Grouper v1.1 Archive

Key:	GrouperWG
Name:	Grouper
Description:	**Grouper Product Documentation for v1.1**

Index of Grouper v1.1 Documentation

Home

This page last changed on Apr 20, 2007 by tbarton@uchicago.edu.

GROUPER: **About** **FAQ** **Software** **Documentation Archives** **Contribute** **WG** **Contact**

http://grouper.internet2.edu - the official website for the Grouper Product.

Welcome to the Grouper Group Management Toolkit Wiki

Grouper Product	**Grouper Project**	**Grouper Public**
Open viewing. Editing restricted to the Grouper Developers.	*Open viewing. Editing restricted to Grouper Working Group members.*	*Open viewing/editing for anonymity.*
Grouper Product Documentation The Grouper Product space, with the latest software and documentation. ***New*** *- Grouper v1.2.0 Release Candidate 1!* - Grouper v1.1 is the current stable release. - 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. *Return to the Web*: Grouper Product Home ***New*** - Grouper Infosheet (PDF)	**The Grouper Working Group** The Grouper Project space, with the design and development work of the MACE-led Grouper Working Group. - 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. *Return to the Web:* Grouper Working Group Home - Minutes - notes from the WG bi-weekly conference calls - WG Final Documents - link coming soon!	**Grouper Public** Intended to house items for those authors wishing to retain greater privacy than offered by the Grouper Project wiki. - No Confluence login needed! - Other Grouper-related questions, comments, & suggestions welcome here! - Items will be reviewed and periodically re-fed to the mailing list(s) for group feedback.

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 v1.0 design includes support for: basic group management by distributed authorities; subgroups; composite groups (groups whose membership is determined by the union, intersection, or relative complement of two other groups); custom group types and custom attributes; traceback of indirect membership; delegation.

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

Working Group Chair   **Working Group Flywheel**

[Tom Barton](#) - University of Chicago   [Jessica Bibbee](#) - Internet2

News

... more to come soon!

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

Related Internet2 Middleware projects

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

Want Editing Permissions?

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

1. Please sign up with your email as your username, and
2. Notify Steve Olshansky <steveo AT internet2 DOT edu> of your request.

Steve will then enable your access to these pages.

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

Questions or comments? Contact us.

GROUPER: **About**   **FAQ**   **Software** **Documentation** Archives   **Contribute**   **WG**   **Contact**

Comments

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

Posted by at Sep 18, 2006.

....

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

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

Posted by at Mar 27, 2007.

....

Contributions

| GROUPER: | **About** | **FAQ** | **Software** | **Documentation** | **Archives** | **Contribute** | **WG** | **Contact** |

Contributions

Use this page to post and browse community contributions.

Software, Code, Documentation, etc.

Please attach related materials and provide links.   *NOTE: The below contributions may not all be verified by the Grouper development team.*

	Date	Code Contribution	Description	Contributor	Contributor Home	Grouper Verified
3						
2		struts-patch	Ensures that the current Strut's 'module' is maintained in links and internal forwards / includes. This enables Strut's modules to be used to provide multiple user interfaces in one web application instance.		University of Bristol, UK	
1		yale-cas-auth	Provides the CAS client library and configures Servlet API Filters to manage authentication.		University of Bristol, UK	

Use Cases

Have a use case that you would like to share or suggest? Detail it here by Adding a Page, then select the "use case" template option.

Adopter Sketches

Getting started with your implementation of Grouper? Have useful docs to share with others? Add a page for your institution here.

Brown University - Watch as Brown Grouper migrates to MACE Grouper...

Cornell University - Stay posted as Cornell progresses in our implementation of Grouper...

Ohio State University - See what GridGrouper is doing in the caBIG environment...

University of Kansas - Review several useful items that KU is sharing, as they press onwards in their Grouper implementation...

Questions or comments? Contact us.

GROUPER: **About** **FAQ** **Software** **Documentation** **Archives** **Contribute** **WG** **Contact**

About Grouper

GROUPER: About   **FAQ**   **Software** **Documentation**   **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.

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.

Questions or comments? Contact us.

Grouper Product

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

GROUPER: **About**   **FAQ**   **Software**   Documentation   **Archives**   **Contribute**   **WG**   **Contact**

Welcome to the Grouper v1.1 product documentation space.

NEW - Grouper v1.2.0 Release Candidate 1 *is available!*

Software Overview

Download - Download the latest product version from the main Grouper Software web site.

Specsheet - Offers operational specifications for the development and deployment of Grouper.

Glossary - Terms and definitions relevant to Grouper.

Release Details & Previous Releases - Details feature information and downloads of previous Grouper versions.

License - Terms under which Grouper is licensed, the Apache 2.0 license.

Grouper Trademark Guidelines and Styleguide - Usage policy for the Grouper logo.

Campus Management

Identity Management Model - How well does your infrastructure meet the needs of your campus? Compare with this model...

Groups Management & Your Institution - Understanding how your campus benefits from Groups Management.

Supporting Your Campus - Additional steps to a smoother running infrastructure.

Design Guidelines - Considerations as you prepare to deploy Grouper at your campus.

Use Cases - Find out how Grouper addresses many of the current challenges in managing groups.

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.

