# Grouper external subjects

This topic is discussed in the Lite UI - External Users training video.

Grouper has support for external or federated users (invitation, registry, self-serve attributes, etc). In order for Grouper to be used in a Federated environment, external subjects must be able to be fed into a subject source. In this case federated or external is similar, but the intent is that the subject id would be an eppn (whether or not the IdP's are in a federation). However, the subject id's do not have to be an eppn, but they do need to be unique in the source. This should be handled by an Identity management system, though generally it is not. So to prevent many institutions from having to create their own custom process, Grouper has a built-in that can be used. Note, a custom or IdM solution could be used instead, or it could be disabled entirely. COmanage has a solution for this, and this solution is similar to theirs.

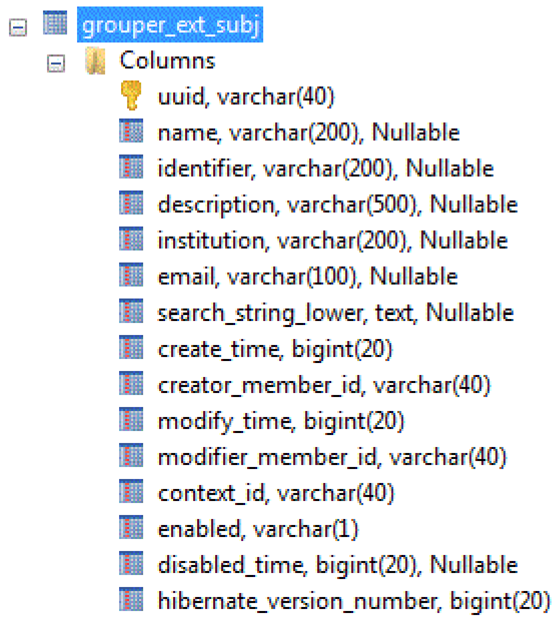
The enhancement to Grouper consists of a few DB tables, and some UI screens. Users will self register (perhaps from an invite).

Admins can just enter the subjectId of external users. This might be rare since it is difficult to know what the user id will be until they login to the UI screen.

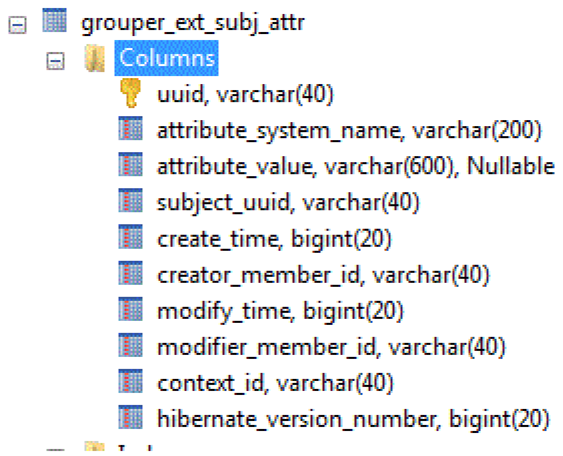
- Grouper external subjects on new UI
- Grouper external users on demo server

### Data model

Grouper has two tables similar to the quickstart subject tables which are:



```
grouper_ext_subj
  Columns
    uuid, varchar(40)
    name, varchar(200), Nullable
    identifier, varchar(200), Nullable
    description, varchar(500), Nullable
    institution, varchar(200), Nullable
    email, varchar(100), Nullable
    search_string_lower, text, Nullable
    create_time, bigint(20)
    creator_member_id, varchar(40)
    modify_time, bigint(20)
    modifier_member_id, varchar(40)
    context_id, varchar(40)
    enabled, varchar(1)
    disabled_time, bigint(20), Nullable
    hibernate_version_number, bigint(20)
```



```
grouper_ext_subj_attr
  Columns
    uuid, varchar(40)
    attribute_system_name, varchar(200)
    attribute_value, varchar(600), Nullable
    subject_uuid, varchar(40)
    create_time, bigint(20)
    creator_member_id, varchar(40)
    modify_time, bigint(20)
    modifier_member_id, varchar(40)
    context_id, varchar(40)
    hibernate_version_number, bigint(20)
```

