Space Details

Key:	GrouperWG
Name:	GrouperWG
Description:	Grouper Working Group wiki
Creator (Creation Date):	steveo@internet2.edu (Apr 07, 2006)
Last Modifier (Mod. Date):	steveo@internet2.edu (Apr 07, 2006)

Available Pages

- Home
  - About Grouper
  - Grouper Product Documentation
    - API Building & Testing
    - API Configuration
    - Architecture
    - Contact Information
    - Deployment Overview
    - Glossary
    - Grouper Software Download
    - Grouper UI Components
    - import-export
    - Initializing Administration of Privileges
    - License
    - Prerequisites
    - Release Details & Previous Releases
      - News
    - Specsheet
    - UI Building & Configuration
    - UI Customization Guide
  - GrouperWG Wiki
    - Contributions
    - Extended Discussion
    - Presentations & Documents
- Grouper Web & Wiki Outline
- To Dos

This page last changed on Jun 07, 2006 by tbarton@uchicago.edu.

Welcome to the Grouper Product & Project Space.

More **About Grouper**. Questions or comments? ***Contact us***.

The 3 unique areas below have been created for your convenience.

Grouper Wiki

Initially, this working space will house all the documentation and software offered by Grouper. Eventually, it is intended to house the lower-level, more technical works of the Grouper Working Group.

- *Grouper Product Documentation*
  - Intended to house the more technical documents regarding the first production release of the Grouper product.
  - *Open viewing. Editing restricted; certain documents will be open for editing by WG developers/members.*

- **GrouperWG Wiki**
  - Intended to house items beyond the convenience of the mailing list.
  - Contribute your campus' software, documentation, use cases here.
  - Storage for Member presentations, detailed documents, proposals, etc.
  - *Open viewing. Editing restricted to Grouper Working Group members*
- **GrouperWG\-Public**
  - Intended to house items for those authors wishing to retain greater privacy than offered by the GrouperWG Wiki.
  - Other Grouper-related questions, comments, & suggestions welcome here!
  - Items will be reviewed by Jessica, and periodically re-fed to the mailing list(s) for group feedback.
  - *Open viewing/editing for **anonymity**.*

Grouper Web

http://grouper.internet2.edu This space eventually intended to house the higher-level documentation and software download; updates will be noted below:

- [Home] [About] [FAQ] [Software] [Documentation] (*Quick links will be added once the pages go online*)

- *Grouper Working Group* - The Grouper Project web space, with the design and development work of the MACE-led Grouper Working Group.
  - http://middleware.internet2.edu/dir/groups/#MinutesMinutes - notes from the WG bi-weekly conference calls
  - Documents - working drafts submitted by WG members (to be moved over to WG forum wiki

space?)

+Grouper Mailing Lists+

This space intended for Grouper-related questions, discussion items, news events, release updates, etc.

- Share your campus' questions, concerns, & exptertise with the Grouper community.
- Subscribe to 1 or all 3 Grouper mailing lists:
  - grouper-user
  - grouper-dev
  - grouper-announce
- Archives available here.

Questions or comments? *Contact us*.

About Grouper

This page last changed on Jun 08, 2006 by ghbrett.

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 2004, 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.

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 and 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, Renee Frost, Ann West, and Steve Olshansky from Internet2.

Development of this software was supported with funding from Internet2, University of Chicago, University of Bristol, and the NSF Middleware Initiative (NSF 02-028, Grant No. OCI-0330626). 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 Product Documentation

This page last changed on Jun 09, 2006 by tbarton@uchicago.edu.

Welcome to the Grouper v1.0 product documentation space.

Questions or comments? *Contact us*.

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 - This is an internal link to the 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.

Campus Management

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

- Deployment Overview - What you should know before getting started with a deployment of Grouper.
- 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.

Import/Export Tool - Documentation for the XML Import/Export tool.

Applications Development

Developer's Guide to the Grouper API - Blair's doc

Grouper UI Customization Guide - click here for more information regarding the following documents:

- Grouper UI Components
- Architecture
- Struts actions and tiles
- Customising the Grouper UI
- Grouper UI Development Environment

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.

Community Area

Presentations & Documents - 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.

Contact us if you have additional comments or suggestions.

API Building & Testing

This page last changed on Jun 13, 2006 by tbarton@uchicago.edu.

Step-by-Step Instructions to Build and Test the Grouper API

These instructions assume that you have a shell open on the grouper-api 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-api/build directory. It also places default forms of three configuration files in the grouper-api/conf directory. To facilitate testing, these should not be modified at this point in the deployment process.

Note: If you're running in cygwin, you will need to fix two pathnames in the grouper-api/conf/log4j.properties file that is generated during this step. Those are the "log4j.appender.event.File" and "log4j.appender.rfile.File" properties. The issue is mixing of forward and backward slashes - they should all be forward-leaning, unix-like.

2. **ant schemaexport** - This will generate the DDL appropriate for the database configured in the grouper-api/conf/grouper.hibernate.properties file and install the Groups Registry tables. The default database, designed for testing, is located in grouper-api/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.

4. **ant db-init-jdbcsource** - This innstalls tables required by the I2MI Subject interface's JDBC source adapter, necessary only for testing.

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

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**.

Note: ant db-init-jdbcsource must have been run to run the test suite.

Note: If it says "BUILD FAILED", the tests may indeed still have succeeded. Check to see if there is a message like this in the test output, i.e., in the area with rows of dots:

```
FATAL GrouperSession: unable to get subject associated with session.
```

This error, if it occurs, can safely be ignored.

Note: If you're running in cygwin or Windows, you may ignore any of the following errors, should they occur:

- "testGetStartTime",

- "testGetCreateAttrs", and
- "testGetModifyAttrsNotModified".

The Grouper API is now built and tested. Assuming there were no errors (apart from those acknowledged above), 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 incorporating all of the Grouper API classes in the grouper-api/dist/lib directory.

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 Initializing Administration of Privileges section.

API Configuration

This page last changed on Jun 13, 2006 by tbarton@uchicago.edu.

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-api/conf/grouper.hibernate.properties. After modifying the default properties as needed, Grouper API ant tasks are used to create and install the Groups Registry schema and initialize the database.

General property settings

The 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 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-api 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 install 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.