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](#)
- [Customizing the Grouper UI](#)
- [Grouper UI Development Environment](#)

[*API 1.1 Javadoc*](#)

[*UI 1.1 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 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co grouper-ui
```

[*Bugzilla*](#) - Bugs and fixes found here.

Grouper Archived Documentation

Only rather minor changes have been made to the Grouper v1.0 documentation to arrive at the Grouper v1.1 documentation assembled on this page. All child pages that were modified at all were first copied to a "foo v1.1" page, leaving the original "foo v1.0" page ready for archival. Please verify that you are referencing documentation that corresponds to your Grouper version.

⚠️ You may find older, outdated documentation on the [Release Details & Previous Releases](#) page. Ditto for GrouperShell, which revved from v0.0.1 to v0.1.0.

Community Area

[Documents & Presentations](#) - View others' and Post your Grouper-related drafts and final works.

[Contributions](#) - Share your code, software, use cases, and other contributions with the Grouper community.

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

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

API Building & Testing

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

Step-by-Step Instructions to Build and Test the Grouper API as of v1.0

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.lck 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.lck 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 v1.0_UI Building & Configuration section.

Questions or comments? Contact us.

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

API Configuration v1.1

Configuring the Grouper API

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 v1.1 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 - Grouper uses Apache DBCP for JDBC connection pooling and enables prepared statement pooling by default. Prepared statement pooling must be disabled for Oracle 9i with the setting:

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

Database Initialization Procedure

After setting Hibernate properties for your database, change your command shell to the grouper directory and execute the following two ant tasks to install the appropriate Groups Registry DDL and perform necessary initialization:

ant schemaexport - Generates DDL appropriate for your configured RDBMS and installs the tables.

ant db-init - Populates various tables with required logical schema information (the default set of group types and fields) and creates the root naming stem of the Groups Registry in the configured database.

If you've performed the junit testing using your production database, or for any other reason need to return the Groups Registry to its initial pristine state, do

ant db-reset - Cleans up the database, returning it to its just-initialized state.

Configuration of Source Adapters

Grouper uses 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 1.0 relies on v0.2.1 of the Subject API. Please refer to the section on Subject API v0.2.1 for detailed configuration information.

Three source adapter classes are included in the Grouper API 1.0 package. JDBCSourceAdapter and JNDISourceAdapter classes are included in subject-0.2.1.jar, and GrouperSourceAdapter is built along with the Grouper API. Every Grouper API deployment MUST include a*source* element in grouper/conf/sources.xml for the GrouperSourceAdapter so that Grouper can refer to its own groups in the same manner as other subjects.

Choosing Identifiers for Subjects

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

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

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

Grouper accommodates subject identifier issues in two ways. First, it maintains UUIDs for every subject and group within the Groups Registry. These are never exposed by the API, but are associated with externally supplied subject identifiers within the Groups Registry. This approach allows the identifier associated with a given subject to be changed without any need to change actual memberships.

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

Grouper Privileges

All configuration of Grouper privileges detailed in this section occur in the grouper/conf/grouper.properties file.

Default privileges

Grouper requires that all subjects must be explicitly granted access or naming privileges (cf Glossary v1.0), with one caveat. There is a special "subject" internal to Grouper called the ALL subject, which is a stand-in for any subject. The ALL subject can be granted a privilege in lieu of assigning that privilege explicitly to each and every subject.

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

Property Name	Value in Grouper 1.0 Distribution
groups.create.grant.all.admin	false
groups.create.grant.all.optin	false
groups.create.grant.all.optout	false
groups.create.grant.all.update	false
groups.create.grant.all.read	true
groups.create.grant.all.view	true
stems.create.grant.all.create	false
stems.create.grant.all.stem	false

Super-user Privileges

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

Property Name	Description
groups.wheel.use	"true" or "false" to enable or disable this capability.
groups.wheel.group	The group name of the group whose members are to be considered security-equivalent to GrouperSystem.

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

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 cahcing 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
privileges.access.interface	classname of the java class that implements the Access Interface
privileges.naming.interface	classname of the java class that implements the Naming Interface

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

Logging

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

In addition, there are several configuration parameters in grouper/conf/grouper.properties that may be

adjusted to control the logging of effective membership modifications in the event log.

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

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

Questions or comments? Contact us.

GROUPER: **About**   **FAQ**   **Software** **Documentation**   **Contribute**   **WG**   **Contact**

Architecture v1.0

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

Grouper UI Architecture

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:

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:

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:

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

or

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 pf 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 Actionsnormally 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.
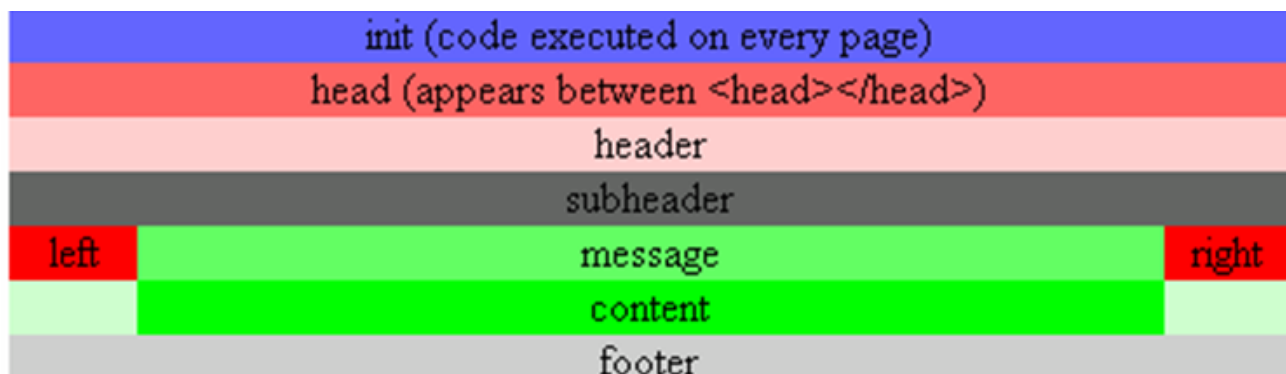
Such a page can be represented using a Tiles definition:

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.,:

```
```

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.,:

```
```

In the distributed Grouper UI it is likely that areas such asheader and footer will be static, whilstsubheader, 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:

```
```

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:

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:

or

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:

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   Contribute   WG   Contact**

Custom Group Types

GROUPER: **About** **FAQ** **Software** **Documentation** **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

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:

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

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

Using GrouperShell to create a group type

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

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

Using GrouperShell to remove a group type

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