## Configuration

grouper.properties

```
#####################################
## External subjects
#####################################


externalSubjects.desc.el = ${grouperUtil.appendIfNotBlankString(externalSubject.name, ' - ', externalSubject.
institution)}
# true if the description should be managed via EL (config above)
externalSubjects.desc.manual = false

# quartz cron where subjects are recalculated if necessary (empty means dont run), e.g. everyday at 3am
externalSubjects.calc.fields.cron = 0 0 3 * * ?

externalSubjects.name.required = true
externalSubjects.email.required = false
externalSubjects.email.enabled = true

# these field names (uuid, institution, identifier, uuid, email, name) or attribute names
# will be toLowered, and appended with comma separators
externalSubjects.searchStringFields = name, institution, identifier, uuid, email, jabber

externalSubjects.institution.required = false
externalSubjects.institution.enabled = true

# note, this must be only alphanumeric lower case or underscore
# (valid db column name, subject attribute name)
externalSubjects.attributes.jabber.systemName = jabber
externalSubjects.attributes.jabber.required = false
# comment on column in DB (no special characters allowed)
externalSubjects.attributes.jabber.comment = The jabber ID of the user

# if wheel or root can edit external users
externalSubjects.wheelOrRootCanEdit = true

# group which is allowed to edit external users
externalSubjects.groupAllowedForEdit =

# if the view on the external subjects should be created.
# turn this off if it doesnt compile, othrewise should be fine
externalSubjects.createView = true
```

```
#name of external subject source, defaults to grouperExternal
externalSubject.sourceName = grouperExternal

# grouper can auto create a jdbc2 source for the external subjects
externalSubjects.autoCreateSource = false

# put in fully qualified classes to add to the EL context.  Note that they need a default constructor
# comma separated.  The alias will be the simple class name without a first cap.
# e.g. if the class is test.Test the alias is "test"
externalSubjects.customElClasses =

# change these to affect the storage where external subjects live (e.g. to store in ldap),
# must implement each respective storable interface
externalSubjects.storage.ExternalSubjectStorable.class = edu.internet2.middleware.grouper.externalSubjects.
ExternalSubjectDbStorage
externalSubjects.storage.ExternalSubjectAttributeStorable.class = edu.internet2.middleware.grouper.
externalSubjects.ExternalSubjectAttributeDbStorage

# you can use the variables $newline$, $inviteLink$.  Note, you need to change this default message...
externalSubjectsInviteDefaultEmail = Hello,$newline$$newline$This is an invitation to register at our site to
be able to access our applications.  This invitation expires in 7 days.  Click on the link below and sign in
with your InCommon credentials.  If you do not have InCommon credentials you can register at a site like
protectnetwork.com and use those credentials.$newline$$newline$$inviteLink$$newline$$newline$Regards.

# default subject for email
externalSubjectsInviteDefaultEmailSubject = Register to access applications

# numner of days after which this request will expire.  If -1, then will not expire
externalSubjectsInviteExpireAfterDays = 7

#put some group names comma separated for groups to auto add subjects to
externalSubjects.autoaddGroups=
#should be insert, or update, or insert,update
externalSubjects.autoaddGroupActions=insert,update
#if a number is here, expire the group assignment after a certain number of days
externalSubjects.autoaddGroupExpireAfterDays=

#add multiple group assignment actions by URL param: externalSubjectInviteName
externalSubjects.autoadd.testingLibrary.externalSubjectInviteName=library
#comma separated groups to add for this type of invite
externalSubjects.autoadd.testingLibrary.groups=
#should be insert, update, or insert,update
externalSubjects.autoadd.testingLibrary.actions=insert,update
#should be insert, update, or insert,update
externalSubjects.autoadd.testingLibrary.expireAfterDays=

#if registrations are only allowed if invited or existing...
externalSubjects.registerRequiresInvite=true

#make sure the identifier when logging in is like an email address or eppn, e.g. username@school.edu
externalSubjects.validateIndentiferLikeEmail=true

#put regexes here, increment the 0 for multiple entries, e.g. restrict your own institution
#note, the extensions must be sequential (dont skip), regex e.g. ^.*@myschool.edu$
externalSubjects.regexForInvalidIdentifier.0=
```

## Built in field metadata

The built in subject fields can be enabled/disabled or required or not.  Things like email are in the subject table since they might be common in deployments.

```
externalSubjects.name.required = true
externalSubjects.email.required = false
externalSubjects.email.enabled = true
externalSubjects.institution.required = false
externalSubjects.institution.enabled = true
```

### External subject attributes

Grouper allows configuration of which external subject attributes to keep for all external users.  E.g. phone, email, jabber, firstName, lastName, etc.  This is in the grouper.properties. You can configure the friendly name, comment, if required, etc

```
# this can change, and is shown on screen
externalSubjects.attributes.jabber.friendlyName = Jabber ID
# note, this must be only alphanumeric lower case or underscore
# (valid db column name, subject attribute name)
externalSubjects.attributes.jabber.systemName = jabber
externalSubjects.attributes.jabber.required = false
# comment on column in DB (no special characters allowed)
externalSubjects.attributes.jabber.comment = The jabber ID of the user
```

### Auto provisioning into groups

When someone logs in, they can be auto provisioned into groups.  You can pass parameters in the URL to change which groups are assigned

Auto provision into groups no matter the URL

```
#put some group names comma separated for groups to auto add subjects to
externalSubjects.autoaddGroups=someStem:someGroup
#should be insert, or update, or insert,update
externalSubjects.autoaddGroupActions=insert,update
#if a number is here, expire the group assignment after a certain number of days
externalSubjects.autoaddGroupExpireAfterDays=
```

If you want to add to a group based on URL, use these configs (you can add multiple):

```
#add multiple group assignment actions by URL param: externalSubjectInviteName
externalSubjects.autoadd.testingLibrary.externalSubjectInviteName=library
#comma separated groups to add for this type of invite
externalSubjects.autoadd.testingLibrary.groups=someStem:libraryUsers
#should be insert, update, or insert,update
externalSubjects.autoadd.testingLibrary.actions=insert,update
#should be insert, update, or insert,update
externalSubjects.autoadd.testingLibrary.expireAfterDays=365

#add multiple group assignment actions by URL param: externalSubjectInviteName
externalSubjects.autoadd.testingWhatever.externalSubjectInviteName=otherApplication
#comma separated groups to add for this type of invite
externalSubjects.autoadd.testingWhatever.groups=someStem:otherApplication
#should be insert, update, or insert,update
externalSubjects.autoadd.testingWhatever.actions=insert,update
#should be insert, update, or insert,update
externalSubjects.autoadd.testingWhatever.expireAfterDays=
```

Based on these configs, these are the URLs for users to use:

https://server.school.edu/grouper/grouperExternal/appHtml/grouper.html?operation=ExternalSubjectSelfRegister.
externalSubjectSelfRegister&externalSubjectInviteName=library

-or-

https://server.school.edu/grouper/grouperExternal/appHtml/grouper.html?operation=ExternalSubjectSelfRegister.
externalSubjectSelfRegister&externalSubjectInviteName=otherApplication

## Validate the identifier

If you are using shib, you can make sure the identifier is in the format a@b.c, and you can also filter out some regexes (e.g. filter out from your own institution by suffix).

```
#make sure the identifier when logging in is like an email address or eppn, e.g. username@school.edu
externalSubjects.validateIndentiferLikeEmail=true

#put regexes here, increment the 0 for multiple entries, e.g. restrict your own institution
#note, the extensions must be sequential (dont skip), regex e.g. ^.*@myschool.edu$
externalSubjects.regexForInvalidIdentifier.0=
```

Note, if you filter out your own institution, if someone gets an invite, then notifications about registration will still be sent, and the user will still be provisioned into groups based on their local ID.  This is assuming that the external identifier (e.g. user@myschool.edu) is an identifier in the local subject source...