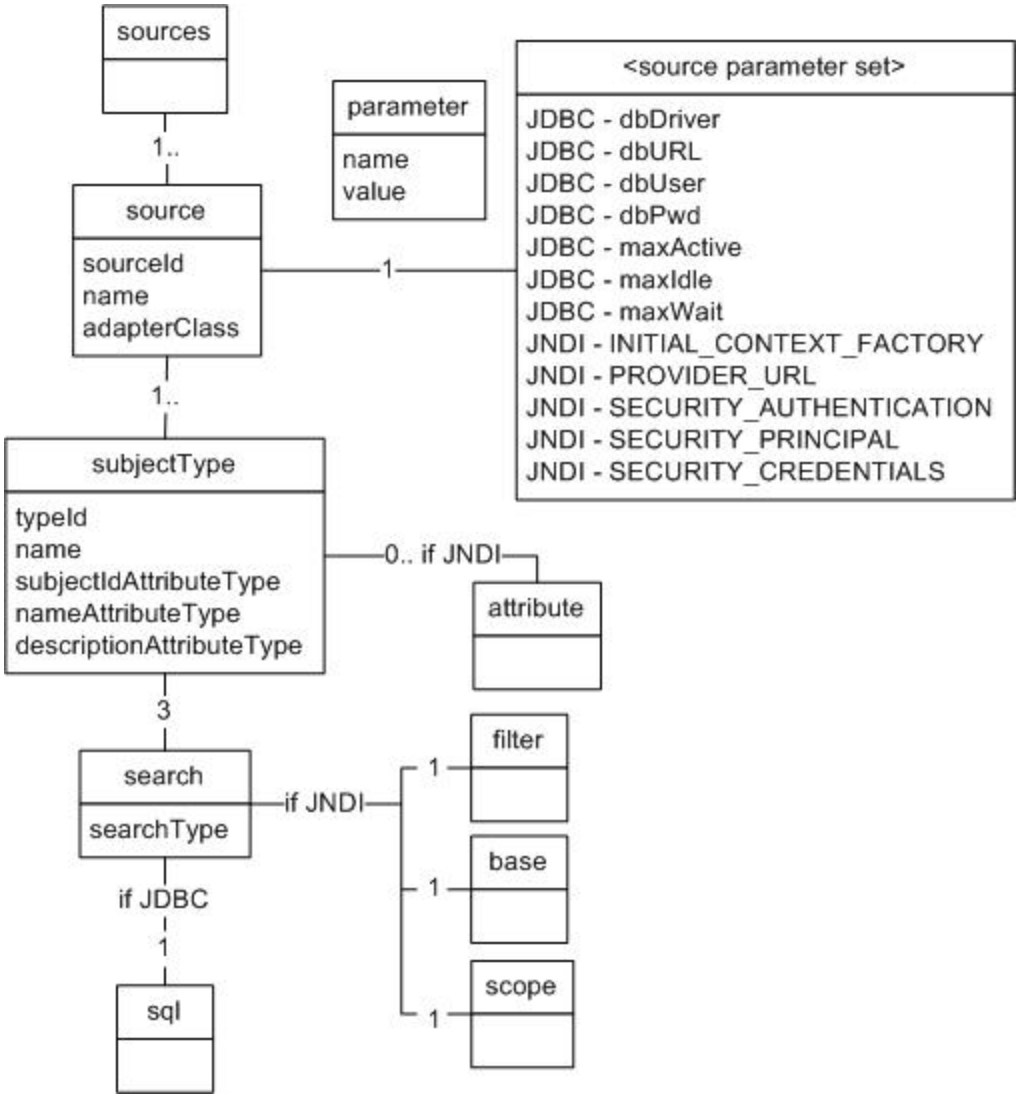
Configuration of Source Adapters

"Subjects" are the things 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 maintain identifiers in some type of "identity source". With the sole exception of Grouper groups, Grouper treats all subjects opaquely. See XXX for further background and details concerning subjects, source adapters, and other aspects of the Subject API.

Grouper uses one or more "source adapters" to integrate with external identity sources. Each one connects with a single back-end store using JDBC or JNDI. Back-end stores containing one or more than one type of subject are supported. Grouper makes no specific assumptions about the schema of any subject types. Instead, sections of the configuration file, grouper-api/conf/sources.xml, declare the details of each back-end store, the identifier to be used for each subject type it contains, how to select and search for subjects of each type, and which subject attributes should be made available to Grouper

for display in the UI.

The schema for the Subject API v1.0 sources.xml configuration file is graphically presented below. In this representation, each box is an element whose name is in the top portion of the box. Attributes for each element are listed in the bottom portion of each box, and subordinate elements are connected by a line with a cardinality indicator. Individual parameters in a parameter set are entered using a common "parameter" element schema. The set of parameters required depends on whether the source element is configuring an instance of the JDBCSourceAdapter or an instance of the JNDISourceAdapter.



The sources element

A sources.xml file contains a single **sources** element with one or more subordinate **sources** elements.

The source element

Each **sources** element configures one instance of a source adapter. Its attributes are

sourceId	A string identifying this identity source
name	A displayed name for this identity source
adapterClass	The classname of the java class configured by this element

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

The parameter element

The general form of a parameter declaration is

```
<parameter name=parameter_name value=parameter_value />
```

The parameters required for each source adapter class and descriptions of each follow. The GrouperSourceAdapter requires no parameters.

adapterClass	parameter name	parameter value
JDBCSourceAdapter	dbDriver	JDBC driver classname
JDBCSourceAdapter	dbURL	JDBC URL for the database
JDBCSourceAdapter	dbUser	database user
JDBCSourceAdapter	dbPwd	database user's password
JDBCSourceAdapter	maxActive	refer to Apache Commons DBCP documentation
JDBCSourceAdapter	maxIdle	refer to Apache Commons DBCP documentation
JDBCSourceAdapter	maxWait	refer to Apache Commons DBCP documentation
JNDISourceAdapter	INITIAL_CONTEXT_FACTORY	A string specific to the java you are using. For Sun's java it is "com.sun.jndi.ldap.LdapCtxFactory". See theJNDI documentation
JNDISourceAdapter	PROVIDER_URL	The LDAP URL of the LDAP server to connect to.
JNDISourceAdapter	SECURITY_AUTHENTICATION	See the list of allowable values
JNDISourceAdapter	SECURITY_PRINCIPAL	The DN to BIND as to the LDAP server specified in the PROVIDER_URL.
JNDISourceAdapter	SECURITY_CREDENTIALS	A hashed password, clear-text password, key, certificate, whatever you use to

		authenticate the SECURITY_PRINCIPAL to the LDAP server at the PROVIDER_URL.

The subjectType element

A **subjectType** element is used to define how subjects of a given type are found in the back-end identity store and presented to Grouper. Each type of subject in this identity source that is to be made available to Grouper must have its own **subjectType** element. Its attributes are:

typeId	A string identifying this identity source
name	A displayed name for this identity source
subjectIdAttributeType	The name of the column (for JDBC) or attribute (for JNDI) containing subjectId's
nameAttributeType	The name of the column (for JDBC) or attribute (for JNDI) containing subject names
descriptionAttributeType	The name of the column (for JDBC) or attribute (for JNDI) containing subject descriptions

The GrouperSourceAdapter does not require the subjectIdAttributeType, nameAttributeType, or descriptionAttributeType attributes to be declared. In Grouper API 1.0 the GrouperSourceAdapter presents group subjects with "subjectId" set to the "name" field of a Grouper group and the "name" subject attribute set to the "displayName" field of a Grouper group.

The attribute element

For the JNDISourceAdapter, the set of attributes returned from the JNDI source by any of the Subject API search methods is the union of the attributes listed in the "subjectAttributeType", "nameAttributeType", and "descriptionAttributeType" attributes of the subjectType element, together with any declared in **attribute** elements. The resultant set is used to construct subject objects. Each **attribute** element specifies a single attribute type. Example:

```
<attribute>attribute_type</attribute>
```

The search element

The Subject API defines three methods used to select or search for subjects. There must be one search element for each of these three methods, and the "searchType" attribute is the corresponding method name, as given in the table below. The parameter set for each search element defines how the selection or search is to be carried out against the back-end identity store, and which columns or attributes will be used as attributes of the subject objects that are returned. The string "%TERM%" should be used in search filters or WHERE clauses - it is replaced by the selection criterion or search term presented to the corresponding method.

searchType	%TERM% is ...	What the search should accomplish
getSubject	a subjectId value	Select the unique record or entry with subjectId=%TERM%, or none if %TERM% is no subject's subjectId.
getSubjectByIdentifier	the value of an identifying attribute	Select the unique record or entry which has %TERM% for the value of one of its identifying columns or attributes, or none if %TERM% is not the value of any subject's identifying column or attribute.
search	a string	Select all records or entries in which the %TERM% causes a match.

The "getSubject" method is used to select a specific subject from the back-end identity store, for example, to show the name and department of a person belonging to a group.

The "getSubjectByIdentifier" method enables identifying a subject by means of a column or attribute different from that used as the subjectId. For example, if a UI user authenticates with a loginId, but the subjectId is an opaque registryId, this method is used to identify the subject given their loginId.

The "search" method is used to help a UI user select the subject they want from a list. It is typically implemented as a substring search on several non-identifying columns or attributes such as lastname, firstname, and department. The results of a search are displayed in a checkbox list to the UI user.

