtable@ku.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?](#)

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 Ldapdc 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> 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.

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

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

UI Building and Configuration

This page last changed on Dec 05, 2007 by jbibbee@internet2.edu.

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

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

UI Customization Guide

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

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

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

Document	Description
Grouper UI Components Architecture	An overview of Grouper UI components. 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 May 09, 2008 by tzeller@memphis.edu.

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

Unresolvable Subject Deletion Utility (USDU)

This document is released alongside Grouper v1.3.0.

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.

Installation

Usdu is provided as an extension. It is installed by running 'ant build' and 'ant dist' after it is unpacked in the extensions directory \$GROUPER_HOME/ext.

```
cd $GROUPER_HOME/ext
unzip usdu.zip
cd ..
ant build
ant dist
```

CVS source code is available via

```
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi login
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co grouper-ext/usdu
```

Usage

Without any arguments, usdu prints its usage

```
$ ext/bin/usdu.sh
```

```
$ ext/bin/usdu.bat
```

```
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

```
$ ext/bin/usdu.sh -all
member_uuid='e58697e4-...' subject='idl'/'source'/'person' group='stem:group1' list='members'
member_uuid='e58697e4-...' subject='idl2'/'source'/'person' group='stem:group2' list='members'
...
```

Find unresolvable subjects from a specified source

```
$ ext/bin/usdu.sh -source CustomSource
```

Find unresolvable subject via member uuid

```
$ ext/bin/usdu.sh -uuid e58697e4-11a5-4082-b318-cb1e79191923
```

Delete unresolvable subject from all groups

```
$ ext/bin/usdu.sh -uuid e58697e4-... -delete
member_uuid='e58697e4-...' subject='idl'/'source'/'person' group='stem:group1' list='members'
delete
```

Details

Kinds of unresolvable members

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.

Launching

By default, usdu is run using [invoker.jar](#), which should make customizing runtime parameters (such as the java classpath) easier across operating systems. A shell script that does not use `invoker.jar` may be more convenient for some system administrators and is provided at `$GROUPER_HOME/ext/bin/usdu.sh.noinvoker`.

 Questions or comments?  Contact us.

GROUPER: About FAQ Software Documentation Archives Contribute WG Contact

Add Member

This page last changed on Nov 10, 2008 by [mchyzer](#).

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 27, 2008 by [mchzyer](#).

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 httpRequest.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: httpRequest.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 Mar 31, 2008 by [mchzyer](#).

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 Mar 31, 2008 by [mchyzer](#).

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 Mar 31, 2008 by [mchzyer](#).

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 27, 2008 by [mchzyer](#).

Description

"[Get grouper privileges](#)" will retrieve the privileges for a subject and (group or stem). If you dont specify the privilege name, you will get all permissions for the user and (group or stem). If you specify the privileges you are looking for, you will get also get the response in the return code (which is an HTTP header)

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)

Get Groups

This page last changed on Mar 31, 2008 by [mchyzer](#).

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 Mar 31, 2008 by [mchzyer](#).

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 [mchzyer](#).

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 mchzyer
 */
```

```

*/
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: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>bla6fe70-9f66-4b7d-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 mchyzer
 *
 */
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: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>bla6fe70-9f66-4b7d-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 Mar 31, 2008 by [mchyzer](#).

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(BeaParent[] _beanParents) {
        this.beaParents = _beanParents;
    }
}

/**
 * @author mchzyer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;

/**
 * parent bean
 */
public class BeaParent {
    /** field */
    private String parentField1;

    /** field */
    private String parentField2;

    /** field */
    private String[] parentStringArray;

    /** field */
    private int intField;

    /** ean child */
    private BeaChild beaChild;

    /** child */
    private BeaChild beaChild2;

    /** child */
    private BeaChild[] beaArray;

    /** child */
    private BeaChild[] beaArray2;

    /**
     * @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>
    </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>
</beanGrandparent>
<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":{

```

Warnings example

```
<div title=" BeanChild " id=" something">
  <span>something</span>
  <p class="childField1">va<lt;ll</p>
  <p class="childField2">va2</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 Nov 06, 2008 by [mchyzer](#).

Description

Group save will insert or update a group's uuid, extension, display name, or description (with restrictions)

Features

- Can pass SaveMode which is INSERT, UPDATE, or INSERT_OR_UPDATE (default)
- If the stem doesn't 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)

Has Member

This page last changed on Mar 31, 2008 by [mchzyer](#).

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 22, 2008 by [mchzyer](#).

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 Mar 31, 2008 by [mchyszer](#).

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 Mar 31, 2008 by [mchyzer](#).

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