## Pluggability

This uses the Grouper UI pluggable authenticator so that Shib or non-Shib authentication would work or would be pluggable.

The storage is pluggable also so someone could use a different storage e.g. ldap.   Implement the external subject storable interfaces (one for subject, one for attribute).  The built in implementation just call the Grouper DAO to store in the Grouper DB

```
# change these to affect the storage where external subjects live (e.g. to store in ldap),
# must implement each respective storable interface
externalSubjects.storage.ExternalSubjectStorable.class = edu.internet2.middleware.grouper.externalSubjects.
ExternalSubjectDbStorage
externalSubjects.storage.ExternalSubjectAttributeStorable.class = edu.internet2.middleware.grouper.
externalSubjects.ExternalSubjectAttributeDbStorage
```

Here are examples of the interfaces:

```
package edu.internet2.middleware.grouper.externalSubjects;

import java.util.Set;

import edu.internet2.middleware.grouper.internal.dao.QueryOptions;

/**
 * implement this to change how external subjects are stored
 * @author mchyzer
 */
public interface ExternalSubjectStorable {

  /**
   * find all external subjects which have a disabled date which are not disabled
   * @return the set of subjects
   */
  public Set<ExternalSubject> findAllDisabledMismatch();

  /**
   * find all external subjects
   * @return the set of subjects
   */
  public Set<ExternalSubject> findAll();

  /**
   * find an external subject by identifier
   * @param identifier
   * @param exceptionIfNotFound
   * @param queryOptions
   * @return the external subject or null or exception
   */
  ExternalSubject findByIdentifier(String identifier, boolean exceptionIfNotFound, QueryOptions queryOptions);

  /**
   * delete an external subject and all its attributes
   * @param externalSubject
   */
  void delete(ExternalSubject externalSubject);

  /**
   * insert or update an external subject to the DB
   * @param externalSubject
   */
  void saveOrUpdate( ExternalSubject externalSubject );

}
```

```
package edu.internet2.middleware.grouper.externalSubjects;

import java.util.Set;

import edu.internet2.middleware.grouper.internal.dao.QueryOptions;

/**
 * interface to implement to keep external subjects somewhere besides in the Grouper DB
 * @author mchyzer
 */
public interface ExternalSubjectAttributeStorable {
  /**
   * delete an external subject and all its attributes
   * @param externalSubjectAttribute
   */
  void delete(ExternalSubjectAttribute externalSubjectAttribute);

  /**
   * insert or update an external subject attribute to the DB
   * @param externalSubjectAttribute
   */
  void saveOrUpdate( ExternalSubjectAttribute externalSubjectAttribute );

  /**
   * find an external subject attribute by identifier
   * @param uuid
   * @param exceptionIfNotFound
   * @param queryOptions
   * @return the external subject or null or exception
   */
  ExternalSubjectAttribute findByUuid(String uuid, boolean exceptionIfNotFound, QueryOptions queryOptions);

  /**
   * find attributes by subject, order by system name
   * @param subjectUuid
   * @param queryOptions
   * @return the external subject or null or exception
   */
  Set<ExternalSubjectAttribute> findBySubject(String subjectUuid, QueryOptions queryOptions);

}
```

### Calculated fields

Some fields are or can be calculated.  The search string (string where subject searches are based on) is based on certain fields, and the description can be based on expression language.  These calculations will be recalculated whenever a subject is changed (or attributes), or when the daemon runs.

```
externalSubjects.desc.el = ${grouperUtil.appendIfNotBlankString(externalSubject.name, ' - ', externalSubject.
institution)}
# true if the description should be managed via EL (config above)
externalSubjects.desc.manual = false

# these field names (uuid, institution, identifier, uuid, email, name) or attribute names
# will be toLowered, and appended with comma separators
externalSubjects.searchStringFields = name, institution, identifier, uuid, email, jabber
```

For the expression language, you can add custom classes

```
# put in fully qualified classes to add to the EL context.  Note that they need a default constructor
# comma separated.  The alias will be the simple class name without a first cap.
# e.g. if the class is test.Test the alias is "test"
externalSubjects.customElClasses =
```

### Built in subject source

Grouper can auto create a jdbc subject source 2 view for the subjects and attributes, and it can auto create a source for that view.  Or you can disabled that and do it yourself.

```
# if the view on the external subjects should be created.
# turn this off if it doesnt compile, othrewise should be fine
externalSubjects.createView = true

# grouper can auto create a jdbc2 source for the external subjects
externalSubjects.autoCreateSource = true
```

The view will look like this (depending on which fields and attributes are enabled/included):

```
CREATE VIEW grouper_ext_subj_v (uuid, name, identifier, description, institution, email, search_string_lower,
jabber)
AS SELECT ges.uuid, ges.name, ges.identifier, ges.description , ges.institution , ges.email , ges.
search_string_lower ,
(SELECT gesa.attribute_value FROM grouper_ext_subj_attr gesa WHERE gesa.subject_uuid = ges.uuid AND gesa.
attribute_system_name = 'jabber' ) AS jabber
FROM grouper_ext_subj ges WHERE ges.enabled = 'T';
```

The auto created source does not need anything in the sources.xml and will map to the above view with the jdbc2 source adapter which allows for more intelligent searching (put in a phrase, and any string in the phrase.cna be anywhere in the search string will match a subject)

### External user security

Wheel or root can be enabled to edit external subjects, or a group can be configured:

```
# if wheel or root can edit external users
externalSubjects.wheelOrRootCanEdit = true

# group which is allowed to edit external users
externalSubjects.groupAllowedForEdit =
```

### Enabled/disabled subjects

External subjects can have expire dates.  These dates work like membership delete dates where the disabledDate daemon will check for deltas and update the enabled flag.