```
gsh-0.0.1 0% subj=SubjectFinder.findById("GrouperSystem")
subject: id='GrouperSystem' type='application' source='g:isa' name='GrouperSystem'
gsh-0.0.1 1% sess=GrouperSession.start(subj)
edu.internet2.middleware.grouper.GrouperSession:
eb0c794b-09a1-4cc5-b45b-4c2e4a6d3434,'GrouperSystem','application'
gsh-0.0.1 2% type=GroupTypeFinder.find("teaching")
edu.internet2.middleware.grouper.GroupType: teaching
gsh-0.0.1 3% type.delete(sess)
```

Questions or comments? Contact us.

GROUPER: About FAQ Software Documentation Contribute WG Contact

Customizing the Grouper UI v1.0

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

How to Customise the Grouper User Interface (UI)

- 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
- 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 theGrouper 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 no where 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 Strut's Tiles to define core page components. In the standard UI these are defined
in **grouper-qs/grouper-ui/webapp/WEB-INF/tiles-def.xml**. In the custom UI some definitions are
modified and another added in the
file**grouper-qs/custom-grouper-ui/webapp/WEB-INF/tiles-def-custom.xml**. New definitions for
headerDef and *footerDef* allow site specific branding. *groupStuffDef* defines a template which is included
in the Group Summary page of the UI.

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

Defining Custom Dynamic Templates

Grouper recognises some core entities such as Groups, Stems, Subjects and Collections. The Grouper UI
dynamically chooses the appropriate template for an entity at runtime based on its type and the UI
context. The default templates for an entity and view are defined in
grouper-qs/grouper-ui/resources/grouper/media.properties. When a specific entity-view key is

not found, a default is used. Key-value pairs can be overridden, or more specific keys added to **grouper-qs/custom-grouper-ui/resources/grouper/media.properties**. The algorithms used to choose appropriate keys are described in the Javadoc for [DefaultTemplateResolverImpl](). This class can be extended to add support for other entity types, or a completely new implementation plugged in (see Javadoc for the [TemplateResolver]() interface.

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

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

In the custom UI:

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

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

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

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

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

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

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

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

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

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 implentation 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 Strut's 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. PrepareMenuActionreads **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 Strut's action **prepareMenu.do**.

Personal Groups

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

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 default deployment descriptor is found at **grouper-qs/grouper-ui/webapp/WEB-INF/web.core.xml**. The UI provies 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.ml). 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.

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 URl */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 ?*key*?
Show dynamic tiles	If selected, a hierarchy of templates used to construct the page is shown. If a template was loaded dynamically, the *view*, *entity type*, *chosen key* and *template name* 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 feedback to, or request help via the grouper-users mailing list.

Questions or comments? Contact us.

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

Developer's Guide to the Grouper API v1.0

This page last changed on Sep 27, 2006 by jbibbee@internet2.edu.

GROUPER: **About**   **FAQ**   **Software**   **Documentation**   **Contribute**   **WG**   **Contact**

Using the Grouper API to bootstrap your Groups Registry

- 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**   **Contribute**   **WG**   **Contact**

Glossary v1.0

GROUPER: **About**   **FAQ**   **Software**   **Documentation**   **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.

Note: To view an HTML version of this document, open the attachment above.

TERM	DEFINITION
Access Privileges	Privileges that determine what a ***Subject*** can do with a ***Group***. They are: • **ADMIN** - can assign access privileges and manage all group information, • **UPDATE** - can manage membership of the group (implies **READ**), • **READ** - can see the membership of the group (implies **VIEW**), and • **VIEW** - can see the group. In addition, a group may have options for its members to: • **OPTIN** - can add self to the membership, and • **OPTOUT** - can remove self from membership.
Attribute	A single-valued string associated with a ***Group*** or a ***Naming Stem***. By default, Grouper supports six attributes (one of two kinds of ***Field***): • **id** - a Grouper-assigned, globally unique identifier. • **extension**- the relative name of the group or naming stem within its parent naming stem; the contribution of a single element, such as a group or a naming stem, to the cumulative name. • **name** - used to facilitate searching for groups by name, it is a read-only string representation of the logical ordered pair of *(parent stem, extension)*. This attribute is system-maintained. The string representation of the *name* attribute is: *<parent stem>:<extension>*. • **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 *(displayName of parent stem, displayExtension)*. This attribute is system-maintained. The string representation of the *displayName* attribute is: *<displayName of parent stem>:<displayExtension>*. - **description** - a description of the group or naming stem.
Composite Group	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: - **union** - all subjects must be a member of one OR the other factor group, e.g., Group Z = members of either Group X **OR** Group Y, or Z = X U Y. - **intersection** - all subjects that are members of the first factor group AND the second factor group, e.g., Group Z = members of both Group X **AND** Group Y, or Z = X # 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**.
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	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: • six naming **Attributes**, • a **description** attribute, and • a **members** list. This information model can be extended to include additional site-defined attributes and lists.
Group Math	Any combination of groups for the purpose of creating another group based on the memberships of those groups. See ***Composite Group***.
Indirect Membership	A ***Subject*** that is a member of a ***Subgroup*** of a ***Group***, or a member of a ***Factor Group*** that contributes positively to a group's membership, has an indirect membership in the group. Also see ***Direct Membership***.
List	A multi-valued list of ***Subject*** references, (one of two kinds of ***Field***). The ***direct members*** of a group are the values of the group's **members** list. Lists are also used to identify which subjects have which ***Naming*** or ***Access Privileges***. Sites can extend a group type to include custom lists; however, their semantics are external to Grouper. See ***Group***.
Member	Any ***Subject*** in the membership list of at least one group. Also, a ***Member*** of a ***Group*** is any ***Subject*** with a ***Direct*** or ***Indirect Membership*** in the ***Group***.
Membership	The direct-only, indirect-only, or direct plus indirect members of a ***Group***. A specific variety of membership is determined by context or configuration, i.e., the default User Interface allows the user to select among these three types of membership where appropriate.
Naming Privileges	These privileges determine what a ***Subject*** can do with a ***Naming Stem***. They are: • **CREATE** - can create a group(s) named with a naming stem, and • **STEM** - can assign who has CREATE for the naming stem, and can create naming stems subordinate to this one.
Naming Stem	A string that forms the leading part of a ***Group's*** name. By linking the ability to create groups to a specified naming stem (via the **CREATE** privilege), the possibility that different groups can be given the same name is substantially reduced, and the

	name of each group can be made to reflect something about the authority under which it was created. ...see Examples below.
Stem	A synonym for a *Naming Stem*.
Subgroup	A *Group* that is a *Direct Member* of another group.
Subject	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.
Type	There are two distinct uses for this term in Grouper. • **Group Type** - each *Group* has one or more group types associated with it. The Grouper distribution contains support for a single group type called "**base**", but sites may register additional types, together with the attributes and lists associated with them, within their Grouper installation. Doing so enables management of groups with a richer information model or a more diverse set of information models. • **Subject Type** - the Subject API v0.2.1 that Grouper 1.0 relies on uses the notion of a subject type, such as "person", "group", or "computer", etc.

Examples

Step 1: Create a Root Naming Stem

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

Naming Stem uofc

attribute	value
naming stem	empty
extension	uofc
displayExtension	The University Of Chicago
name	uofc

displayName	The University Of Chicago

Step 2: Create a Group

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

Group **uofc:exec_council**

attribute	value
naming stem	uofc
extension	exec_council
displayExtension	Executive Council
name	uofc:exec_council
displayName	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**

attribute	value
naming stem	uofc
extension	bsd
displayExtension	Biological Sciences Division
name	uofc:bsd
displayName	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**

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

Grouper FAQ

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**   **Contribute**   **WG**   **Contact**

Grouper Software Download

GROUPER: **About** **FAQ** Software **Documentation** **Contribute** **WG** **Contact**

Download Grouper

Grouper v1.2.0 Release Candidates 1 and 2

We are readying the v1.2.0 release of Grouper and offer these initial release candidates for your testing and feedback. Grouper v1.2.0-RC2 Quickstart, and Grouper v1.2.0-RC1 API, and UI packages are available.

New Release Features and Fixes

Grouper v1.1

Software release: [3-November-2006]

The Internet2 GrouperWG team is pleased to announce the availability of Grouper v1.1. 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 v1.1 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 v1.1 tarball	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.1 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.

For a complete listing of feature updates and changes to Grouper's current v1.1 and previous releases, read:

- Release Details and Previous Releases

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.

Note: Grouper v1.1 works fine with Grouper v1.0 databases - the database schema itself is unchanged. However, we changed the column order in two tables to net some performance gains. The [Database Conversion v1.0 \- v1.1](#) page contains details of how to convert a Grouper v1.0 database to reap the benefit of these performance gains.

If you're interested...

- [Licensed](#) under the Apache 2.0 license.
- [Spec Sheet](#) 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:
        - <u>Connection type</u>: pserver
        - <u>User</u>: anoncvs
        - <u>Passwd</u>: <your email address>
        - <u>Host</u>: anoncvs.internet2.edu
        - <u>Repository Path</u>: /home/cvs/i2mi
        - <u>Use default port</u>: 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_1 grouper
cvs -z3 -d :pserver:anoncvs@anoncvs.internet2.edu:/home/cvs/i2mi co -r GROUPER_UI_1_1
grouper-ui
```