The sql element

The value of this element is a (possibly compound) SQL statement. Before being executed, all occurrences of the %TERM% macro in the SQL statement are replaced with the corresponding method's argument. The SQL statement should return exactly one table with zero or more rows. Each row corresponds to exactly one subject, and the column names of the returned table are used as the attribute names of the subject objects created for each row. The set of rows is assumed to be the set of all subjects meeting the selection or search criterion of the containing **search** element.

The **sql** element is only used for configuring the JDBCSourceAdapter.

The filter, base, and scope elements

These elements are only used for configuring the JDBCSourceAdapter. They correspond to the various parts of an LDAP URL of the same name. Thus, the **filter** element defines a boolean search filter, the **base** and **scope** specify the portion of the Directory Information Tree to be searched, and zero or more **attribute** elements are appended to a list of attributes to be returned with each matching entry.

The **scope** element MUST contain one of the values "OBJECT_SCOPE", "ONELEVEL_SCOPE", or "SUBTREE_SCOPE", which corresponds to an RFC2255 scope parameter of "0", "1", or "2", respectively.

The **base** element is the DN (distinguished name) of an entry in the directory which is the root of the portion of the Directory Information Tree to be searched.

Example of a sources.xml file

```
<?xml version="1.0" encoding="utf-8"?>

<sources>

  <!-- Group Subject Resolver -->

  <source adapterClass="edu.internet2.middleware.grouper.GrouperSourceAdapter"
    sourceId="g:gsa"
    name="Grouper: Group Source Adapter"/>
    <subjectType typeId="group" name="Group"/>
  </source>

  <!-- Example JDBC source adapter configuration -->

  <source adapterClass="edu.internet2.middleware.subject.provider.JDBCSourceAdapter"
    sourceId="uc-people"
    name="UC People"/>
    <subjectType typeId="person" name="Person"
      subjectIdAttributeType="id"
      nameAttributeType="name"
      descriptionAttributeType="description"/>
      <parameter name="dbDriver" value="org.hsqldb.jdbcDriver"/>
      <parameter name="dbURL" value="jdbc:hsqldb:hsql://localhost:9001/uc-people"/>
      <parameter name="dbUser" value="sa"/>
      <parameter name="dbPwd" value=""/>
      <parameter name="maxActive" value="4"/>
      <parameter name="maxIdle" value="2"/>
      <parameter name="maxWait" value="5"/>
      <search searchType="getSubject"/>
        <sql>
select id,
  concat(firstname, concat(' ', lastname)) as name,
  concat(lastname, concat(', ', firstname)) as lfname,
  lastname, firstname, middlename,
  account.name as loginid,
  department, curriculum, appointment
from individual
  left join account on (account.individualid = id)
  left join faculty on (faculty.individualid = id)
  left join staff on (staff.individualid = id)
  left join student on (student.individualid = id)
where (id='%TERM%')
        </sql>
      </search>
      <search searchType="getSubjectByIdentifier"/>
        <sql>
select id,
  concat(firstname, concat(' ', lastname)) as name,
  concat(lastname, concat(', ', firstname)) as lfname,
  lastname, firstname, middlename,
  account.name as loginid,
  department, curriculum, appointment
from individual
  left join account on (account.individualid = id)
  left join faculty on (faculty.individualid = id)
  left join staff on (staff.individualid = id)
  left join student on (student.individualid = id)
where (account.name='%TERM%')
        </sql>
      </search>
      <search searchType="search"/>
        <sql>
select id,
  concat(firstname, concat(' ', lastname)) as name,
  concat(lastname, concat(', ', firstname)) as lfname,
  lastname, firstname, middlename,
  account.name as loginid,
  department, curriculum, appointment
```

```
from individual
  left join account on (account.individualid = id)
  left join faculty on (faculty.individualid = id)
  left join staff on (staff.individualid = id)
  left join student on (student.individualid = id)
where (firstname like '%%TERM%%')
  or (lastname like '%%TERM%%')
  or (department like '%%TERM%%')
  or (account.name like '%%TERM%%')
        </sql>
      </search>
    </subjectType>
  </source>

  <!-- Example JNDI source adapter configuration -->

  <source adapterClass="edu.internet2.middleware.subject.provider.JNDISourceAdapter"
    sourceId="kitn-person"
    name="KITN People"/>
    <parameter name="INITIAL_CONTEXT_FACTORY" value="com.sun.jndi.ldap.LdapCtxFactory"/>
    <parameter name="PROVIDER_URL" value="ldap://ldap.example.edu:389/"/>
    <parameter name="SECURITY_AUTHENTICATION" value="simple"/>
    <parameter name="SECURITY_PRINCIPAL" value="%BIND_DN%"/>
    <parameter name="SECURITY_CREDENTIALS" value="%PASSWORD%"/>
    <subjectType typeId="kitn-person" name="KITN Person"
      subjectIdAttributeType="kitnEduPersonRegID"
      nameAttributeType="cn"
      descriptionAttributeType="description"/>
      <attribute>uid</attribute>
      <attribute>department</attribute>

      <!-- Scope Values can be: OBJECT_SCOPE, ONELEVEL_SCOPE, SUBTREE_SCOPE -->
      <search searchType="getSubject"/>
        <filter>(&amp;(kitnEduPersonRegId=%TERM%)(objectclass=kitnEduPerson))</filter>
        <scope>SUBTREE_SCOPE</scope>
        <base>ou=people,dc=kitn,dc=edu</base>
      </search>

      <!-- this search and the one above return the basic set of attributes:
kitnEduPersonRegID, cn, description -->
      <search searchType="getSubjectByIdentifier"/>
        <filter>(&amp;(uid=%TERM%)(objectclass=kitnEduPerson))</filter>
        <scope>SUBTREE_SCOPE</scope>
        <base>ou=people,dc=kitn,dc=edu</base>
      </search>

      <!-- return additional attributes for this search, to help humans recognize subjects -->
      <search> searchType="search"/>
        <filter>
(&amp;(|(uid=%TERM%)(cn=*%TERM%*)(kitnEduPersonRegId=%TERM%))(objectclass=kitnEduPerson))
        </filter>
        <scope>SUBTREE_SCOPE</scope>
        <base>ou=people,dc=kitn,dc=edu</base>
      </search>
    </subjectType>
  </source>
</sources>
```

Grouper Privileges

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

Default privileges

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

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

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

Super-user privileges

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

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

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

Using a privilege management system external to Grouper

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

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

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

Logging

Logging is configured in the grouper-api/conf/log4j.properties configuration file. By default Grouper will write event log information to grouper-api/event.log while more low-level logging will be sent to grouper-api/dist/run/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-api/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
```

This page last changed on Jun 08, 2006 by jbibbee@internet2.edu.

Grouper UI Architecture

1. Introduction
2. The architecture and how it matches the requirements
   2.1. Core technology
   2.2. Framework
   2.3. Internationalization
   2.4. Extensibility
   2.5. Separation of core Grouper UI code from site-specific code
   2.6. Accessibility*
3. Implementation details
   3.1. Grouper UI out of the box
   3.2. Grouper UI tailored for the University of Bristol (UoB)
   3.3. Supporting an additional language
   3.4. Overloading in ResourceBundles
      3.4.1. How it works
      3.4.2. Multiple UIs one instance
      3.4.3. Chaining ResourceBundles
   3.5. Overloading Struts config elements
   3.6. Additional hooks to extend the Grouper UI
   3.7. Page composition using Tiles
   3.8. Content composition using Tiles
   3.9. Dynamic tiles
   3.10. JSP Tag libraries

1. Introduction

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

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

2. The Architecture and How It Matches the Requirements

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

2.1. Core Technology

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

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

2.2. Framework

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

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

2.3. Internationalization

Struts supports internationalizationthrough 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 overriden 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 ResourceBundlesautomatically 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 focusses 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 acessed by servlets within this web application. For consistency all pages are processed by the Struts ActionServlet.
lib	.jar files placed here are available to the web application through its classloader. Jar files for the Struts framework and the Grouper API would go here

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

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

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

Note the configparameter. The ActionServlet initialization is driven by this file.

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

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

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

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

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

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

The file contents would be something like:

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

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

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

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

or

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

This assumes that nav and navMap have been made available as JSTL variables. The Grouper UI code is responsible for setting them up in each users' session.
In this case, no matter what the locale, each user would get the same text, the default Grouper UI text, however, as discussed in the next section it is relatively straightforward to provide a new language, or even a variation on the default language e.g. British english as opposed to US english.

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

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

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

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

