

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
 - Architecture
 - Contact Information
 - Custom Group Types, Fields, Attributes, Lists
 - Customising the Grouper UI
 - Developer's Guide to the Grouper API
 - Glossary
 - UI Terminology
 - Grouper Design Guidelines
 - Grouper UI Components
 - Grouper UI Development Environment
 - Grouper Use Cases
 - Grouper Web Services
 - Authentication for Grouper Web Services
 - Grouper Web Services FAQ
 - GrouperShell (gsh)
 - Groups Management & Your Institution
 - Import-Export
 - Initializing Administration of Privileges
 - Intro FAQ
 - License
 - Media Properties
 - Nav
 - Prerequisites
 - Resources
 - Software Download
 - Database Conversion v1.0 - v1.1

- Database Conversion v1.2.0 - v1.2.1
- Grouper change log
- Release Notes
- Specsheet
- Supporting Your Campus
- Technical FAQ
- UI Building and Configuration
- UI Customization Guide
- Unresolvable Subject Deletion Utility (USDU)

Home

This page last changed on May 05, 2008 by tbarton@uchicago.edu.

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

Welcome to the Grouper Group Management Toolkit Wiki

Grouper Product	Grouper Project
<i>Open viewing. Editing restricted to the Grouper Developers.</i>	<i>Open viewing. Editing restricted to Grouper Working Group members.</i>
Grouper Product Documentation The Grouper Product space, with the latest software and documentation. <ul style="list-style-type: none">• New! - *{ }Grouper v1.3.0 RC2{ } 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 (PDF) 	The Grouper Working Group The Grouper Project space, with the design and development work of the MACE-led Grouper Working Group. <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:

- a common user interface and standard API for managing groups
- 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

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 the namespace
- per-group opt-in or opt-out entitlement

In addition to basic group management and search capabilities, Grouper's v1.2.1 design includes support for: substantially improved performance, API and UI use new strategies to check privileges, and improved API caching strategy.

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](#).

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 Apr 29, 2008 by tzeller@memphis.edu.

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

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

Welcome to the Grouper v1.2.1 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.

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.
- [UI Tutorial](#) - A little help in understanding the UI capabilities.

Applications Development

[Developer's Guide to the Grouper API](#) - Documentation on how to use the Grouper API in other java applications.

[*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](#)

[*API v1.2.1 Javadoc*](#)

[*UI v1.2.1 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_2_1 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_UI_1_2_1
grouper-ui
```

[*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.

API Building and Testing

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

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

Step-by-Step Instructions to Build and Test the Grouper API as of v1.2.0

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 default forms 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](#).

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

API Configuration

This page last changed on Apr 18, 2008 by tbarton@uchicago.edu.

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

Configuring the Grouper API as of v1.2.1

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

Section	Configuration File	Purpose
Database-Related Settings and Procedures	grouper.hibernate.properties	integrating the Grouper API with the database that will house your Groups Registry
Configuration of Source Adapters	sources.xml	integrating the Grouper API with chosen identity sources
Grouper Privileges	grouper.properties	defaults for Grouper privileges, enabling identified external users to act with elevated root-like privilege
Logging	log4j.properties, grouper.properties	logging

Database-Related Settings and Procedures

Grouper uses Hibernate to persist objects in the Groups Registry, so all of the database-specific settings are located in `grouper/conf/grouper.hibernate.properties`. After modifying the default properties as needed, Grouper API ant tasks detailed below are used to create and install the Groups Registry schema and initialize the database.

Database Driver Location

Place the jar file containing the JDBC driver for your database in the `grouper/lib/` folder. The Grouper v1.2.1 package includes the JDBC driver for HSQLDB v1.7.2.11 in this folder. You may need to 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
```

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 v1.2.1 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 v1.2.1 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 v1.2.1 Distribution
<code>groups.create.grant.all.admin</code>	false
<code>groups.create.grant.all.optin</code>	false
<code>groups.create.grant.all.optout</code>	false
<code>groups.create.grant.all.update</code>	false
<code>groups.create.grant.all.read</code>	true
<code>groups.create.grant.all.view</code>	true
<code>stems.create.grant.all.create</code>	false
<code>stems.create.grant.all.stem</code>	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
<code>groups.wheel.use</code>	"true" or "false" to enable or disable this capability.

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

Note that, as of v1.0, the Grouper UI enables users that belong to the wheel group to choose when to act with the privileges of GrouperSystem and when to act as their normal selves.

Changing default privilege and subject 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.SimpleWheelPrivilegeCache`.