- [Bug Reports](#) here, please.

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

Contact

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

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

Database Conversion v1.0 - v1.1

This page last changed on Oct 27, 2006 by tbarton@uchicago.edu.

Grouper Database Conversion

Grouper v1.1 works with a Grouper a 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 terchnology.

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)

Release Details and Previous Releases

This page last changed on Mar 23, 2007 by jbibbee@internet2.edu.

| **GROUPER**: | **About** | **FAQ** | **Software** | **Documentation** | **Contribute** | **WG** | **Contact** |

❓ Questions about Grouper's development or a particular feature? ⓘ ***Contact us***.

Previous versions of the software may also be downloaded from this location.

Grouper Release Details & Previous Releases

Click here for Grouper News, a lower-level, highly-detailed version of the below feature sets. archived docsArchived documentation for past releases may also be found below.

Release	Description of Feature Adds, Improvements, Changes, Revisions, & Fixes	Date
Grouper v1.2	Grouper v1.2 will include the following funcationality improvements and miscellaneous changes: • • •	**?-2007**
Grouper v1.1	Release **v1.1** is mostly a maintenance release; however, there are a few functional and operational enhancements: • Added: Additional GrouperQuery filters, especially to support provisioning applications • Improved: Configurable Privilege and Subject caching regimens • Improved: UI support for removing many members from a group • Added: Support for XML Import/Export & custom group types & fields to the GrouperShell command line shell • Improved: Query performance through caching	**30-Nov-06**

	# [Grouper-QuickStart v1.1](#) - tarball with documentation & instructions for cvs access.
	    # [Grouper API v1.1](#) tarball...
	    # [Grouper UI v1.1](#) tarball...	

Grouper v1.0	Release **v1.0** is the first production release of the Grouper product:
	
	• Added: [Group Math](#)
	• Added: [Custom Group Types](#), attributes and lists
	• Added: Basic [XML export-and-import](#) of the Groups Registry
	• Added: [GrouperShell](#) command line shell - gsh
	• Improved: Query performance through caching
	
	    # [Grouper-QuickStart v1.0](#) - tarball with documentation & instructions for cvs access.
	    # [Grouper API v1.0](#) tarball...
	    # [Grouper UI v1.0](#) tarball...
	    # [Archived Documentation, v1.0](#) (PDF)
	
	[API 1.0 Javadoc](#)
	[UI 1.0 Javadoc](#)
	
	The Grouper v1.0 UI embodies our first attempt to enable two tasks that are complicated in a UI context: managing composite groups, and managing custom group types and attributes. We've prepared a brief [tutorial](#) to help smooth your exploration of these new UI capabilities.
		20-Jul-06

Grouper v0.9	Release **v0.9** offers the following new and/or different items from the v0.6 pre-release:	**19-Dec-05**
	- Added: New query API that enables sites to add their own custom queries - Improved: Enhancements to the Access and Naming interfaces - Added: New "all" subject that can be assigned memberships and granted privileges that map to all subjects identifiable through the Subject API - Added: "Wheel" group whose members have root-like privileges within the API - Improved: Enhanced UI support for searching - Improved: Enhanced UI support for management of effective memberships and privileges - Improved: Logging capabilities	

The v0.9 toolkit includes a full UI designed to be deployed to a java application server, java source implementing the Grouper 0.9 API, documentation for developers and implementers, and sample utilities.