### Calculated fields daemon

There is a daemon which runs in the grouper-loader on a cron which will recalculate the calculated fields (e.g. nightly).  This needs to be done if the configuration changes, or if data changes without recalculating (not sure why this would happen).  Just set the quartz cron in the grouper.properties

```
# quartz cron where subjects are recalculated if necessary (empty means dont run), e.g. everyday at 3am
externalSubjects.calc.fields.cron = 0 0 3 * * ?
```

If you want to kick off the recalc manually (e.g. if you change how subjects look), you can do that in GSH:

```
gsh 0% grouperSession = GrouperSession.startRootSession();
gsh 1% ExternalSubject.internal_daemonCalcFields();
```

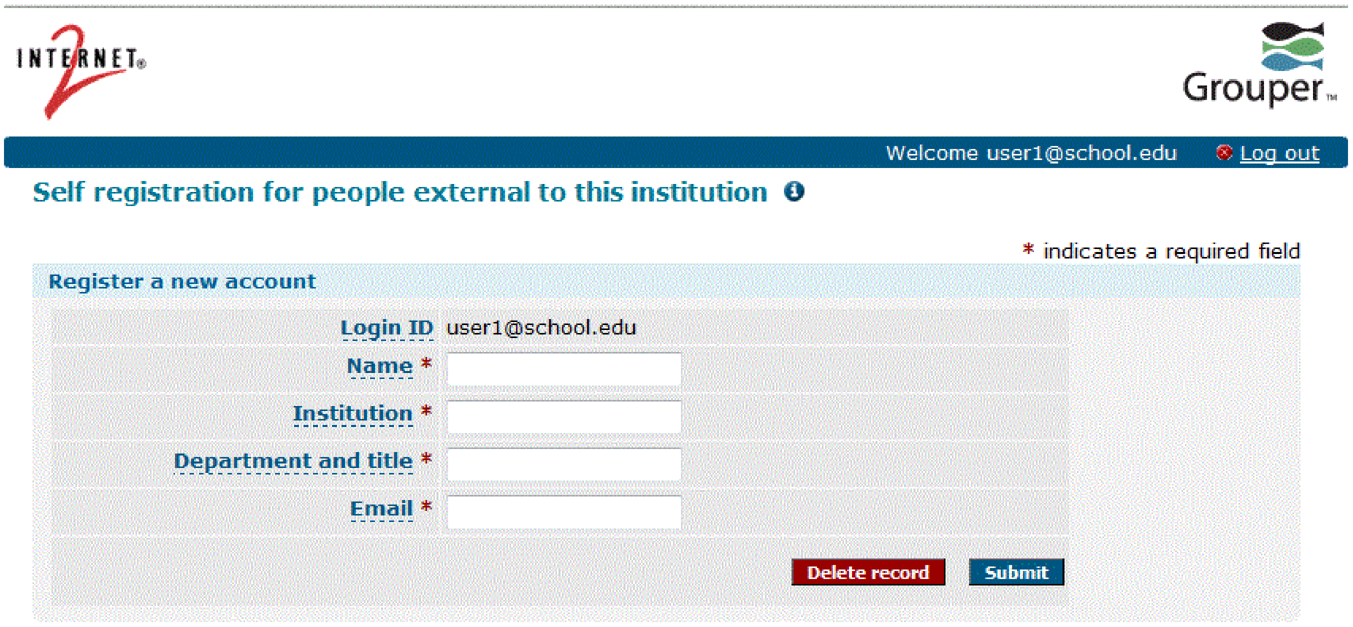
### GSH commands

These commands can create, edit, and delete external subjects

```
gsh 0% grouperSession = GrouperSession.startRootSession();
edu.internet2.middleware.grouper.GrouperSession: 552eb161e98f47c4b98876528e36a409,'GrouperSystem','application'

//create a new external subject
gsh 1% externalSubject = new ExternalSubject();
edu.internet2.middleware.grouper.externalSubjects.ExternalSubject:
gsh 2% externalSubject.setIdentifier("abcd@school.edu");
gsh 3% externalSubject.setInstitution("My Institution");
gsh 4% externalSubject.setName("My Name");
gsh 5% externalSubject.setEmail("a@b.c");
gsh 6% externalSubject.store();

//assign an attribute
gsh 7% externalSubject.assignAttribute("jabber", "e@r.t");
true

//find the object to edit it
gsh 8% externalSubject = GrouperDAOFactory.getFactory().getExternalSubject().findByIdentifier("abcd@school.
edu", true, null);
edu.internet2.middleware.grouper.externalSubjects.ExternalSubject: uuid: 759cf0f0b6874cef886a4f3138244125,
identifier: abcd@school.edu, name: My Name, description: My Name - My Institution,

//search by subject in grouper by the subject api
gsh 9% subject = SubjectFinder.findByIdentifier("abcd@school.edu", true);
subject: id='759cf0f0b6874cef886a4f3138244125' type='person' source='grouperExternal' name='My Name'
gsh 10% subject.getName();
My Name

//edit the external subject
gsh 11% externalSubject.setName("My Name2");
gsh 12% externalSubject.store();

//find by any substring of the subject
gsh 14% subject = SubjectFinder.findAll("naMe2 mY INSTITUTION").iterator().next();
subject: id='759cf0f0b6874cef886a4f3138244125' type='person' source='grouperExternal' name='My Name2'

//note that all attributes are exposed by the subject api
gsh 15% subject.getAttributeValue("jabber");
e@r.t

//delete the external subject
gsh 16% externalSubject.delete();

//note its not there
gsh 17% subject = SubjectFinder.findByIdentifier("abcd@school.edu", false);

//recalculate descriptions (e.g. if the format has changed)
gsh 18% ExternalSubject.internal_daemonCalcFields();
24
gsh 19%
```

### Self service screen

The fields below can be customized per institution, as well as the text, look and feel, etc.  Some applications might require a lot of user data, and others do not need as much data about the user.  It would be nice to have a lot of data, e.g. so the application can use the data (e.g. email address), and so we can have descriptive person pickers, though it is a little risky since the data is user entered and unvetted.