Grouper also includes two SubjectCache implementations:

- NoCacheSubjectCache - No caching performed
- SimpleSubjectCache - Simple caching is performed

The `subjects.cache.id.interface` and `subjects.cache.identifier.interface` properties can be set to determine the subject caching regimen. The default is `edu.internet2.middleware.grouper.SimpleSubjectCache`.

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
<code>privileges.access.interface</code>	classname of the java class that implements the Access Interface
<code>privileges.naming.interface</code>	classname of the java class that implements the Naming Interface

Note: although we've provided the can and the dish, we haven't as yet eaten our own dogfood!

Logging

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

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

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

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

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

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

Architecture

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

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

Grouper UI Architecture

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

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

1. Introduction

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

1. **Core technology**
Java-based web application to complement the Grouper API which is also Java-based
2. **Web application framework**
Use of established and well-documented framework not tied to particular platform
3. **Internationalization**
Designed to facilitate sites who wish to use a language other than english and in such a way that an institution can offer more than one language

4. **Extensibility**

Institutions will likely want to tailor the UI to reflect local business rules and available meta data

5. **Separation of core Grouper UI code from site-specific code**

Separation is important to ensure that local changes do not require modification of Grouper sources thus making upgrades much easier

6. **Accessibility**

Content must be accessible through use of DOM /CSS layouts rather than table layouts

2. The Architecture and How It Matches the Requirements

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

2.1. Core Technology

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

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

2.2. Framework

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

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

2.3. Internationalization

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

2.4. Extensibility

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

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

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

The same is true of Tiles definition descriptors.

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

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

2.6. Accessibility

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

3. Implementation Details

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

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

3.1. Grouper UI Out-of-the-Box

grouper	The web application document root.
grouper	Grouper specific stylesheets
images	Grouper specific images

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

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

```
<!-- [Redacted code] -->
```

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

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

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

```
<!-- [Redacted code] -->
```

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

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

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

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

The file contents would be something like:

```
<!-- [Redacted code] -->
```

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

```
<!-- [Redacted code] -->
```

or

```
<!-- [Redacted code] -->
```

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

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

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

GroupsManager	The web application document root. Note that we can name the web application as we see fit
grouper	Grouper specific stylesheets / JavaScript
images	Grouper specific images
i2mi	Internet 2 specific stylesheets / JavaScript
images	Internet 2 specific images
uob	UoB specific stylesheets / JavaScript
images	UoB specific images
WEB-INF	The Struts ActionServlet configuration must be modified.
classes	Additional resources / Java classes required by UoB copied here
jsp	Additional JSP files placed here - probably in a UoB subdirectory
lib	Any additional JAR files required by UoB code placed here

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



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

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

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



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

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

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

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

*see [3.4](#) for a discussion of the alternative approaches to overloading properties for locales and application components.

3.3. Supporting an Additional Language

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

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

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

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

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

3.4. Overloading in ResourceBundles

3.4.1. How it Works

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

1. Language code
2. Country code
3. Variant

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

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

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

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

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

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

3.4.2. Multiple UIs One Instance

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

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

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

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

- nav_en_GB
- nav_en_GB_student
- nav_en_GB_staff

- nav_en_GB_portal

By choosing the correct Locale appropriate text can be presented.

In the consortium example:

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

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

3.4.3. Chaining ResourceBundles

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

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

- nav -> audit trail
- nav_es->sendero auditoría
- nav_uob_en_GB->log
- nav_uob_es ->
{no key/value}

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

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

3.5. Overloading Struts Config Elements

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

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

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

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

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

3.6. Additional Hooks to Extend the Grouper UI

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

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

3.7. Page Composition Using Tiles

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



Such a page can be represented using a Tiles definition:

```
xxxDef {  
  init {  
    /xxx/init.jsp  
  }  
  head {  
    /xxx/head.jsp  
  }  
  header {  
    /xxx/header.jsp  
  }  
  subheader {  
    /xxx/subheader.jsp  
  }  
  message {  
    /xxx/message.jsp  
  }  
  content {  
    /xxx/content.jsp  
  }  
  footer {  
    /xxx/footer.jsp  
  }  
}
```

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

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

[Redacted]

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

[Redacted]

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

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

[Redacted]

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

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

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

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

3.8. Content Composition Using Tiles

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

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

MANAGE GROUPS - BROWSE GROUPS HIERARCHY
Groups I can manage

BROWSE GROUPS 0

GROUPS HOME- Bristol University 1

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

SEARCH GROUPS 5

MANAGE STEM 6

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

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

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

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

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

The tile would be inserted into the content, thus:

```
<include src="/WEB-INF/jsp/ManageGroups.jsp" type="jsp" />
```

or

```
<include src="/WEB-INF/jsp/ManageGroups.jsp" type="jsp" controllerUrl="/manageGroups" />
```

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

3.9. Dynamic Tiles

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

A generic dynamic tile has been defined thus:

```
<tile class="org.apache.grouper.ui.tiles.DynamicTile" />
```

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

dynamic tile:

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

3.10. JSP Tag Libraries

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

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

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

GROUPER: [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>](mailto:grouper-dev@internet2.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` 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 [mchalyzer](#).