> \# Grouper-QuickStart v0.9 tarball with documentation & instructions for cvs access.
> \# Grouper API v0.9 tarball...
> \# Grouper UI v0.9 tarball...
> \# archived docsArchived Documentation, v0.9:
> > Grouper API
> > Grouper
> Changes

	[Grouper ERD](#) [Grouper Javadoc](#) [Grouper Known Issues](#) [Grouper News](#) [Grouper Roadmap](#) [Grouper User Interface](#) [UI - Demo](#) [UI - Components](#) [UI - Architecture](#) [UI - Struts Actions & Tiles](#) [UI - Customizing the UI](#) [UI - Contributed Code](#) [UI - Development Environment](#) [UI components](#)	
Grouper v0.6	Release **v0.6** is the initial UI pre-release, supporting Phase 1 functionality: • Improved: Subject interface • Added: Full implementations of the privilege interfaces • Fixed: Proper schema validation (#267) • Added: GrouperAccess.has() • Improved: search and browse capabilities • Added: Search by the name, extension, or displayName attributes of groups [S&B#1] • Added: Search by the name, extension, or displayName attributes of namespaces [S&B#2] • Added: Search by the name, extension, or displayName attributes of	**16-Sep-05**

	groups scoped to groups within a namespace subordinate to a given stem [S&B#7]
	
	• Added: Search by the name, extension, or displayName attributes of namespaces scoped to namespaces subordinate to a given stem [S&B#8]
	• Added: More verbose and precise error reporting via exceptions
	• Added: Ability to delete groups containing members
	
	       #
	Grouper-QuickStart v0.6 tarball with documentation & instructions for cvs access.
	       # Grouper API v.6 tarball...
	       # Grouper UI v0.6 tarball...	

Grouper v0.5.6	The API release **v0.5.6** contains fixes to several bugs found in the 0.5.5 pre-release. From its [News file:] • Fixed: Effective memberships for non-"members" lists (e.g. access and naming privileges) were being calculated incorrectly (#350) • Fixed: Non-root subjects could not create stems or groups (#353) • Fixed: STEM, not ADMIN, needed to modify namespace attributes (#352) • Fixed: Added NullGrouperAttribute class (which extends GrouperAttribute) to handle group attributes that either do not have values or have had their values deleted (#356)	**29-Apr-05**

	- Fixed: GrouperMember.load(session subject) replaces GrouperMember.load (subject) (#348) - Fixed: GrouperGroup.loadByID() now returns properly casted GrouperGroup objects (#349) - Fixed: GrouperStem.loadByID() now returns properly casted GrouperStem objects (#349) # [Grouper API v0.5.6](#) tarball with documentation & instructions for cvs access.	
Grouper v0.5.1	Release **v0.5.1** is a maintenance release, and includes the following additions and revisions: - Revised: Hibernate session and transaction handling code* Revised: effective membership algorithm - Added: GrouperStem class for managing and representing namespaces - Improved: compatibility with Oracle - Added: public methods to list all groups and stems within a given stem # [Grouper v0.5.1](#) tarball with documentation & instructions for cvs access.	
Grouper v0.5	The API release **v0.5** is the first release of Grouper: - Added: Creation, update, and removal of groups from the Groups Registry - Added: Subgroups - Added: Export capabilities - Added: Limited querying	**Dec-04**

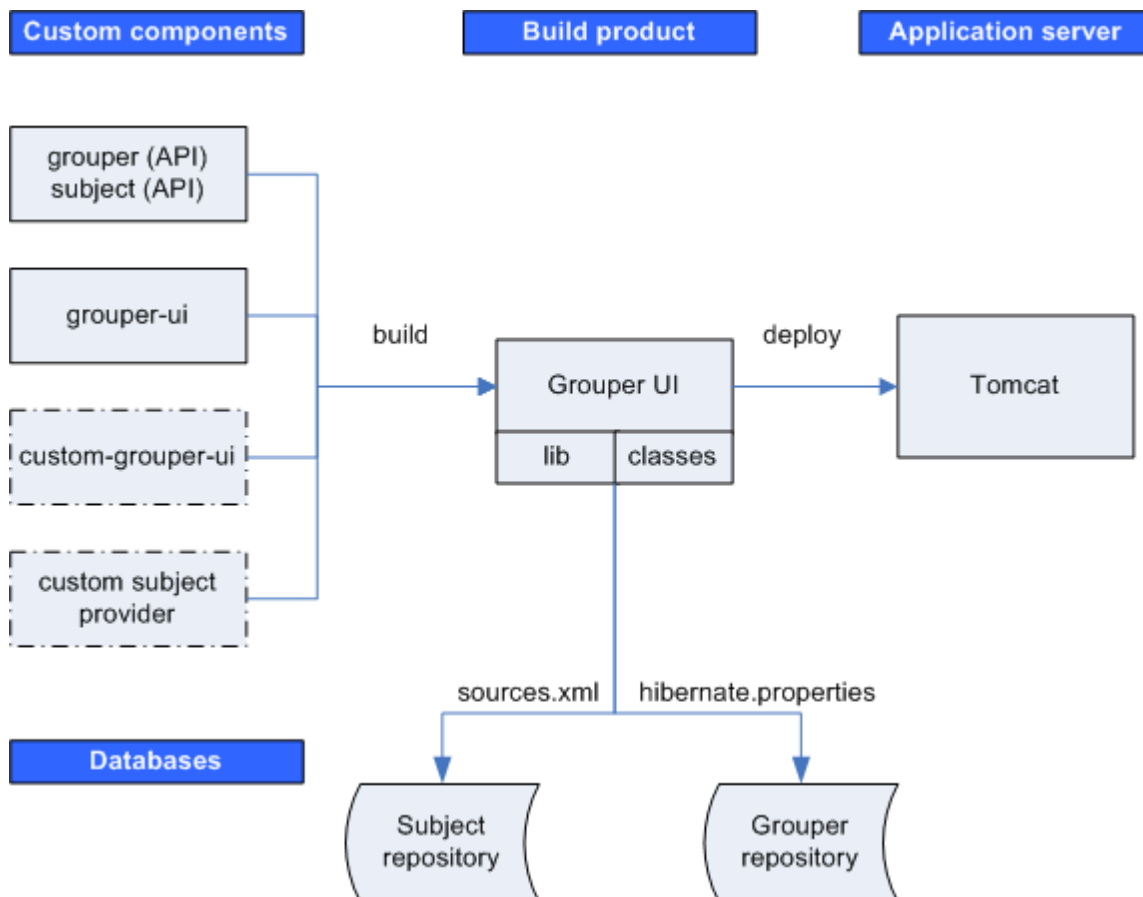| | • Added: Graphical user interface for manual groups management
• Added: Logging