Welcome user1@school.edu Log out

Self registration for people external to this institution

* indicates a required field

Register a new account

| | |
|---|---|
| Login ID | user1@school.edu |
| Name * | |
| Institution * | |
| Department and title * | |
| Email * | |

### URLs and servlets

There is a new external servlet so that external users can be protected by Shib (or whatever), and the rest of the UI can be protected by a local singl sign on system.  Or both.  Or you could run the UI twice.  The URL of the external part is e.g.

```
http://localhost:8090/grouper/grouperExternal/appHtml/grouper.html?operation=ExternalSubjectSelfRegister.
externalSubjectSelfRegister
```

All the ajax is in grouperExternal.  The admin UI is all struts do's and jsp's.  The lite UI is in grouper/grouperUi/etc.  You cannot access the admin console, or lite UI from the external URL and vise versa.  Note there is a new servlet mapping and filter mapping in the web.xml

### Admin console

There could be an admin screen where certain groups of users could add/edit users.  There could be a delete screen to delete users.  There could also be a time to live on users for admins to set.

### How to change your login ID

For example, if a user is not a member of the federation, and signs up at protect network, uses services, and then later on their home institution becomes a member of incommon, a Grouper admin could edit the "identifier" attribute of the user via GSH or SQL.  This will keep the memberships of the user intact, but allow them to log in with the new ID.

### Allowed to register?

This paramter affects if a user is allowed to register their information

From grouper.properties

```
#if registrations are only allowed if invited or existing...
externalSubjects.registerRequiresInvite=true
```

| Invite required? | Invite UUID in URL? | User previously registered (edit not insert)? | Error message? | Result |
|---|---|---|---|---|
| F | F | F | F | Continue to screen |
| F | F | T | F | Continue to screen |
| F | Invalid UUID | F | T | Continue to screen |
| F | Invalid UUID | T | T | Continue to screen |
| F | Valid UUID | F | F | Continue to screen |
| F | Valid UUID | T | F | Continue to screen |
| T | F | F | T | Do NOT continue to screen |
| T | F | T | T | Continue to screen |
| T | Invalid UUID | F | T | Do NOT continue to screen |
| T | Invalid UUID | T | T | Continue to screen |
| T | Valid UUID | F | F | Continue to screen |
| T | Valid UUID | T | F | Continue to screen |

sda

## External users hook

You can have custom java logic executed when someone registers or edits their information.

Configure this in the grouper.properties:

```
#implement an external subject hook by extending edu.internet2.middleware.grouper.hooks.ExternalSubjectHooks
#hooks.externalSubject.class=edu.yourSchool.it.YourSchoolExternalSubjectHooks
```

The class you extend looks like this:

```
public abstract class ExternalSubjectHooks {

  //*****  START GENERATED WITH GenerateMethodConstants.java *****//

  /** constant for method name for: postEditExternalSubject */
  public static final String METHOD_POST_EDIT_EXTERNAL_SUBJECT = "postEditExternalSubject";


  //*****  END GENERATED WITH GenerateMethodConstants.java *****//
  /**
   * called right after an edit of external subject (same transaction)
   * @param hooksContext
   * @param editBean
   */
  public void postEditExternalSubject(HooksContext hooksContext, HooksExternalSubjectBean editBean) {

  }
}
```

Override the postEditExternalSubject method and do whatever you want...

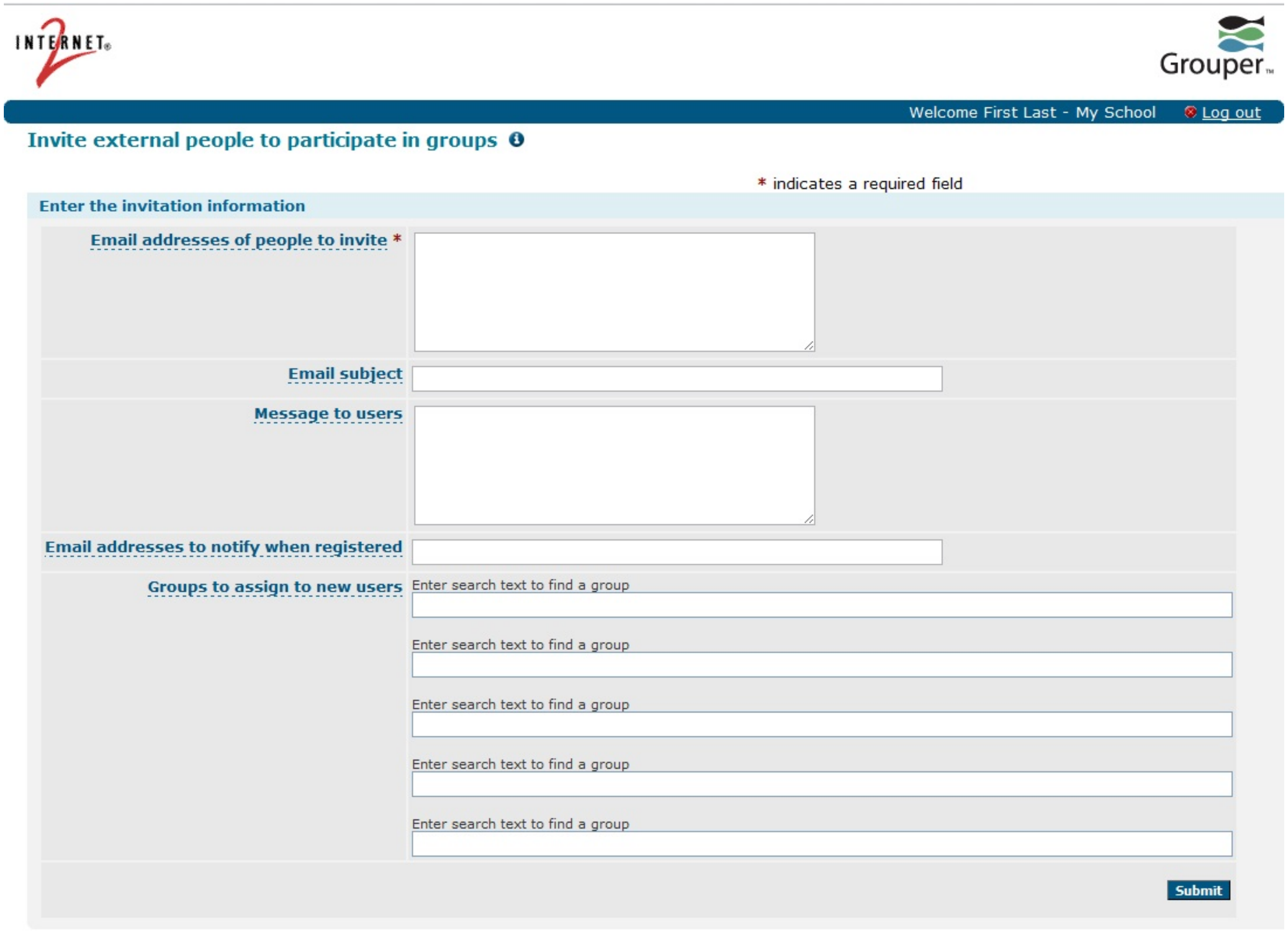
## Invites with group provisioning (New UI)