GROUPER: [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.

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

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:

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:

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

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

Customising the Grouper UI

This page last changed on Apr 23, 2008 by gary.brown@bristol.ac.uk.

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 v1.2.1 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 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.html](#). 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 Tiles to define core page components. In the standard UI these are defined in **grouper-qs/grouper-ui/webapp/WEB-INF/tiles-def.xml**. In the custom UI some definitions are modified and another added in the file **grouper-qs/custom-grouper-ui/webapp/WEB-INF/tiles-def-custom.xml**. New definitions for *headerDef* and *footerDef* allow site specific branding. *groupStuffDef* defines a template which is included in the Group Summary page of the UI.

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

Defining Custom Dynamic Templates

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

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

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

In the custom UI:

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

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

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

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

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

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

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

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

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

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

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

Customising Authentication

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

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

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

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

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

Customising 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.browser.<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/CustomQSPr`

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,creat
#subject.attributes.order.qsuob=LOGINID,LFNAME,subjectType,id
```

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 <code>?key?</code>
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](#)

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	<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>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.</p> <ul style="list-style-type: none"> • name - used to facilitate searching for groups by name, it is a read-only string representation of the logical ordered pair of (<i>parent stem</i>, <i>extension</i>). This attribute is system-maintained. The string representation of the <i>name</i> attribute is: <i><parent stem>: <extension></i>. • displayExtension - a displayed form of the extension. • displayName- used to facilitate searching for groups by the displayed name, it is a read-only string representation of the logical ordered pair of (<i>displayName of parent stem</i>, <i>displayExtension</i>). This attribute is system-maintained. The string representation of the <i>displayName</i> attribute is: <i><displayName of parent stem>: <displayExtension></i>. • description - a description of the group or naming stem. 	
<p>Composite Group</p>	<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 	

	<p>Y, or $Z = X \cup Y$.</p> <ul style="list-style-type: none"> • intersection - all subjects that are members of the first factor group AND the second factor group, e.g., Group Z = members of both Group X AND Group Y, or $Z = X \cap Y$. • relative complement - all members of the first factor group that are NOT members of the second factor group. e.g., Group Z = members of Group X AND NOT Group Y, or $Z = X - Y$. 	
Direct Membership	A Subject that is listed in the Membership list of a Group has a direct membership in the group. Also see Indirect Membership .	Subject is a UI "entity"
Factor Group	A Group in combination (union, intersection, or relative complement) with that of another factor group, which defines the membership of a resulting Composite Group .	
Field	Either an Attribute or a List . Grouper groups are a collection of attributes and lists, i.e., a collection of fields. The set of fields attached to a given group is a function of the set of Group Types it has been assigned.	
Group	<p>A list of Subjects having Membership in the group, together with other attributes about the group. A list can have zero or more entries. In Grouper, a list contains only subject references, and an attribute is a single-valued string. A group must be created in an existing Naming Stem. If a group is made a member, i.e., a Subgroup, of another group, the members of the group will also be made members. By default, a Grouper group has:</p> <ul style="list-style-type: none"> • six naming Attributes, • a description attribute, 	naming stem is a UI "folder"

	<p>and</p> <ul style="list-style-type: none"> • a members list. <p>This information model can be extended to include additional site-defined attributes and lists.</p>	
Group Math	<p>Any combination of groups for the purpose of creating another group based on the memberships of those groups. See Composite Group.</p>	
Indirect Membership	<p>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.</p>	
List	<p>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.</p>	
Member	<p>Any Subject in the membership list of at least one group. Also, a Member of a Group is any Subject with a Direct or Indirect Membership in the Group.</p>	
Membership	<p>The direct-only, indirect-only, or direct plus indirect members of a Group. A specific variety of membership is determined by context or configuration, i.e., the default User Interface allows the user to select among these three types of membership where appropriate.</p>	