    # [Grouper API v0.5](#) tarball with documentation & instructions for cvs access. | |

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

GROUPER: **About** **FAQ** **Software Documentation** **Contribute** **WG** **Contact**

Grouper UI Components v1.0

Overview

Custom Components

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

	repository(s) in use, a site may also need to implement a custom Subject provider

Build Product

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

Application Server

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

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

Databases

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

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

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

The Grouper QuickStart Distribution

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

Questions or comments? Contact us.

GROUPER: **About** **FAQ** **Software Documentation** **Contribute** **WG** **Contact**

Grouper UI Development Environment v1.0

| GROUPER: | **About** | **FAQ** | **Software** | **Documentation** | **Contribute** | **WG** | **Contact** |

Grouper UI Development Environment

Introduction

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 webappp 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.

gsh

GROUPER: **About**   **FAQ**   **Software** **Documentation**   **Contribute**   **WG**   **Contact**

GrouperShell v0.1.0

GrouperShell (gsh) is a shell for administering and interacting with the Grouper API. It can be used in both a batch and interactive manner.

API Compability

This version of of **GrouperShell** is compatible with Grouper 1.1.

Build GrouperShell

Test GrouperShell's scripting capabilities

Grouper must have a JDBCSourceAdapter for subjects for the test suite to complete successfully. This need is adequately met by testing GrouperShell using the same grouper/conf directory contents used to test the Grouper API.

Note: In some environments the tests may fail due to a bug exposed by the testing procedure which might not indicate any actual error with the gsh utility itself. This is known to occur, for example, under cygwin on Windows. An alternative testing process is to run each of the test suites, which are gsh scripts, individually as follows:

Build `gsh.jar` file in the `dist` subdirectory

Build javadoc in `doc/html`

Use GrouperShell

- Run **GrouperShell** in an interactive manner from a Unix-like environment:

- Run **GrouperShell** (crudely) from Ant:

- Read **GrouperShell** commands from STDIN:

- Read **GrouperShell** commands from a script file:

GrouperShell Commands

Command	Description
addComposite(group, type, left group, right group.	Add composite membership.
addGroup(parent, extension, displayExtension)	Add group beneath parent stem with the specified extension and displayExtension.
addMember(group, subject id)	Add subject as a member to the group.
addRootStem(extension, displayExtension)	Add root stem with the specified extension and displayExtension.
addStem(parent, extension, displayExtension)	Add stem beneath parent stem with the specified extension and displayExtension.
addSubject(id, type, name)	Add a HibernateSubject to the Groups Registry.
delComposite(group)	Delete composite membership from the specified group.
delGroup(name)	Delete group with the specified name.
delMember(group, subject id)	Remove subject as a member of the group.
delStem(name)	Delete stem with the specified name.
exit	Terminate shell.
findSubject(id)	Find a subject.
findSubject(id, type)	Find a subject.
findSubject(id, type, source)	Find a subject.
getGroupAttr(stem, attr)	Get value of group's attr attribute.
getGroups(name)	Find all groups with name in any naming attribute value.
getMembers(group)	Get members of the group.
getSources()	Find all Subject sources.
getStemAttr(stem, attr)	Get value of stem's attr attribute.
getStems(name)	Find all stems with name in any naming attribute value.
grantPriv(name, subject id, privilege)	Grant privilege to subject id on name. *privilege* must be an *AccessPrivilege* (e.g. `AccessPrivilege.ADMIN`) or *NamingPrivilege* (e.g. `NamingPrivilege.STEM`) constant.
groupAddType(group, type)	Add group type *type* to *group*.
groupDelType(group, type)	Delete group type *type* from *group*.
groupGetTypes(group)	Get *group*'s group types.
groupHasType(group, type)	Check whether *group* has group type *type*.