```
      </init-param>
      <load-on-startup>2</load-on-startup>
</servlet>
```

Note the config/i2miparameter. 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.xmlfile has a modified plug-in section:

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

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 theconfig 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 verison 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 Localeobjects 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 ->
  Unknown macro: {no key/value}

If the Locale was set to es, grouper.audit-trail would returnlog. Therefore, any keys overriden in nav_uob_en_GB would need to be overriden 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:

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

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

Other page definitions can extend the BaseDef e.g.

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

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

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

In the distributed Grouper UI it is likely that areas such 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:

```
<%@include file="/WEB-INF/jsp/include.jsp"%>

<tiles:insert attribute="init" />
<html:html locale="true">
<head>
      <tiles:insert attribute="head"/>
</head>
<body>
      <div id="Header">
           <tiles:insert attribute="header" />
      </div>
      <div id="Navbar">
           <tiles:insert attribute='subheader'/>
      </div>

      <div id="Sidebar">
           <tiles:insert attribute="left" />
      </div>
      <div id="ContentSpace">
           <div id="TitleBox">
                <tiles:insert attribute="title" />
           </div>
           <c:if test="${!empty message}">
                <div id="Message">
                     <tiles:insert attribute="message" />
                </div>
           </c:if>
           <!--content-->
           <div id="Content">
                <tiles:insert attribute='content' />
           </div>
           <!--/content-->
     </div>
     <div id="Right">
         <tiles:insert attribute="right" />
     </div>
     <div id="Footer">
         <tiles:insert attribute="footer" />
     </div>
</body>
</html:html>
```

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

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

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

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

3.8. Content composition using Tiles

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

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

Manage Groups Image

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

1. Bread crumb trail showing where you are in the hierarchy
2. The child stems and groups for the current location in the hierarchy (not actually used in the example below)
3. A tile which displays a stem
4. A tile which displays a group
5. A tile for displaying a Search groups form.
6. A tile for showing what actions are available for the current stem
   All of these tiles could be re-used elsewhere. In addition they can be parameterized in exactly the same way as BaseDef in the previous section. e.g.,

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

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

The tile would be inserted into the content thus:

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

or

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

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

3.9. Dynamic tiles

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

A generic dynamic tile has been defined thus:

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

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

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

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

3.10. JSP Tag libraries

Where possible the Java Standard Tag Library will be used in templates rather than scriptlets. Other open source tag libraries available from the Apache Jakarta sitemay 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.

Contact Information

This page last changed on May 31, 2006 by jbibbee@internet2.edu.

Email Us - If you have questions or comments about the wiki, documentation, and/or the Grouper project, please email us at: <grouper-info@internet2.edu>.

Support & Resources

Grouper Working Group
If you are interested in or have questions regarding the development of Grouper, please visit the Grouper Working Group home. Below is a description of the mailing lists you can join to assist your deployment of Grouper.

Grouper Mailing Lists

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-announce Jane Doe
Newsletter for periodic news and announcements about Grouper - very low traffic.

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@internet2.edu> mailing list. A bi-weekly Grouper (dev) Working Group conference call is held in discussion of development efforts.

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

This should include, but not be limited to, questions about the documentation, undocumented problems, installation or operational issues, and other product issues that arise.
To unsubscribe from any of these lists, send email to sympa AT internet2 DOT edu with the subject:
unsubscribe grouper-announce
unsubscribe grouper-dev
unsubscribe grouper-users

Archive: <grouper-announce@internet2.edu>
Archive: <grouper-dev@internet2.edu>
Archive: <grouper-users@internet2.edu>
Archive: <i2mi-ui@internet2.edu> - Internet2 Middleware Initiative SW User Interface Development list

Deployment Overview

This page last changed on Jun 09, 2006 by tbarton@uchicago.edu.

Note: This page is just a placeholder for now, noting what might go here.

Before getting started with a deployment of Grouper...

- Database
- System Administration
- Servlet Container
- External Authentication
- Optional External Authorization
- Identity Sources & Choice of Subject Identifiers
- Guidelines for Naming Stems

Glossary

This page last changed on Jun 01, 2006 by jbibbee@internet2.edu.

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 glossary.html attachment above.

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:

- **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.

... see Group; also Examples below.

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. See Indirect Membership.

Factor Group

A ***Group*** in combination (**union, intersection,** or **relative compliment)** with that of another factor group, which defines the membership of a resulting ***Composite Group***.

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. If a group is made a member, i.e., a ***Subgroup***, of another group, the members of the group will also be made members. A group must be created in an existing ***Naming Stem***. 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.

List

A multi-valued list of ***Subject*** references. The ***direct members*** of a group are the values of its **members** list. Lists are also used to identify which subjects have which ***Naming or Access Privileges***.

Sites can extend a group type to include other lists; however, their semantics are external to Grouper. See Group.

Member

Any ***Subject*** within the membership list of a group; a member may be a person, group, application, service, etc., as configured per Grouper installation.

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 ***Listed*** as a member of another group.

Subject

An abstraction of any object whose ***Group*** **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 system 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** 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***
parent 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**
parent 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

Subsequent display values now propagate down through subordinate namespaces as well, e.g the Biological Sciences Division within U of C:

naming stem, uofc:bsd

attribute	value
parent 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, within the above naming stem, and is displayed as follows:

group, uofc:bsd:eis_staff

attribute	value
parent 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 Software Download