Naming Privileges	<p>These privileges determine what a Subject can do with a Naming Stem. They are:</p> <ul style="list-style-type: none"> • CREATE - can create a 	<p>Naming privileges are now referred to as Creation privileges and the two types are Create Group (replaces CREATE) and</p>

	<p>group(s) named with a naming stem, and</p> <ul style="list-style-type: none"> • STEM - can assign who has CREATE for the naming stem, and can create naming stems subordinate to this one. 	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.</p> <p>...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 	

	<p>groups with a richer information model or a more diverse set of information models.</p> <ul style="list-style-type: none"> • 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	Entities With Privileges	
	Subject	Entity	An entity is an abstract item which may be a member of a group. The two most common types of entities are 'person' or 'group'. (In the future, additional entity types may be used to describe computers or applications.)
	is a direct privilegee	has direct privileges	as a member of the group
	is a indirect privilegee	has indirect privileges	within a group that is a member of the group
	Extension	ID	An internal name describing this group that is generally not exposed to the user. This name cannot be changed after it is edited
	Name	ID Path	An internal concatenation of the hierarchy to this group that is generally not exposed to the user
	Display extension	Name	The group name that is displayed when browsing or searching
	Display name	Path	The path is the concatenation of the hierarchy (folders and groups) that lead to the unique location of this

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

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

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

TERM	DEFINITION
Access Privileges	<p>Privileges that determine what a Entity can do with a Group. They are:</p> <ul style="list-style-type: none"> • ADMIN - can assign access privileges and manage all group information, • UPDATE - can manage membership of the group (implies READ), • READ - can see the membership of the group (implies VIEW), and • VIEW - can see the group. <p>In addition, a group may have options for its members to:</p> <ul style="list-style-type: none"> • OPTIN - can add self to the membership,

	<p>and</p> <ul style="list-style-type: none"> • OPTOUT - can remove self from membership.
<p>Attribute</p>	<p>A single-valued string associated with a Group or a Folder. By default, Grouper supports six attributes (one of two kinds of Field):</p> <ul style="list-style-type: none"> • UUID - a Grouper-assigned, globally unique identifier. • ID- the relative name of the group or folder within its parent folder; the contribution of a single element, such as a group or a folder, to the cumulative name. • ID Path- used to facilitate searching for groups by name, it is a read-only string representation of the logical ordered pair of (<i>parent folder, ID</i>). This attribute is system-maintained. The string representation of the ID Path attribute is: <i><folder>:<ID></i>. • Name- a displayed form of the ID. • Path -used to facilitate searching for groups by the path, it is a read-only string representation of the logical ordered pair of (<i>Path of parent folder, Name</i>). This attribute is system-maintained. The string representation of the path attribute is: <i><Path of parent folder>:<name></i>. • description - a description of the group or folder.
<p>Composite Group</p>	<p>A Group whose Membership is determined by combining the membership lists of two other groups, without listing its members explicitly. These two groups are called its Factor Groups. Three methods of combining the factor groups' memberships are supported:</p> <ul style="list-style-type: none"> • union - all entities must be a member of one OR the other factor group, e.g., Group Z = members of either Group X OR Group Y, or Z = X 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.
<p>Direct Membership</p>	<p>An entity that is listed in the Membership list of a</p>

	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	<p>A string that precedes the Group's name. By linking the ability to create groups to a specified folder (via the Creation privilege), the possibility that different groups can be given the same name is substantially reduced, and the name of each group can be made to reflect something about the authority under which it was created. ...see Examples below.</p>
Subgroup	A Group that is a Direct Member of another group.
Entity	<p>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.</p>
Type	<p>There are two distinct uses for this term in Grouper.</p> <ul style="list-style-type: none"> • Group Type - each Group has one or more group types associated with it. The Grouper distribution contains support for a single group type called "base", but sites may register additional types, together with the attributes and lists associated with them, within their Grouper installation. Doing so enables management of groups with a richer information model or a more diverse set of information models. • Entity Type - the Subject API v0.2.1 that Grouper 1.3 relies on uses the notion of a subject type, such as "person", "group", or "computer", etc.

Examples

Step 1: Create a Root Folder

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

Folder uofc

<i>attribute</i>	<i>value</i>
<i>folder</i>	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

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

Step 3: Create a Subordinate Folder and Group

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

Folder uofc:bsd

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

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

Group uofc:bsd:eis_staff

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

 Questions or comments?  Contact us.

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

Grouper Design Guidelines

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

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