hasMember(group, subject id)	Is subject a member of this group.
hasPriv(name, subject id, privilege)	Does subject id have privilege on name? *privilege* must be a n_AccessPrivilege_ (e.g. `AccessPrivilege.ADMIN`) or *NamingPrivilege* (e.g. `NamingPrivilege.STEM`) constant.
help()	Display usage information.
history()	Print commands that have been run.
history(N)	Print the last N commands that have been run.
last()	Run the last command executed.
last(N)	Execute command number N.
p(command)	Pretty print results. This command is more useful when GSH_DEVEL is enabled.
quit	Terminate shell.
resetRegistry()	Restore the Groups Registry to a default state.
revokePriv(name, subject id, privilege)	Revoke privilege from subject id on name. *privilege* must be an *AccessPrivilege* (e.g. `AccessPrivilege.ADMIN`) or *NamingPrivilege* (e.g. `NamingPrivilege.STEM`) constant.
setGroupAttr(group, attr, value)	Set value of *group*'s *attr* attribute.
setStemAttr(stem, attr, value)	Set value of *stem*'s *attr* attribute.
typeAdd(name)	Create custom group named *name*.
typeAddAttr(type, name, read, write, required)	Create a custom group attribute named *name* on group type *type*. *read* and *write* must be an `AccessPrivilege` (e.g. `AccessPrivilege.ADMIN`).
typeAddList(type, name, read, write)	Create a custom membership list named *name* on group type *type*. *read* and *write* must be an `AccessPrivilege` (e.g. `AccessPrivilege.ADMIN`).
typeDel(name)	Delete the group type named *name*.
typeDelField(type, name)	Delete the custom field named *name* from group type *type*.
typeFind(name)	Find the group type named *name*
typeGetFields(name)	Get the fields associated with the group type named *name*.
version()	Return version information.
xmlFromFile(filename)	Load Groups Registry with XML contained in *filename*.
xmlFromString(xml)	Load Groups Registry with XML in the *xml* string.
xmlFromURL(url)	Load Groups Registry with XML located at *url*.
xmlToFile(filename)	Exports Groups Registry to *filename*.
xmlToString()	Exports Groups Registry to string.

xmlUpdateFromFile(filename)	Update Groups Registry with XML contained in *filename*.
xmlUpdateFromString(xml)	Update Groups Registry with XML in the *xml* string.
xmlUpdateFromURL(url)	Update Groups Registry with XML located at *url*.

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

GrouperShell Variables

Variable	Description
GSH_DEBUG	If set to true, stack traces will be printed upon failure.
GSH_DEVEL	If set to true, summaries of returned objects are not automatically printed.
GSH_TIMER	If set to true, the time taken to evaluate each command will be displayed.

Example:

Questions or comments? Contact us.

Import-Export v1.1

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

Grouper v1.1 XML Import / Export

Grouper includes XML import / export tools. Exported XML may be used for:

- provisioning to other systems
- reporting
- backups
- switching database backends - including to upgraded schemas (required by new Grouper API versions) in the same database

Imported XML may be used for:

- loading - adding to or updating existing Stems, Groups and Group Types. Whole or partial Grouper registries can be exported, and subsequently imported at a specified Stem (or the Root Stem if not specified) in the *new* instance.*
- initialising a new, *empty* registry to a known state** - useful for demos, testing and system recovery

In general, exported data can be imported into the same Grouper instance it was exported from, or a different instance. Stems and Groups and Group Types will be created, if not already present, or updated if they already exist (depending on import options provided).

The XML formats for import and export are very similar, however, there are some differences.

The export format:

- defines what is actually exported,
- includes some meta data about the export,

while the import format:

- allows import options to be embedded in the XML,
- defines additional attributes for Stems and Groups which may affect the importing of Stems and Groups,
- does not require all of the information that is exported.

Any tool which can create XML, in the correct format, can be used as a loader.

*To successfully load Subject data, the new Grouper instance must be configured with the same Subject Sources. The export tool does not export Subject registries. Subjects which cannot be resolved will be logged, but otherwise ignored.

**Although data can be exported from one Grouper instance and imported into another, system attributes are not maintained. A group with the same name will have a different uuid and create times, etc, will reflect the time of import rather than the creation time in the original Grouper instance. A future version of the import tool may have options to maintain system attributes.

Export Tool in More Detail

A Java class, XmlExporter, provides the export functionality. It can be run from the command line, from within Java code, or as an ant task in grouper/build.xml:

```
ant xml-export -Dcmd="command line arguments"
```

The command line usage is:

Command	Summary of args.
args: -h	Prints this message
args:	subjectIdentifier [(-id] [-name)] [-relative] [-includeParent] fileName [properties]

The above export args. can be explained as follows:

Command	Description
subjectIdentifier	Identifies a Subject 'who' will create a GrouperSession.
-id	The Uuid of a Group or Stem to export. 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 *context group* for which the Subject is being processed as a member*, list member or privilegee. *Actually the API will prevent a Group becoming a member of itself
*	As desribed 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 v1.1 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.

This page last changed on Feb 26, 2007 by jbibbee@internet2.edu.

| GROUPER: | **About** | **FAQ** | **Software** | **Documentation** | **Contribute** | **WG** | **Contact** |

Initializing Administration of Grouper Privileges

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. With version 1.0 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** | **Contribute** | **WG** | **Contact** |

Prerequisites v1.0

GROUPER: **About** **FAQ** **Software** **Documentation** **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 v1.0 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 v1.4.2 or later to build & run Grouper, and you will need Apache Tomcat to run Grouper. Java & Tomcat versions must be chosen to work together. Choose either:

- Java 1.5 (5.0) 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. These instructions presume that you will use Apache v1.3 or v2.X.

JK Connector - The **mod_jk** connector is used to pipe requests between Apache and Tomcat.

- Configure mod_jk.
  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.

- 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:

```

```

It is critical to ensure that login names are being successfully passed between Apache and Tomcat via mod_jk. The parameter tomcatAuthentication="false" must be present in the <Ajp13Connector> configuration element to ensure that the user's identity is passed from Apache to the servlet environment.

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 v1.0 Distribution

The Download page contains instructions for downloading the following two components of Grouper v1.0:

- Grouper API, and
- Grouper UI.

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 v1.0 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** **Contribute** **WG** **Contact**

Specsheet v1.0

This page last changed on Nov 02, 2006 by tbarton@uchicago.edu.