This page last changed on Jun 07, 2006 by tbarton@uchicago.edu.

Download Grouper

Grouper v1.0
Software release: [30-Jun-2006]

The Internet2 GrouperWG team is pleased to announce the availability of Grouper v1.0. This is the first production-level 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 v1.0 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 v1.0 tarball	Contains the full source for the Grouper Application Program Interface.
Grouper UI v1.0 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 v1.0 and previous releases, read:

- Release Details & Previous Releases

Grouper v1.0 is the first major release that satisfies the requirements of a production level privileges management system. 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 Documentation page.

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.
- Grouper WG Wiki houses information, documents, presentations, and contributions of the Grouper WG members.

- CVS offers access to the Grouper source code.
  - Connection type: pserver
  - User: anoncvs

    - Passwd: <your email address>
    - Host: anoncvs.internet2.edu
    - Repository Path: /home/cvs/i2mi
    - Use default port: yes
- Bug Reports here, please.

NOTE WELL: All Internet2 Activities are governed by the Internet2 Intellectual Property Framework.

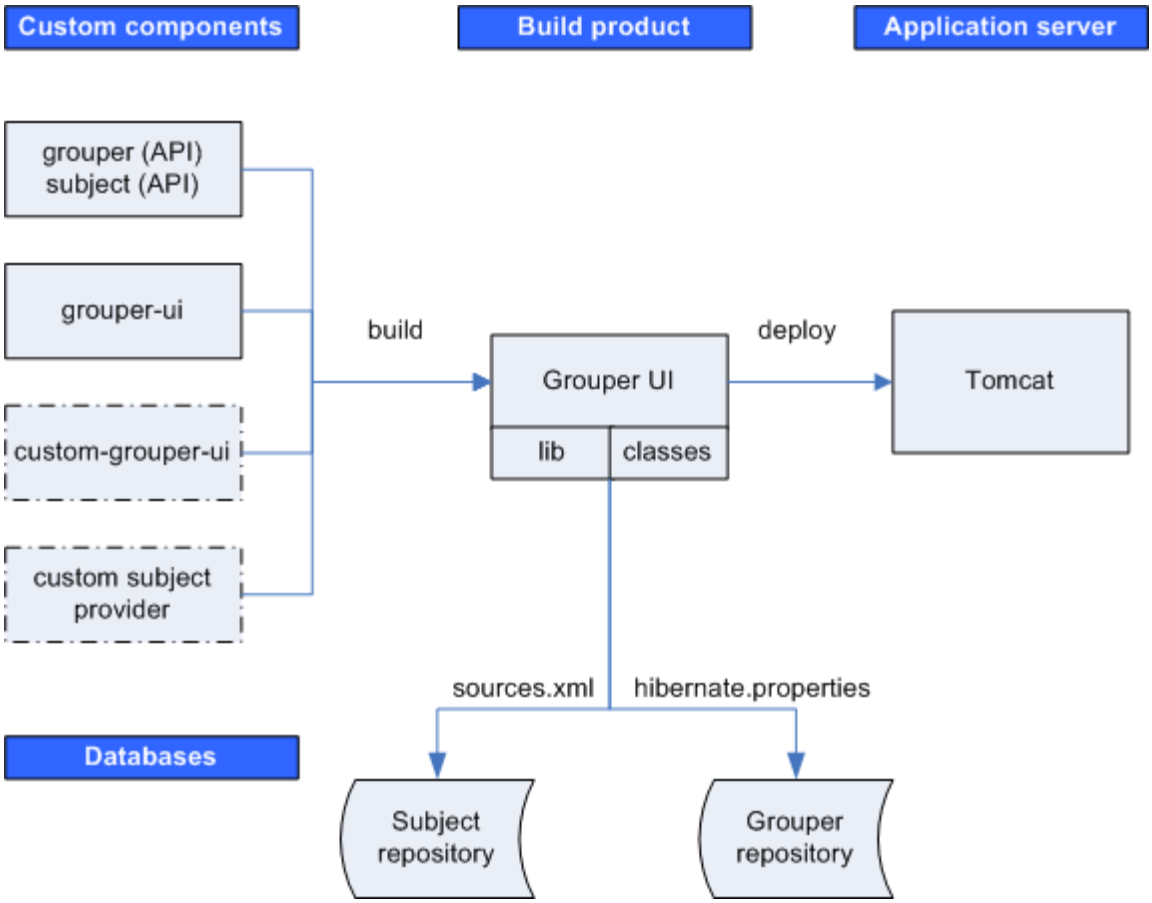
Contact

If you have a question about the Grouper software, you may Grouper Contact Informationcontact us.

Grouper UI Components

This page last changed on Jun 07, 2006 by jbibbee@internet2.edu.

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 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.

This page last changed on May 25, 2006 by tbarton@uchicago.edu.

Grouper XML import / export

As of version 1.0, Grouper includes XML import / export tools. Exported XML may be used for:

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

Imported XML may be used for:

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

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

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

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

while the import format:

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

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

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

**Although data can be exported from one Grouper instance and imported into another, system attributes are not maintained. A group with the same name will have a different uuid and create times,

etc, will reflect the time of import rather than the creation time in the original Grouper instance. A future version of the import tool may have options to maintain system attributes.

Export tool in more detail

A Java class, XmlExporter, provides the export functionality. it can be run from the command line or from within Java code. The command line usage is: args: -h, Prints this message
args: subjectIdentifier \(\-id | \-name \) \-relative \-includeParent fileName properties

subjectIdentifier, Identifies a Subject 'who' will create a
GrouperSession
-id, The Uuid of a Group or Stem to export
-name, The name of a Group or Stem to export
-relative, If id or name specified do not export parent
Stems
-includeParent, If id or name identifies a Stem export this
stem and child Stems or Groups
filename, The file where exported data will be written.
Will overwrite existing files
properties, The name of a standard Java properties file
which configures the export. Check Javadoc for
a list of properties. If 'properties' is not
specified, XmlExporter will look for
'export.properties' in the working directory.
If this file does not exist XmlExporter will
look on the classpath. If 'properties' is not
specified and 'export.properties' cannot be
found, the export will fail.
The JavaDoc describes the export methods. Including a method which can be used to export an arbitrary Collection of Stems, Groups, Subjects or Memberships returned by various Grouper API methods. This means that the results of any *list* or *search* methods can be exported.

An XML Schema which describes the exported XML is available.

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.

Import tool in more detail

A Java class, XmlImporter, provides the import functionality. it can be run from the command line or from within Java code. The command line usage is: args: -h, Prints this message
args: subjectIdentifier (-id filename properties

subjectIdentifier, Identifies a Subject 'who' will create a GrouperSession
-id, The Uuid of a Stem, into which, data will be imported*
-name, The name of a Stem, into which, data will be imported*
*If no -id / -name is specified, use=ROOT stem
-list, File contains a flat list of Stems or Groups which may be updated. Missing Stems and Groups are not created
filename, The file to import
properties, The name of a standard Java properties file which configures the import. Check Javadoc for a list of properties. If 'properties' is not specified, XmlImporter will look for 'import.properties' in the working directory. If this file does not exist XmlImporter will look on the classpath. If 'properties' is not specified and 'import.properties' cannot be found and import options are not included in the XML, the import will fail.
The JavaDoc describes the load methods.

An XML Schema which describes the format of XML which can be loaded is available.

It is possible to generate an XML file which validates against the schema, but which does not load properly. The annotations in the schema describe appropriate usage of attributes and elements.

The Grouper QuickStart includes a demo registry. quickstart.xml is a minimal XML import file which creates the demo registry*.
*For Grouper 0.9 and earlier, the demo registry was created by loading an XML file using a contributed XmlLoader class. The format of XML recognised by XmlLoader is significantly different to the *official* format understood by XmlImporter, however, if created files in the old format, these can still be loaded at the Root stem, simply add <data></data> inside the <registry> tags but outside your actual Styem and Group data. You can then export the registry to obtain an XML file in the new format. The XmlLoader format will not be supported in the next Grouper release.

When generating XML in the import format it is likely that relationships between stems and groups will need to be specified. This is problematic because the uuids of groups and stems are unknown prior to creation. In addition, it is not always possible to know the full name of a new Stem or Group, because this will depend on which stem it is imported into. When importing Subjects which 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.

Initializing Administration of Privileges

This page last changed on Jun 09, 2006 by tbarton@uchicago.edu.

Note: This section is under construction.

Prior to v1.0, Grouper required on-going, though 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 principals who can choose to act with the privilege of GrouperSystem. The UI enables them to select when to act as ordniary users and when to act with elevated privileges. Hence, it is necessary to use the GrouperSystem account only once, during installation, to bootstrap the designation of these individuals.

Grouper 1.0 lacks an installation tool specific to this purpose, but there are two approaches to completing this task. One is to use the Grouper UI, which requires a possibly temporary arrangement to authenticate the GrouperSystem account. The other is to use the contributed command line tool called **cdg**. Both are explained below.

> Note: We might provide an installation tool for this purpose yet, so I'm holding up any further doc on the alternative work arounds.

Authenticating GrouperSystem

There are several ways in which to arrange for 'GrouperSystem' to be authenticated to the Grouper UI.

Add a "GrouperSystem" account to your external authentication service.

Bypass Apache's protection by accessing the UI through Tomcat directly, and use Tomcat's basic authentication.

Deploy a second instance of the UI protected with Apache's basic authentication, and use Apache's htpasswd utility to create a local account for GrouperSystem.

To use the second method, add the following lines to your tomcat-users.xml file:

```
<role rolename="grouper_user"/>
<user username="GrouperSystem" password="chang3m3" roles="grouper_user"/>
```

Bootstrapping the Wheel Group

If you've enabled the wheel group (cf. wheel group) you must create the group named in the groups.wheel.group property in the grouper-api/conf/grouper.properties configuration file and add some members to that group. Use the UI to create the root naming stem, create any subordinate naming stems, and finally the wheel group itself. It is essential that the "name" attribute of the group you've created (as displayed on the group summary page, for example) exactly matches the value of the groups.wheel.group property.

Using cdg to Bootstrap the Wheel Group

The cdg tool is located in the grouper-api/contrib/cdg directory. Follow the instructions in the README file there to build it, test it, and create the cdg.jar file.

License

This page last changed on Jun 06, 2006 by jbibbee@internet2.edu.

Grouper, as of version 0.9, is licensed under the Apache 2.0 license. See http://www.apache.org/licenses/LICENSE-2.0.html for a copy of this license.

NOTE WELL:
All Internet2 Activities are governed by the *Internet2 Intellectual Property Framework*.

Internet2 Contributor License Agreement:

To clarify the intellectual property license granted with contributions of software from any person or entity (the "Contributor"), Internet2 would like to have an Internet2 Contributor License Agreement on file that has been signed by the Contributor, indicating agreement to the license terms. Please download, print, and complete the corresponding Internet2 Contributor License Agreement for an Individual (pdf) or Company (pdf)and send it by facsimile to Internet2 at +1-734-913-4255, followed by regular mail to Attn: John Kennedy, Internet2, 1000 Oakbrook Drive, Suite 300, Ann Arbor MI 48104 U.S.A.

Questions or comments? *Contact us*.

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 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 Shibboleth components to Tomcat. This may be done by including the following text directly in httpd.conf, or making a separate file and including it in httpd.conf.

```
<IfModule \!mod_jk.c>
  LoadModule jk_module libexec/mod_jk.so
</IfModule>
JkWorkersFile "/usr/local/tomcat/conf/jk/workers.properties"
JkLogFile "/usr/local/apache/logs/mod_jk.log"
JkLogLevel emerg
JkMount /grouper/* ajp13
```

- Add address="127.0.0.1" to Tomcat's server.xml inside the <Ajp13Connector> configuration element to prevent off-host access.
  - *For Tomcat 5.5* or newer, add request.tomcatAuthentication="false" to the <Ajp13Connector> configuration element in server.xml to ensure that the user's identity is passed from Apache to the servlet environment.
  - *For Tomcat 5.0.x* or older, add tomcatAuthentication="false" to the <Ajp13Connector>

configuration element in server.xml to ensure that the user's identity is passed from Apache to the servlet environment.
    - *Tomcat 4.1.x* defaults to having the Coyote connector enabled in /conf/server.xml. This fails with mod_jk and must be commented out. Then, uncomment and modify the traditional AJP 1.3 connector as indicated above.
- The AJP13Connector for tomcat is not compatible with the new JMX support. To remove some warnings that will appear in the Tomcat log every time Tomcat is restarted, comment out all of the JMX stuff (anything that says "mbeans") from server.xml.

Apache-based user authentication - The interaction between the Grouper UI and an Apache-based local authentication system is implemented by providing the UI with the identity of the browser user through REMOTE_USER. Any authentication system that is capable of protecting a block of webspace using httpd.conf and populating the REMOTE_USERheader variable is compatible with Grouper. This associates the appropriate authentication mechanism with the URL of the Grouper servlet, ensuring users authenticate and that their login name is passed to Grouper. The following example demonstrates use of a very basic authentication method with the Grouper UI:

```
<Location /grouper>
  AuthType Basic
  AuthName "Example University Login"
  AuthUserFile /usr/local/apache/conf/user.db
  require valid-user
</Location>
```

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

There are two components to Grouper v1.0:
- Grouper API, and
- Grouper UI.

The download page contains instructions for downloading both components.

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-api/	**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-api/lib/README lists the third party software used by Grouper and identifies the version, source, and license for each.

Release Details & Previous Releases

This page last changed on Jun 07, 2006 by jbibbee@internet2.edu.

This document will house the pertinent README information, including new features, updates, changes, and bug fixes as each new release is born.

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

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

Grouper News

Click here for a lower-level, highly-detailed version of the below:

Grouper v1.0

Release 1.0 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 Groups Registry
- Improved: query performance through caching

☐ Grouper-QuickStart v1.0 - tarball with documentation & instructions for cvs access.

Grouper v0.9

Release v0.9 offers the following new and/or different items from the v0.6 pre-release:

- 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

☐ Grouper-QuickStart v0.9 tarball with documentation & instructions for cvs access.
☐ Grouper API v0.9 tarball...
☐ Grouper UI v0.9 tarball...

Grouper v0.6

Grouper v0.6 is the initial UI 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 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 API v0.5.6

The API release v0.5.6 contains fixes to several bugs found in the 0.5.5 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)
- 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 API v0.5

The API release v0.5 is the first release of Grouper.

- info
- info
- info

☐ Grouper API v0.5 tarball with documentation & instructions for cvs access.

Questions or comments? Contact us.

News

This page last changed on Jun 07, 2006 by jbibbee@internet2.edu.

Grouper News

This document contains every change, addition, etc. that born with each release, and will continue to be updated.

$Id: NEWS,v 1.84 2005/12/19 19:11:18 blair Exp $

Grouper v0.9

Released: 19-Dec-2005

Changes:

- NEW: New public API.
- NEW: New query API. This includes the ability for sites to create their own custom query filters for use within the query API.
- NEW: New access and naming adapter APIs and interfaces.
- NEW: "all" subject that can be assigned memberships and granted privileges that maps to all subjects that are identifiable by Grouper's Subject API configuration.
- NEW: Event logging
- NEW: Configuration options for controlling what privileges, if any, are granted to the "all" subject whenever a new group or stem is created. By default, *READ* and *VIEW* are granted to "all" upon group creation and no privileges are granted to "all" upon stem creation.
- NEW: Subject IDs associated with *Member* objects within the Groups Registry can now be changed with the *Member.setSubjectId()* method.
- NEW: Internal source adapter for resolving the "root" (*GrouperSystem*) and "all" (*GrouperAll*) subjects. No configuration is necessary to use this adapter within Grouper.
- NEW: Experimental wheel-group support. This is disabled by default. If enabled, all members of the group are treated as root-like subjects with all access and naming privileges.
- UPDATE: New internals.
- UPDATE: ehcache-based caches for access and naming privilege resolution.
- UPDATE: Grouper source adapter no longer needs to be configured within *conf/sources.xml*.
- UPDATE: License changed to Apache License, Version 2.0

Grouper v0.6

Released: 16-Sep-2005

Changes:

- NEW: VIEW Access privilege (#339)
- NEW: READ Access privilege (#338)
- NEW: OPTIN Access privilege (#343)

- NEW: OPTOUT Access privilege (#344)
- NEW: *GrouperGroup.getDisplayExtension()* method
- NEW: *GrouperGroup.getDisplayName()* method
- NEW: *GrouperGroup.getExtension()* method
- NEW: *GrouperGroup.getMembers()* method
- NEW: *GrouperGroup.getName()* method
- NEW: *GrouperGroup.getStem()* method
- NEW: *GrouperMember.getMember()* method
- NEW: *GrouperMember.source()* method
- NEW: *GrouperQuery.base()* filter method
- NEW: *GrouperQuery.group()* filter method
- NEW: *GrouperQuery.group()* scoped filter method (#403)
- NEW: *GrouperQuery.groupAttr()* filter method
- NEW: *GrouperQuery.groupAttr()* scoped filter method (#403)
- NEW: *GrouperQuery.stem()* filter method
- NEW: *GrouperQuery.stem()* scoped filter method (#403)
- NEW: *GrouperQuery.stemAttr()* filter method
- NEW: *GrouperQuery.stemAttr()* scoped filter method (#403)
- NEW: *GrouperQuery.getGroups()* method
- NEW: *GrouperQuery.getListValues()* method
- NEW: *GrouperQuery.getMembers()* method
- NEW: *GrouperQuery.getStems()* method
- NEW: *GrouperStem.create(session, extension)* method
- NEW: *GrouperStem.getDisplayExtension()* method
- NEW: *GrouperStem.getDisplayName()* method
- NEW: *GrouperStem.getExtension()* method
- NEW: *GrouperStem.getMembers()* method
- NEW: *GrouperStem.getName()* method
- NEW: *GrouperStem.getRootStems()* method
- NEW: *GrouperStem.getStem()* method
- NEW: *GrouperStem.load(session, extension)* method
- NEW: PostgreSQL now a supported backend (#169) (Simon McLeish, London School of Economics)
- NEW: *displayName* is now automatically populated and maintained (#270)
- NEW: *Subject* and *SubjectAttribute* tables
- UPDATE: Grouper now uses version 0.1 draft 2 of the *Subject* API specification
- UPDATED: Reimplemented UPDATE Access privilege (#400)
- UPDATED: Reimplemented ADMIN Access privilege (#399)
- UPDATED: Reimplemented CREATE Naming privilege (#401)
- UPDATED: Reimplemented STEM Naming privilege (#402)
- UPDATE: *Grouper.hasSubjectType()* method replaced by *SubjectFactory.hasType()*
- UPDATE: *Grouper.subjectTypes()* method replaced by *SubjectFactory.types()*
- UPDATE: *GrouperSubject* class renamed to *SubjectFactory*
- UPDATE: *grouper_attribute.groupFieldValue* column size increased to 1024
- UPDATE: *grouper_member* table now includes a *subjectSource* column
- FIX: Root stems could be created with an *extension* containing the hierarchy delimiter.
- FIX: Stems and groups could have their *extension* and *isplayExtension* set to include the hierarchy delimiter after being created.
- FIX: *whoHas()* returns inaccurate results
- FIX: Error granting privileges to and revoking privileges from self (#426)
- FIX: *displayName* of parent stems not consulted when creating new groups and stems.
- FIX: VIEW, not READ, is needed to access a group or stem GUID. (#411)
- FIX: The subject that created a session did not have a properly initialized member object (#413)
- FIX: Privilege and effective membership pollution bug (#405)
- FIX: Default privilege implementations would attempt to revoke effective list values under select circumstances which would cause an exception to be thrown. (#361)

- FIX: *hasMember()* would return incorrect results when a member was both immediate and effective (#360)
- REMOVED: Stems can no longer be members or subjects
- REMOVED: *GrouperQuery.query()* method
- REMOVED: *Grouper.subjectType()* method
- REMOVED: *grouper_subjectType* table
- CONTRIB: *cvs2subject* now uses Grouper's *conf/hibernate.properties* for its JDBC configuration information.
- CONTRIB: *csv2subject* now supports additional input formats

Grouper v0.5.6

Released: 29-Apr-2005

Changes:

- Fixed: Effective memberships for non-"members" lists (e.g. access and naming privileges) were being calculated incorrectly (#350)
- Fixed: Non-root subjects could not create stems or groups (#353)
- Fixed: STEM, not ADMIN, needed to modify namespace attributes (#352)
- Fixed: Added NullGrouperAttribute class (which extends GrouperAttribute) to handle group attributes that either do not have values or have had their values deleted. (#356)
- Fixed: GrouperMember.load(session, subject) replaces GrouperMember.load(subject) (#348)
- Fixed: GrouperGroup.loadByID() now returns properly casted GrouperGroup objects (#349)
- Fixed: GrouperStem.loadByID() now returns properly casted GrouperStem objects (#349)

Grouper v0.5.5

Released: 15-Apr-2005

Changes:

- Updated: Hibernate internals completely rewritten to improve session and transaction handling
- Updated: New and more thoroughly tested implementation of the memberOf algorithm. In addition, we are now tracking the entire via chain for effective memberships.
- New: Improved Oracle compatibility and support
- New: Created logical distinction between groups (GrouperGroup) and namespaces (GrouperStem)
- New: GrouperGroup.hasMember() method for verifying whether a member belongs to a group (#328)
- New: GrouperMember.isMember() method for verifying whether a member belongs to a group (#328)
- New: GrouperStem.stems() method to retrieve immediate child namespaces within a namespace
- New: GrouperStem.groups() method to retrieve immediate child groups within a namespace
- New: GrouperSession objects now serializable (#296)
- New: More detailed reporting (via exceptions) of errors caused by various runtime error conditions
- New: Now using Commons DBCP by default for connection pooling
- New: Now using Hibernate named queries defined as defined in *conf/Grouper.hbm.xml*
- New: Use Ant's SQL task to to create, initialize and reset HSQLDB database (#287)
- Fixed: Removed calls to system.exit() (#179) (David Langenberg, The University Of Chicago)
- Fixed: sessionID generation bug (#278)

- Fixed: Effective membership bug with circular group memberships (#286)
- New: GrouperField objects now have public instance methods for getting information about the field
- Fixed: Effective membership now working with lists other than "members" as well as with the default Access and Naming privilege interfaces
- New: commons-dbcp 1.2.1 .jar file
- New: commons-pool 1.2 .jar file
- New: Hibernate's ODMG interface .jar
- Updated: Hibernate .jar from 2.1.7c to 2.1.8
- Updated: HSQLDB .jar from 1.7.1 to 1.7.2.11
- Updated csv2group to handle comments in input files and the -c and -q options

Incompabilities:

- Databases created with Grouper 0.5 will not work with this release.