This document will offer several considerations as you prepare to deploy Grouper at your campus.

... more to come soon!

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

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

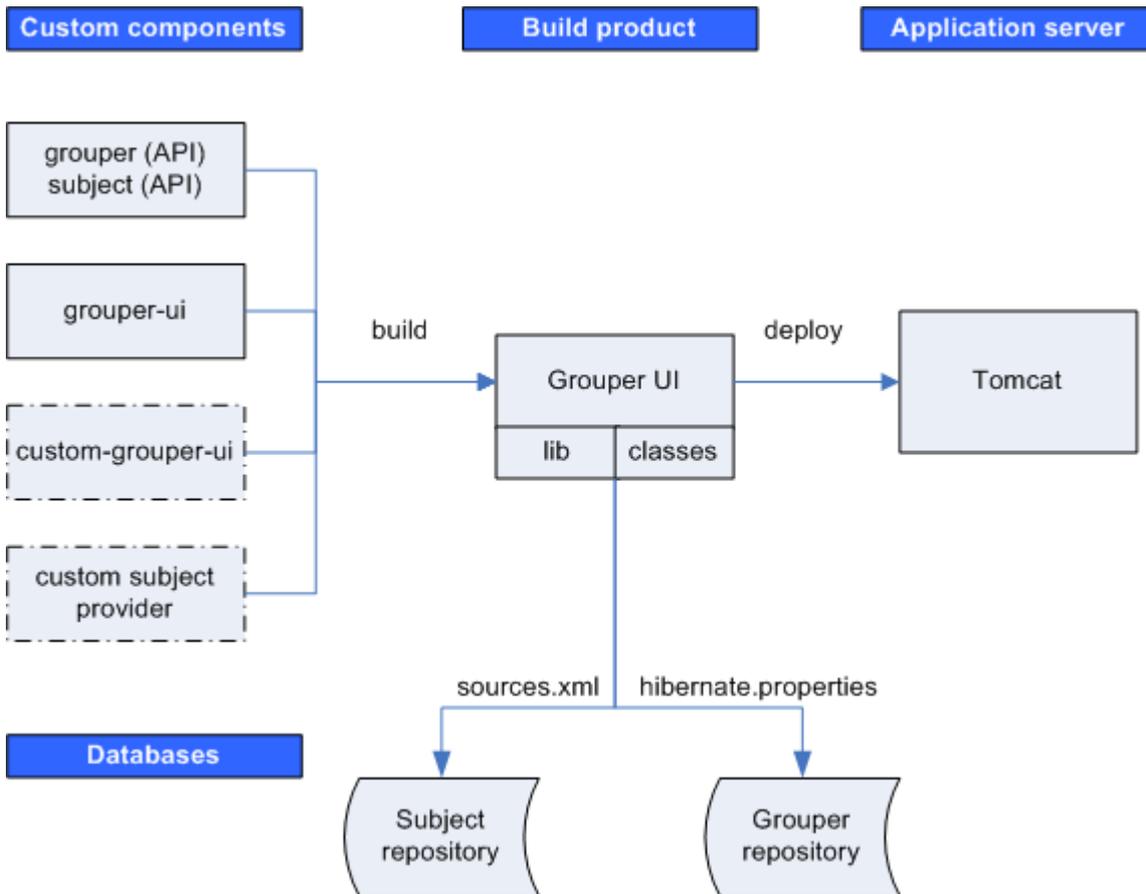
Grouper UI Components

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 (<i>optional</i>)	Sites implementing Grouper will generally want to re-brand (and perhaps heavily customise) the native Grouper UI (see Customising the Grouper

	UI).
custom subject provider (<i>optional</i>)	Depending on the identity management / person repository(s) in use, a site may also need to implement a custom Subject provider

Build Product

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

Application Server

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

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

Databases

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

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

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

The Grouper QuickStart Distribution

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

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

GROUPER: [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

This page last changed on Apr 17, 2008 by tbarton@uchicago.edu.

GROUPER: [About](#) [FAQ](#) [Software](#) [Documentation](#) [Archives](#) [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)
- [getMemberships](#): under construction
 - Will accept member filter (All, Effective, Immediate, Composite)
 - Accepts batches of subjects and groups (non-Lite)
- [hasMember](#): see if a subject is a member of a group
 - Will return true or false
 - Accepts batches of subject ids or identifiers (returns batches of true's / false's) (non-Lite)
 - Will accept member filter (All, Effective, etc)
 - Can query on field (permission)
- [getGroups](#): list groups for a subject
 - Will accept member filter (All, Effective, etc)
 - Accepts batches of subjects (non-Lite)
- [groupSave](#)
 - Create / update a group
 - Accepts batches of groups (non-Lite)
- [groupDelete](#)
 - Delete a group
 - Accepts batches of groups (non-Lite)
- [View Or Edit Privileges](#): under construction
 - View / Grant / Revoke privileges for a member and group
 - Accepts batches of subjects (non-Lite)
 - Will not fail if assigning a privilege which is already assigned, or if revoking a privilege which the subject does not have
- [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.tran
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
5. unit test
6. build a client jar back into web services to unit test
7. make some params to test stuff... (junit to throw exceptions in the middle of tx?)
8. come up with formatter and code style and remove all warnings
9. add logging filter
10. fix javadoc warnings
11. look into axis 1.4, see if error fixed, see if samples/wSDL changes, see about enums
12. add move subject service
13. add metadata service
14. add getGroups with batched groupLookup input
15. add back in privileges service
16. add back in memberships service
17. privileges: Input param replaceAllExisting?
18. filter getMember by privileges (find member?)
19. in rest add GET starting points with links to resources
20. improve auto-toString methods in resultMessage
21. look at acegi
22. add ip source filtering to grouper

Authentication for Grouper Web Services

This page last changed on Apr 17, 2008 by tbarton@uchicago.edu.

 [*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 wont 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>
```

```

        </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=1

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).



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

GrouperShell (gsh)

This page last changed on May 09, 2008 by tzeller@memphis.edu.

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.

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'
uid='d0fa765e-1439-4701-89b1-9b08b4ce9daa'
gsh 3% member.getGroups()
group: name='etc:wheel' displayName='Grouper Administration:Wheel Group'
uid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

Groups

Command	Description
addGroup(parent stem name, extension, displayExtension)	Add group to registry
delGroup(name)	Delete group from registry
getGroupAttr(group name, attr)	Get value of group attribute
getGroups(name)	Find all groups with a matching naming attribute value
setGroupAttr(group name, attr, value)	Set value of group attribute

Group Types

Command	Description
groupAddType(group name, type name)	Add type to group
groupDelType(group name, type name)	Delete type from group
groupGetTypes(group name)	Get group's types
groupHasType(group name, type name)	Check whether group had type
typeAdd(type name)	Create custom group type
typeAddAttr(type name, attr name, read, write, required)	Create custom group attribute. <i>read</i> and <i>write</i> must be an <code>AccessPrivilege</code> (e.g. <code>AccessPrivilege.ADMIN</code>)
typeAddList(type name, attr name, read, write)	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

Memberships

Command	Description
<code>addComposite(group name, composite type, left group name, right group name)</code>	Add composite membership
<code>addMember(group name, subject id)</code>	Add member to group
<code>delComposite(group name)</code>	Delete composite membership from group
<code>delMember(group name, subject id)</code>	Delete member from group
<code>getMembers(group name)</code>	Get members of group
<code>hasMember(group name, subject id)</code>	Check whether subject is member

Privileges

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

Registry

Command	Description
<code>resetRegistry()</code>	Restore registry to default state

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
setStemAttr(stem name, attr, value)	Set value of stem attribute

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
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

XML

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

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:wheel' displayName='Grouper Administration:Wheel Group'
uuid='6f77fb36-b466-481a-84a7-7af609f1ad09'
```