Grouper Product Spec Sheet v1.0-v1.1

Component	Details
RDBMS	• 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	• Servlet API Version 2.3. • Known to run properly in Apache Tomcat 4.1 and 5.5. • Have not tested/verified other servlet containers.
Authentication	Through servlet container via REMOTE_USER, or via user-installable filter. *Contributions*: Yale CAS authentication filter.
Java	JDK v1.4.2 or later.
Compiler	Ant v1.6 or later.
Browser Requirements	• 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 JDK 1.5.0_04+ (5.0) from http://java.sun.com/j2se/1.5.0.

Questions or comments? Contact us.

UI Building & Configuration v1.1

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

Configuring and Deploying the Grouper v1.1 UI

In this section we describe how to configure, build, and deploy the Grouper UI.

Section	Description
Lightweight UI Configuration	Just a few easy bits, leaving in-depth coverage of the UI's extensive customization capabilities to the Grouper UI Guide.
Building & Deploying	All about building and deploying the UI.

Lightweight UI Configuration

In this subsection we'll describe how to replace the logo image included in the Grouper UI tarball and highlight a couple of settings in the grouper-ui/resources/grouper/media.properties file that control how the UI uses subject attributes.

Using Your Own Logo Image

1. Place the image file in grouper-ui/webapp/grouper/images/.

2. Replace the "image.organisation-logo" property in grouper-ui/resources/grouper/media.properties with the name of the file emplaced in the previous step.

Controlling the Use of Subject Attributes in the UI

Subjects are presented in the UI in various contexts. The Grouper UI supports a limited capability to control which subject attributes are displayed in which contexts. Here's a list of associated properties in the grouper-ui/resources/grouper/media.properties file and how to use them.

Property Name	Description	Possible Values
search.group.result-field	The name of the group naming attribute displayed in search results and on the "saved groups" page.	Any group naming attribute: name, displayName, extension, displayExtension, id.
search.stem.result-field	The name of the stem naming attribute displayed in search results.	Any stem naming attribute: name, displayName, extension, displayExtension.

In the case of groups or stems displayed in search results, the media properties above only determine defaults. UI users are enabled to change the default for a UI session. The range of options they are presented are given in the following table.

Property Name	Description	Possible Values
search.group.result-field-choice	The set of choices of naming attributes that a UI user is presented with for display of groups in search results. If the set is empty, the user cannot change the default.	Space-separated list of zero or more of any of the group naming attributes: name, displayName, extension, displayExtension, id.
search.stem.result-field-choice	The set of choices of naming attributes that a UI user is presented with for display of stems in search results. If the set is empty, the user cannot change the default.	Space-separated list of zero or more of any of the stem naming attributes: name, displayName, extension, displayExtension.

As of Grouper 1.1 the following keys from nav.properties are no longer used:

```
groups.edit.name=Extension
groups.edit.display-name=Display extension
groups.edit.description=Description
```

Instead a more generic mechanism is used which can be applied to all group attributes, including custom ones. Use:

```
field.displayName.displayName=Display name
field.displayName.displayExtension=Display extension
field.displayName.extension=Extension
field.displayName.name=Name
field.displayName.description=Description
field.displayName._any=Any attribute*
```

*Appears on the advanced search page for groups.

This change fixes an issue where inconsistent names were used to describe group attributes in different parts of the UI.

Building & Deploying

1. **Copy grouper-ui/build.properties.template to grouper-ui/build.properties**.

2. **Review grouper-ui/build.properties**.

   - If you want the build script to automatically install the UI in your Tomcat instance, uncomment and set the appropriate value for deploy.home. If you do not set this you will need to copy the UI to your Tomcat installation's webapps directory. You will probably want to define the default.webapp.folder to suit how you intend to develop or customise the UI. See the Grouper UI Development Environment for options.

- Make sure you set the grouper.folder property to the location of your Grouper installation.

3. **Copy grouper-ui/template-tomcat-context.xml to grouper-ui/tomcat-context.xml** (or the value of the property deploy.context.xml if you have changed this).
    - Tomcat specific configuration can be added in this file e.g., container managed data sources.

4. **Change directory to grouper-ui and type "ant".**
    - A list of build targets is displayed. If you have set deploy.home enter "default". Otherwise type "dist" or "war". If the former copy <dist.home>/grouper to <TOMCAT_HOME>/webapps, or if the latter, copy <dist.home>/grouper.war to <TOMCAT_HOME>/webapps.
    - If you want to take advantage of the 'nice' targets you must uncomment and set appropriate values for all the deploy properties in grouper-ui/build.properties.

    Note: The build process will attempt to create a directory peer to the grouper-ui directory. Hence, the directory grouper-ui/.. must be writable.

Questions or comments? Contact us.

GROUPER: **About** **FAQ** **Software** **Documentation** **Contribute** **WG** **Contact**

UI Customization Guide

This page last changed on Nov 03, 2006 by tbarton@uchicago.edu.

GROUPER: **About** **FAQ** **Software** **Documentation** **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.

Contact Information

GROUPER: **About** **FAQ** **Software** **Documentation** **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 .

Product Questions - Stay logged in *or* anonymously leave questions related to Grouper (must be logged out for anonymity.)

Documentation Comments & Suggestions - Stay logged in *or* anonymously leave feedback (must be logged out for anonymity.)

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 with the following in the *subject line*:

    subscribe <list name> <your name>

For example:

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

subscribe grouper-users Jane Doe
Questions and comments regarding the implementation of Grouper should be directed to this list. Messages will be archived, and can be accessed for reference. It is anticipated that subsequent discussion will be supported by both the Grouper developers *and* implementing sites. Your contributions here - comments, questions, and concerns - will be of great value to the general Grouper community. This

should include, but not be limited to, questions about the documentation, undocumented problems, installation or operational issues, and other product issues that arise.

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

To **unsubscribe** from any of these lists, send email to sympa AT internet2 DOT edu 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**   **Contribute**   **WG**   **Contact**