- Runtime access to Access and Naming privilege implementations moved from *Grouper* to *GrouperSession*.
- Removed *GrouperSession* argument requirement for most instance method calls.

Known Bugs:

- Significant via chain duplication as reuse between list values is not working.
- Via chains are not deleted when they are no longer in use.
- Logging is not especially effective or useful.

Grouper v0.5

Released: Dec-2004

Changes:

- "Base" and "Naming" group types
- Immediate Memberships
- Effective Memberships
- Search: By immediate and effective membership
- Search: By group type
- Search: By group create time
- Search: By group modify time
- Access privilege interface
- Naming privilege interface
- An implementation of the access privilege interface that uses groups to manage privileges
- An implementation of the naming privilege interface that uses groups to manage privileges
- ADMIN and UPDATE access privileges
- CREATE and STEM naming privileges
- A partial implementation of the I2MI Subject interface for locally-defined people subjects
- A partial implementation of the I2MI Subject interface for groups as subjects
- Basic Event Logging
- Contributed: I2MI Subject Loader
- Contributed: Group Loader
- Contributed: Member Loader
- Contributed: Query Program

Known Bugs:

- Grouper does not fail gracefully or even necessarily informatively
- Insufficient data validation within code
- Not all database updates are atomic transactions (#233) (#248)
- Loading groups by *groupID* does not always fail cleanly.
- *GrouperQuery* objects do not properly reset their state (#255)
- *Modify* attributes *may* be be incorrect (#247)
- Session Handling is dubious at best (#173)
- Insufficient documentation
- Insufficient test coverage
- Slow

Questions or comments? Contact us.

This page last changed on Jun 06, 2006 by jbibbee@internet2.edu.

Grouper v1.0 Product Spec Sheet

Questions or comments? Contact us.

Component	Details
RDBMS	• DDL is provided for HSQLDB, Oracle 9i, and Postgresql 8. • Object persistence is provided by Hibernate v2.1.8, which in turn uses JDBC to connect with a back-end RDBMS.
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, except for debug mode used only by UI developers

UI Building & Configuration

This page last changed on Jun 09, 2006 by tbarton@uchicago.edu.

Note: This section is under construction.

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 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
subject.default.attribute	The default attribute used to identify any subject. Might be superceded by other configuration declarations.	Any subject attribute common to all subjects presented by source adapters. The minimum set available by default is name, description, and subjectId.
group.default.attribute	The default attribute used to identify any group. Might be superceded by other configuration declarations.	Any group naming attribute: name, displayName, extension, displayExtension, id.

stem.default.attribute	The default attribute used to identify any stem. Might be superceded by other configuration declarations.	Any stem naming attribute: name, displayName, extension, displayExtension.
search.group.result-field	The name of the group naming attribute displayed in search results.	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.

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 / customise the UI. See Grouper UI Development Environment for options.

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

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 datasources.

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.

UI Customization Guide

This page last changed on Jun 07, 2006 by jbibbee@internet2.edu.

Document	Desription
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

This page last changed on Jun 08, 2006 by ghbrett.

Welcome to the Grouper Working Group Wiki!

Feel free to use this space for all Grouper-related activities that go beyond mailing list discussions. Want to post anonymously?

Note: *Open viewing. Editing is restricted to Grouper Working Group members only, thank you.*

Questions or comments? ***Contact us***.

Extended Discussion

... When the mailing list cramps your style.

Presentations, etc.

... Attach your WG-related presentations or proposals, etc.

Contributions

... Do you have something to share? Code, documentation, or use cases?

This page last changed on Jun 08, 2006 by tbarton@uchicago.edu.

Contributions

Use this page to post and browse community contributions.

Software, Code, Documentation, etc.

Please attach related materials and provide links appropriately.

Note: Not all of the below contributions will be verified by the Grouper development team.

Date	Contribution	Documentation	Contributor	Contributor Home	Notes	Grouper Verified

Use Cases

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

Questions or comments? Contact us.

Extended Discussion

This page last changed on May 26, 2006 by jbibbee@internet2.edu.

Need a place to communicate in the wiki?

Add your own page and link to it from here, or add a comment below.

Presentations & Documents

This page last changed on Jun 05, 2006 by jbibbee@internet2.edu.

WG Member Presentations & Documents

1. Attach your Presentation or Document.
2. Insert a row at the top (newer items at top, older items at bottom).
3. Add your information!
4. *Note: You may link directly to an attachment of this page.*

Presentations

	Presentation Title	Author / Editor	Author Home	Venue	Date	.HTML	.PDF	.PPT	.ODP	etc.
	Grouper/Signet	Lynn McRae	Stanford U.	Spring 2006 Internet2 Member Meeting; Alexandria, VA	26-Apr-2006	-	-	ppt	-	-
	Use Cases	Lynn McRae	Stanford U.	Spring 2006 Internet2 Member Meeting; Alexandria, VA	24-Apr-2006	-	-	ppt	-	-
	Use Cases Wrap up	Tom Barton	U. Chicago	Signet/Grouper Early Adopters Workshop; USC, LA, CA	22-Mar-2006	-	-	ppt	-	-
	Authorization Tools - I2/NMI Update: Signet, Grouper, & GridShib	Tom Barton	U. Chicago		Feb-2005	-	-	ppt	-	-
	Authority Process &	Sandra Senti	Stanford U.	NMI Advanced CAMP	9-Jul-2003	html	pdf	-	-	-

	Policy									

Documents

	Document Title	Author	Author Home	Venue	Notes	Date	.HTML	.PDF	.TXT
	Group Tools Architecture	Tom Barton	U. Chicago	MACE-Dir		Apr-2005	html	-	-
	LDAP representation of membership in groups (aka duMember)	Keith Hazelton	U. Wisconsin	MACE-Dir		5-Jul-2005	html	-	-
	Group and membership concepts	Keith Hazelton	U. Wisconsin	MACE-Dir		5-Jul-2005	html	-	-
	Grouper Phase 1 Specifications (Draft)	Tom Barton, Blair Christensen	U. Chicago	MACE-Dir-Grouper		3-May-2004	html	-	-
	Grouper API Search & Browse Capabilities	Tom Barton	U. Chicago	MACE-Dir-Groups		4-Nov-2004	html	-	-
	Grouper Roadmap	Tom Barton	U. Chicago	MACE-Dir-Groups		24-Feb-2004	html	-	-
	Strawman architecture for group Tools	Tom Barton	U. Chicago	MACE-Dir-Groups		8-Oct-2003	html	-	-
	SAGE (Service for Authorized Group Editing) (Draft)	Tom Barton	U. Chicago	MACE-Dir-Groups		24-Apr-2003	html	-	-

	Practices in Directory Groups	Tom Barton	U. Memphis	MACE-Dir	-Groups	Oct-2002	htm	-	-
	Inter-Domain Data Exchange	Rob Bai nz	UMB	MACE-Dir		Oct-2002	html	-	-
	Groups Implementers Guide	Eileen Shepard	Boston College	MACE-Dir	-Groups	6-June-2002	html	-	-
	MACE-Dir's Approach to Support of Authorization & Messaging by Core Middleware: Discussion Starter	Tom Barton	U. Memphis	MACE-Dir	-Groups	18-Jul-2001	html	-	-
	Directory Server Groups (People)	Todd Piket	MTU	MACE-Dir	-Groups	17-Jul-2001	html	-	-

Grouper Web & Wiki Outline

This page last changed on May 26, 2006 by jbibbee@internet2.edu.

Document	Notes	on wiki	on web	Who	Rev Date	Done?	finish Date - Priority
home.html				Jess	20-Apr		
about.html				Jess	20-Apr		
news.html				SteveO/Jess			
faq.html				Jess +			
license.html				Jess	20-Apr		
spec_sheet.html		link to web	only;low	Tom/?			
documents.html		only;low	link to wiki	Jess			
software.html	main software page; downloads here	link to web	only;high	Jess	5-May		
• Overview concepts.html - Feature list, etc.				Tom / All?			
• glossary.html		low	high	Tom/Jess	5-May		
• Use Cases	Also under Campus Mngt (wiki)	low	high?	Tom will kickstart			
• demo.html	describes demo; screenshots			Gary/?			
• details.html	Release Details & Previous Releases (see Signet example off Software web page)						

Campus Management		low	high				
• Groups Management & Your Institution?	Impl. Guide - section?			Tom/Jessica			
• Supporting Your Campus	anything existing?			Tom will kickstart			
• Design Guidelines				Tom will kickstart			
• Use Cases	Also under software.html (web)	low	high?				
Systems Integrator	section 7.3 from Impl. Guide? parse into below equivalents	low					
• Doc D1 - Grouper Deployment Overview	Basically the level-set doc that the workshop attendees wished we'd given them. This doc should also include details on configuring Subject API Source Adapters. Those parts of the Grouper Implementation Guide that we'll	low	na	Tom & Jessica			

	preserve (see below) will need some modifying when this doc is written.						
• install_deploy.html							
• running_support.html							
• extend_integrate.html							
• CVS		na					
• Bugzilla		na					
Applications Developer		low					
• Applications using the Grouper API	delete this idea for now			Tom will kickstart	26-May	na	
• Doc D2	Developer's Guide to the API.			Blair			low: post v1.0?
• User Interface Customization UI docs...				Gary			
• JAVAdoc?				Blair/?			
contact.html				Jess	20-Apr		

To Dos

This page last changed on May 24, 2006 by jbibbee@internet2.edu.

Specific To Dos - code, terminology, packaging, etc.

- make a common quickstart database for grouper + signet
- should we bundle tomcat with grouper-QS?
- Change occurances of "effective" to "indirect" membership
- Change occurances of "namespaces" and of "stems" to "naming stems". Maybe keep the "stem" term around as a sort of short name for "naming stem".

General To Dos - approach, etc.

- Combined approach to grouper + signet documentation, or maybe just overview, glossary, that sort of thing.
    - may require creating a whole new structure for the combined doc ... ala tool harmonization

Completed To Dos:

- Jess - the "concepts, glossary, and features" page should hvae the draft glossary, something akin to the Impl Guide's section 2, and it better explain what the heck is a naming stem and all that jazz.