 Questions or comments?  Contact us.

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

Groups Management & Your Institution

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

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

This document will explain how your campus benefits from Groups Management.

...more to come soon!

 Questions or comments?  Contact us.

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

Import-Export

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

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

XML Import / Export for Grouper v1.2.0

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	Prints this message
args:	subjectIdentifier [(-id) [-name)] [-relative] [-includeParent] fileName [properties]

The above export args. can be explained as follows:

Command	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Group or Stem to export. 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	Prints this message
args:	subjectIdentifier [(-id -name -list)] filename [properties]

The above import args. can be explained as follows:

Commands	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Stem into which data will be imported. 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

use

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

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

Initializing Administration of Privileges

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

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

Initializing Administration of Grouper Privileges for v1.2.1

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:wheel' and adds one member to it.

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:

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

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

Intro FAQ

This page last changed on Jul 18, 2007 by jbibbee@internet2.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 Dec 06, 2007 by jbibbee@internet2.edu.

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

media.properties

This guide to media.properties was released with Grouper v1.2.1.

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](#)

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.jpg
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 CreateGroups ManageGroups 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 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@

#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 portion 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 V1.2.0 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             =name
search.stem.result-field              =name

```

#By setting the 'result-field-choice' properties, sites can allow users to select which
#field to use for displaying serach results