Starting with Grouper v2.4, External Subjects and Invites are handled by default in the New UI, and the Lite UI is no longer available in the standard setup.

## Invites with group provisioning (Lite UI)

A picker to allow the person inviting the external subjects to mark them to be added to group(s) once they register.  Note the security (inviter needs UPDATE on groups) will be checked at the invite time, and provisioning time, there is no actAs once the users register.  This information should be put in the email to the inviter (if applicable).  At some point we could do something similar for permissions as well.  Note when an invite email is clicked on, all pending invites for that email address are processed...

Here is the invite screen



Welcome First Last - My School Log out

Invite external people to participate in groups

* indicates a required field

Enter the invitation information

| | |
|---|---|
| Email addresses of people to invite * | |
| Email subject | |
| Message to users | |
| Email addresses to notify when registered | |
| Groups to assign to new users | Enter search text to find a group<br>Enter search text to find a group<br>Enter search text to find a group<br>Enter search text to find a group<br>Enter search text to find a group |

Submit

You can restrict who has access to the invite screen with this setting in the media.properties (blank means everyone has access):

```
#users must be in this group to invite external users to grouper
require.group.for.inviteExternalSubjects.logins=etc:externalSubjectInviters
```

You can allow wheel groups to have invites (normally this is not allowed)

```
# if the wheel group is allowed to be invited
inviteExternalMembers.allowWheelInInvite = false
```

You can enable or disable the invite or registration screen (default is to not be enabled):

```
# if the registration screen is enabled
externalMembers.enabledRegistration = false

# if the invitation screen is enabled
inviteExternalMembers.enableInvitation = false
```

### Email to invitee

When someone invites an external user, an email is sent to each use invited in the "email addresses of people to invite" field.

The default template is in grouper.properties if an email message or subject is not specified in the invite:

```
# you can use the variables $newline$, $inviteLink$.  Note, you need to change this default message...
externalSubjectsInviteDefaultEmail = Hello,$newline$$newline$This is an invitation to register at our site to
be able to access our applications.  This invitation expires in 7 days.  Click on the link below and sign in
with your InCommon credentials.  If you do not have InCommon credentials you can register at a site like
protectnetwork.org and use those credentials.$newline$$newline$$inviteLink$$newline$$newline$Regards.
# default subject for email
externalSubjectsInviteDefaultEmailSubject = Register to access applications
```

If someone typed in a subject, that will be used, if someone types in an email in the invite screen, then that will be used, but the link to register is appended to it (needs to be generated since there is a UUID in it)

An example of this email looks like this:

```
From: "groupersystem@gmail.com" <groupersystem@gmail.com>    <-- this is configurable in grouper.properties
To: someuser@yahoo.com
Sent: Sun, November 28, 2010 10:42:21 AM
Subject: TEST:Register to access applications

Hello,

This is an invitation to register at our site to be able to access our applications.  This invitation expires
in 7 days.  Click on the link below and sign in with your InCommon credentials.  If you do not have InCommon
credentials you can register at a site like protectnetwork.org and use those credentials.

https://server.school.edu/grouper/grouperExternal/appHtml/grouper.html?operation=ExternalSubjectSelfRegister.
externalSubjectSelfRegister&externalSubjectInviteId=55a61c064d62499d97650866824693cc

Regards.
```

### Email addresses to notify when registered

If there are email addresses filled in to the invite screen, then people can be notified when people register (one email sent as each person registers).  The email format is specified in the grouper.properties:

```
# you can use the variables $newline$, $inviteeIdentifier$, $inviteeEmailAddress$.  Note, you need to change
this default message...
externalSubjectsNotifyInviterEmail = Hello,$newline$$newline$This is a notification that user
$inviteeIdentifier$ from email address $inviteeEmailAddress$ has registered with the identity management
service.  They can now use applications at this institution.$newline$$newline$Regards.
externalSubjectsNotifyInviterSubject = $inviteeIdentifier$ has registered
```

The email to the those people looks like this (depending on the template):

```
From: "groupersystem@gmail.com" <groupersystem@gmail.com>    <-- note, this is configurable in grouper.
properties
To: someone@someschool.edu
Sent: Sun, November 28, 2010 10:01:31 AM
Subject: TEST:user1@school.edu has registered                <-- in non prod env's a prefix can be specified in
grouper.properties

Hello,

This is a notification that user user1@school.edu from email address person@yahoo.com has registered with the
identity management service.  They can now use applications at this institution.

Regards.
```

sda

### Invite by entering in user id's (New UI)

Starting with Grouper v2.4, External Subjects and Invites are handled by default in the New UI, and the Lite UI is no longer available in the standard setup.