```

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

```

#Prior to V1.2.0 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 compatability - it was assumed that all Subjects woud 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 V1.2.0 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,createSubj

##subject.attributes.order.qsuob=LOGINID,LFNAME,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
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.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

 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 Apr 18, 2008 by tbarton@uchicago.edu.

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

Prerequisites

The essential prerequisite steps are:

1. [Install & Configure the Prerequisite Infrastructure](#),
2. [Establish a Database for Grouper](#), and
3. [Download the Grouper v1.2.1 Distribution](#).

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

Install & Configure the Prerequisite Infrastructure

[Ant](#) - You will need ant v1.6 or later to build Grouper. 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](#).

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. So far, Grouper instances using HSQLDB, Oracle 9i, and Postgresql 8 have been reported as working successfully.

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
build/	Compiled code
contrib/	Contributed software
doc/	Grouper API documentation
lib/	Third party .jar files required by the Grouper API

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

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.](#)

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

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 May 05, 2008 by tbarton@uchicago.edu.

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

Download Grouper

Grouper v1.3.0 Release Candidate 2

Software release: [5-May-2008]

The Internet2 Grouper team is pleased to announce the availability of Grouper v1.3.0 RC2. The audience for this release is the Grouper Working Group and early Grouper adopters. The latest stable release is [v1.2.1](#).

Release component	Description
Grouper QuickStart v1.3.0 RC2 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 v1.3.0 RC2 tarball	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.3.0 RC2 tarball	Contains the full source for the Grouper User Interface.
Grouper WS v1.3.0 RC2 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.

- [*v1.3.0 Release Notes*](#) - View the changes & fixes for the v1.3.0 release.

Grouper v1.2.1

Software release: [6-December-2007]

The Internet2 GrouperWG team is pleased to announce the availability of Grouper v1.2.1. This is primarily a maintenance release of the Grouper software. The audience for this release is the Grouper Working Group, early Grouper adopters, and the general Higher Ed community.

Release component	Description
Grouper-QuickStart v1.2.1 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 v1.2.1 tarball	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.2.1 tarball	Contains the full source for the Grouper User Interface.
Contributed Software	Provided by the Grouper community, in addition to the above core Grouper products.

- [Release Notes](#) - View the changes & fixes for the v1.2.1 release.
- [Archives](#) - For a complete listing of feature updates and changes to Grouper's current v1.2.1 and 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_2_1 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_UI_1_2_1
grouper-ui

```

- [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 v1.0 - v1.1

This page last changed on Jan 02, 2008 by jbibbee@internet2.edu.

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

Grouper Database Conversion

Grouper v1.1 works with a Grouper v1.0 database just as well as Grouper v1.0 does. However, we found that by reordering the columns in one key table (the grouper_memberships table) significant performance gains occurred across several types of operations. We also reorded the columns in a second table (grouper_members) to explore for additional gains, but found the change only marginal. But the database schema proper - the tables, their columns, indices, and relationships - did not change at all between the v1.0 and v1.1 releases, which is why Grouper v1.1 works with a v1.0 database.

But to reap that performance gain, the columns in a Grouper database must be reordered. To do that we'll use the tools first released in Grouper v1.0 to support a database conversion, should one be needed. Basically, we'll export the entire database (including its metadata) into an XML file, reset the database, then reimport it, following the steps described below.

A database conversion can take a substantial amount of time, depending of course on how large it is. One on demo HSQLDB database containing 628 stems, 640 groups, and 14,447 memberships and non-default privilege assignments, on a laptop it took 1 minute 18 seconds to export, and 3 minutes 57 seconds to re-import, taking on average about 6ms to create each direct or indirect membership and privilege. Your mileage will vary, but it will take time to convert a substantial database.

Conversion steps

This process assumes that you have the Grouper API v1.1 RC3 or later installed and configured to work with your Grouper database, and that you have a shell open on the root of the Grouper API distribution directory.

Note that release candidates 1 and 2 had a bug with the XML export capability that prevented this process from being successful.

1. **ant xml-export -Dcmd="GrouperSystem aFileName"**

The default xml-export properties are correct for doing a complete dump of the database. The filename can refer to a file in the current directory, or it can be a relative or absolute pathname.

2. **Create a new database container**

Do whatever you have to do to setup a fresh database with your RDBMS technology.

3. **Setup the SA account used by the Grouper API**

Establish the credential used by the Grouper API for database access in your new database.

4. Review `conf/grouper.hibernate.properties`

Ensure that properties declaring how the Grouper API will connect to your database are correctly declared in the `conf/grouper.hibernate.properties` file.

4. `ant schemaexport`

5. `ant db-init`

These two steps create the Grouper v1.1 schema in your new database.

6. `ant xml-import -Dcmd="GrouperSystem theSameFileName"`

This re-loads everything into the new database.

An alternative approach

Since this change is only to the column order in two tables, you can consider using SQL to export and suitably re-import the tables. Details will vary by RDBMS type, but this is likely to be far faster than the conversion described above. The two changes are:

- `grouper_members`: removed 'member_uuid' from position 5 and inserted at position 2
- `grouper_memberships`: removed 'membership_uuid' from position 2 and inserted at position 10 (the last column)

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

Database Conversion v1.2.0 - v1.2.1

This page last changed on Apr 17, 2008 by [shilen](#).

Database Conversion

Modifications have been made to the indexes on the Grouper tables for the 1.2.1 release of Grouper. Below you will find a description of each change and an example SQL statement for Oracle that you may execute to make the modification in your database.

For your convenience, we have also included SQL scripts that contain all of these index modifications described below.

1. [Oracle SQL Script](#)
2. [MySQL SQL Script](#)
3. [Postgres SQL Script](#)

Modifications to indexes:

1. Drop the concatenated index on grouper_memberships with columns list_name and list_type.

```
drop index membership_field_idx;
```

2. Drop the index on grouper_memberships.owner_id.

```
drop index membership_owner_idx;
```

3. Drop the index on grouper_memberships.mship_type.

```
drop index membership_type_idx;
```

4. Create a concatenated index on grouper_memberships using the columns member_id, list_name, and list_type.

```
create index membership_member_list_idx on grouper_memberships (member_id, list_name, list_type);
```

5. Create a concatenated index on grouper_memberships using the columns owner_id, list_name, list_type, and mship_type.

```
create index membership_owner_list_type_idx on grouper_memberships (owner_id, list_name, list_type, mship_type);
```

Grouper change log

This page last changed on May 14, 2008 by [mchzyer](#).

This document lists instructions for people with existing groups installations on how to upgrade to newer versions of grouper (or grouper related products). If you notice something missing please let us know. The instructions are in descending order based on date/release. You will find instructions below for Grouper, Grouper-ws, Grouper-ui, GSH, USDJ, 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

- 2008/05/14: 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

Release Notes

This page last changed on Apr 17, 2008 by tbarton@uchicago.edu.

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

Release Notes for Grouper v1.2.1

This page includes the new features and fixes of the current release for the Grouper v1.2.1 API and UI. Download Grouper v1.2.1 on the main [Download](#) page.

Grouper v1.2.1 is a maintenance release in which a few bugs have been fixed and performance has been substantially improved. Some modest functional enhancements have also been made to the UI. Significant gains have been achieved by tuning how the API interacts with the relational back-end and its caching strategy, and both the API and UI use new strategies to check privileges. The project team has also identified some promising new approaches to achieve even further gains that we hope to roll out in Grouper v1.3.0.

Details on the improvements can found in the Internet2 Jira issue tracker for the [API](#) and the [UI](#).

Modifications have been made to the indexes on the Grouper tables for the 1.2.1 release of Grouper. See [Database Conversion v1.2.0 \- v1.2.1](#) for a description of each change and an example SQL statement

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

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

Specsheet

This page last changed on Apr 18, 2008 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. <i>Contributions:</i> Yale CAS authentication filter.
Java	JDK v5.0.6 or later.
Compiler	Ant v1.6 or later.
Browser Requirements	<ul style="list-style-type: none">XHTML 1.0CSS 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	<p>Through servlet container via REMOTE_USER, or via user-installable filter.</p> <p><i>Contributions:</i> Yale CAS authentication filter.</p>
Java	JDK v5.0.6 or later.
Compiler	Ant v1.6 or later.
Browser Requirements	<ul style="list-style-type: none"> • XHTML 1.0 • CSS 2.1 • cookies must be enabled • no javascript needed, except for debug mode used only by UI developers

Reference

Java: If needed, download and install the latest version of JDK 1.5.0+ (5.0) from <http://java.sun.com/j2se/1.5.0>.

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

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

Supporting Your Campus

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

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

This document will discuss additional steps for a smoother running infrastructure.

...more to come soon!

 Questions or comments?  Contact us.

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

Technical FAQ

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

[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?](#)

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. But there is as yet no near-real-time "provisioning connector" that can update LDAP directories or other run-time security infrastructure services. This is perhaps the leading need of the Grouper Project at this time. But several early adopter universities have this need, and we expect to net at least one provisioning connector or other run-time interface (eg, perhaps a web services interface) for Grouper as a product of their efforts.

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.

 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.](#)

GROPER: [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	An overview of Grouper UI components.
Architecture	An explanation of the technologies used to create the Grouper UI and the specific approaches used.
Struts actions and tiles	An overview of the Struts actions and tiles defined by the Grouper UI.
Customising the Grouper UI	The QuickStart distribution contains UI customisations. This document explains those customisations - which can be used as the basis for your own site-specific customisations. Customisations may be limited to branding, page layout and text display, but can also include integrating authentication schemes and adding completely new functionality to integrate Grouper with other systems.
Grouper UI Development Environment	A description of the actual development environment used to develop the Grouper UI.
Grouper UI Contributed Code	A list of contributed code.

Unresolvable Subject Deletion Utility (USDU)

This page last changed on 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='id1'/'source'/'person' group='stem:group1' list='members'
member_uuid='e58697e4-...' subject='id2'/'source'/'person' group='stem:group2' list='members'
...
```

Find unresolvable subjects from a specified source

```
$ 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='id1'/'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](#)