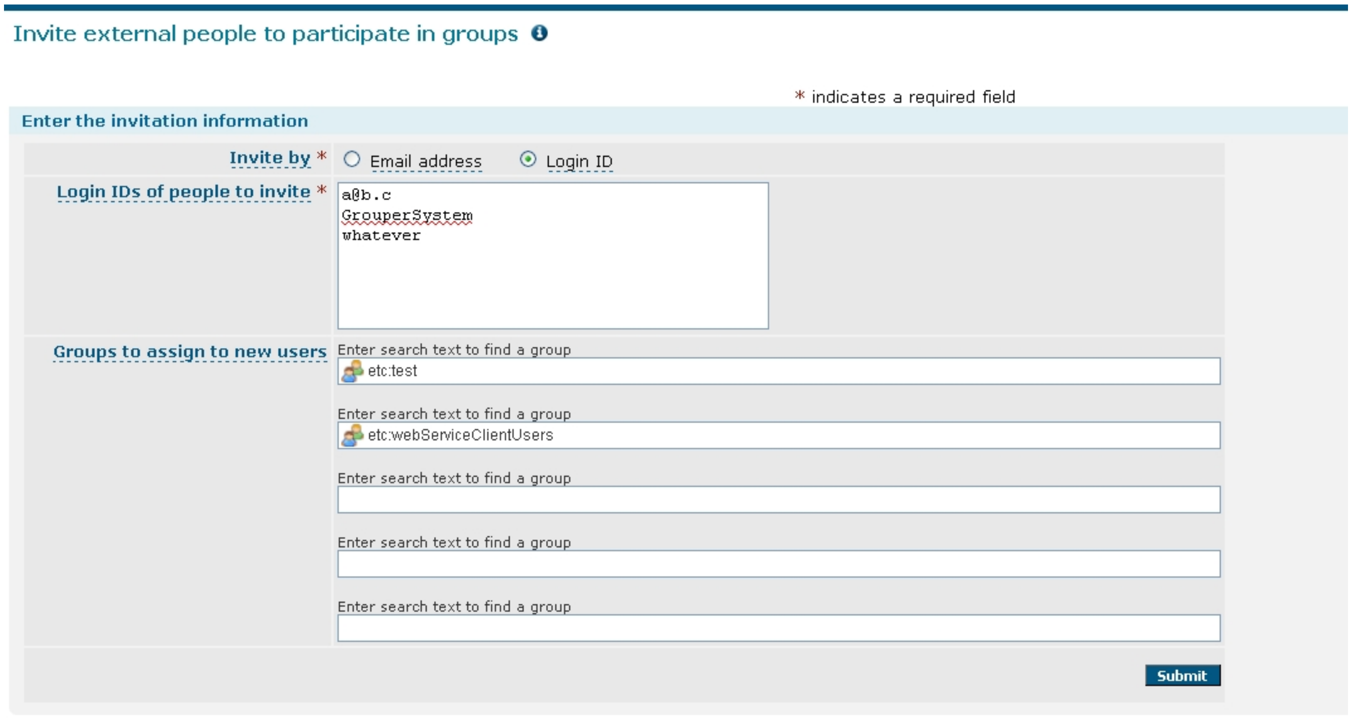
### Invite by entering in user id's (Lite UI)

If you think your users will know the exact user ID of who they want to invite, you can enabled invites by user id's.  The user will have the choice of inviting by email address or user id.

Set this in the media.properties:

```
#if we should allow invite by identifier
inviteExternalMembers.allowInviteByIdentifier = true
```

Then the invite screen has that option



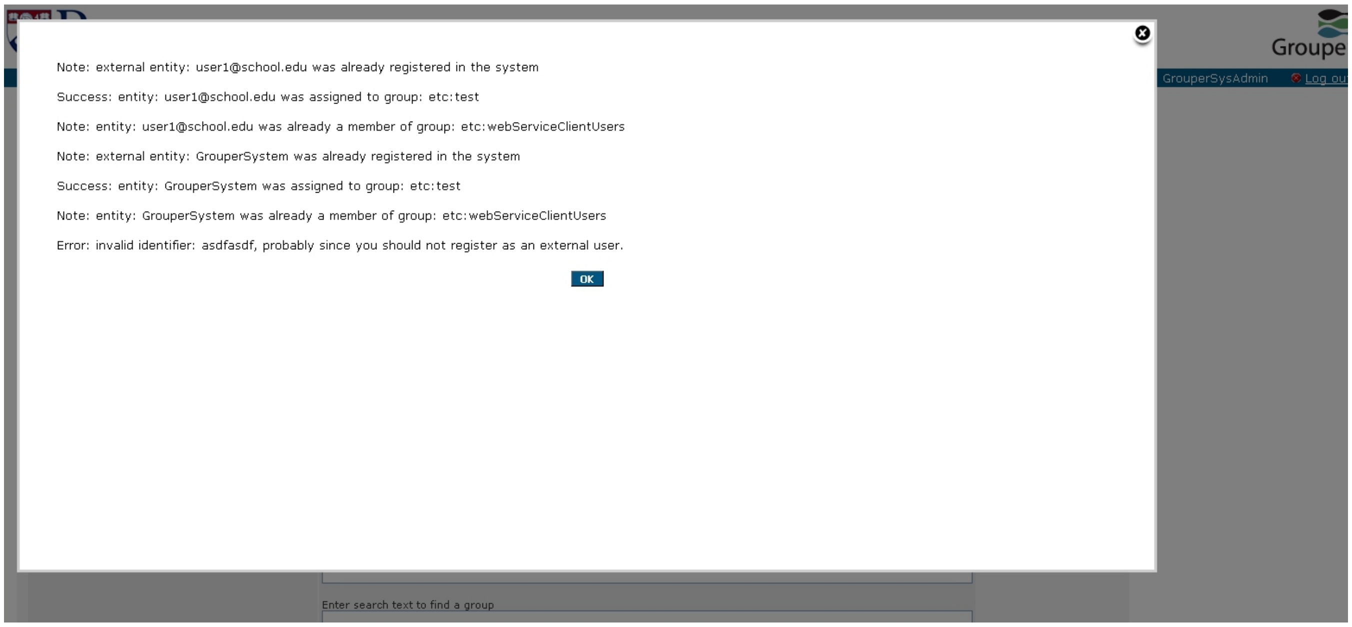
Invite external people to participate in groups

* indicates a required field

Enter the invitation information

| | |
|---|---|
| Invite by * | Email address [✓ Login ID] |
| Login IDs of people to invite * | a@b.c<br>GrouperSystem<br>whatever |
| Groups to assign to new users | Enter search text to find a group: etc:test<br>Enter search text to find a group: etc:webServiceClientUsers<br>Enter search text to find a group<br>Enter search text to find a group<br>Enter search text to find a group |

If you invite by login ID, then you will see all the results when clicking submit (nothing is emailed to the user)



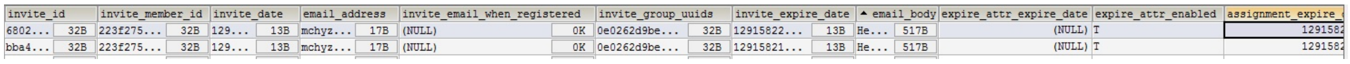
```
Note: external entity: user1@school.edu was already registered in the system
Success: entity: user1@school.edu was assigned to group: etc:test
Note: entity: user1@school.edu was already a member of group: etc:webServiceClientUsers
Note: external entity: GrouperSystem was already registered in the system
Success: entity: GrouperSystem was assigned to group: etc:test
Note: entity: GrouperSystem was already a member of group: etc:webServiceClientUsers
Error: invalid identifier: asdfasdf, probably since you should not register as an external user.
```

sdaf

## Invite view

Note that invites are stored as attributes on a stem, there is no dedicated table for them, but there is a view so you can easily see which invites are pending.  Note once an invite is clicked on form the email, it is processed and deleted.

The view is grouper_ext_subj_invite_v



| invite_id | | invite_member_id | | invite_date | | email_address | | invite_email_when_registered | | invite_group_uuids | | invite_expire_date | | ▲ email_body | | expire_attr_expire_date | expire_attr_enabled | assignment_expire_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6802... | 32B | 223f275... | 32B | 129... | 13B | mchyz... | 17B | (NULL) | 0K | 0e0262d9be... | 32B | 12915822... | 13B | He... | 517B | (NULL) | T | 1291582 |
| bba4... | 32B | 223f275... | 32B | 129... | 13B | mchyz... | 17B | (NULL) | 0K | 0e0262d9be... | 32B | 12915821... | 13B | He... | 517B | (NULL) | T | 1291582 |

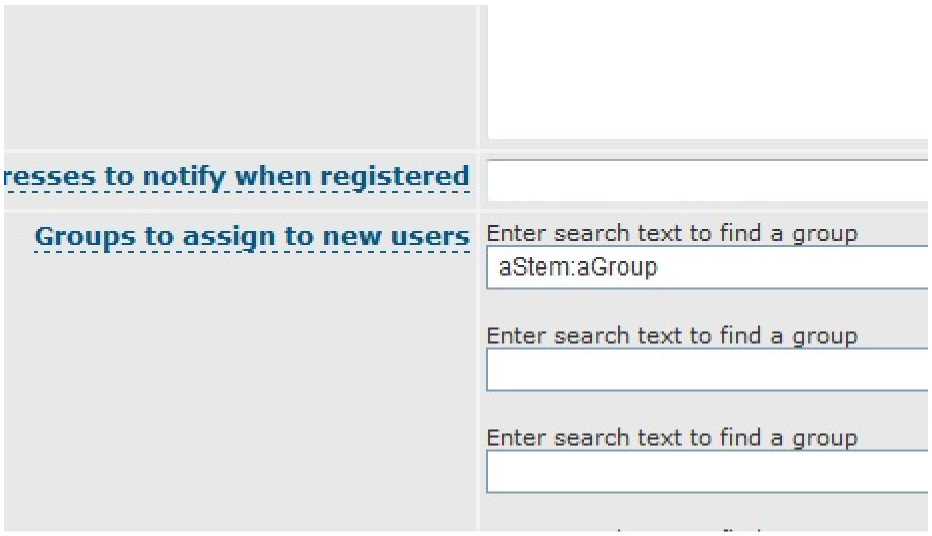
## Invite linking from other screens (Lite UI)

The invite screen (to invite other users) can be passed a groupId or groupName in the URL, and that will prepopulate the invite screen to provision the group specified in the URL.  You can pass the groupId or groupName but not both.  e.g.

If the URL is: http://localhost:8091/grouper/grouperUi/appHtml/grouper.html?operation=InviteExternalSubjects.inviteExternalSubject&groupId=0e0262d9be924774914052c12f0e7fd2

or: http://localhost:8091/grouper/grouperUi/appHtml/grouper.html?operation=InviteExternalSubjects.inviteExternalSubject&groupName=aStem:aGroup

then the screen will propulate like this:



| | |
|---|---|
| resses to notify when registered | |
| Groups to assign to new users | Enter search text to find a group<br>aStem:aGroup<br>Enter search text to find a group<br><br>Enter search text to find a group |

## Link from group membership screens (New UI)

Starting with Grouper v2.4, External Subjects and Invites are handled by default in the New UI, and the Lite UI is no longer available in the standard setup.

## Link from group membership screens (Lite UI)

There is a link from the Admin UI group membership screen.  Enable this in the media.properties:

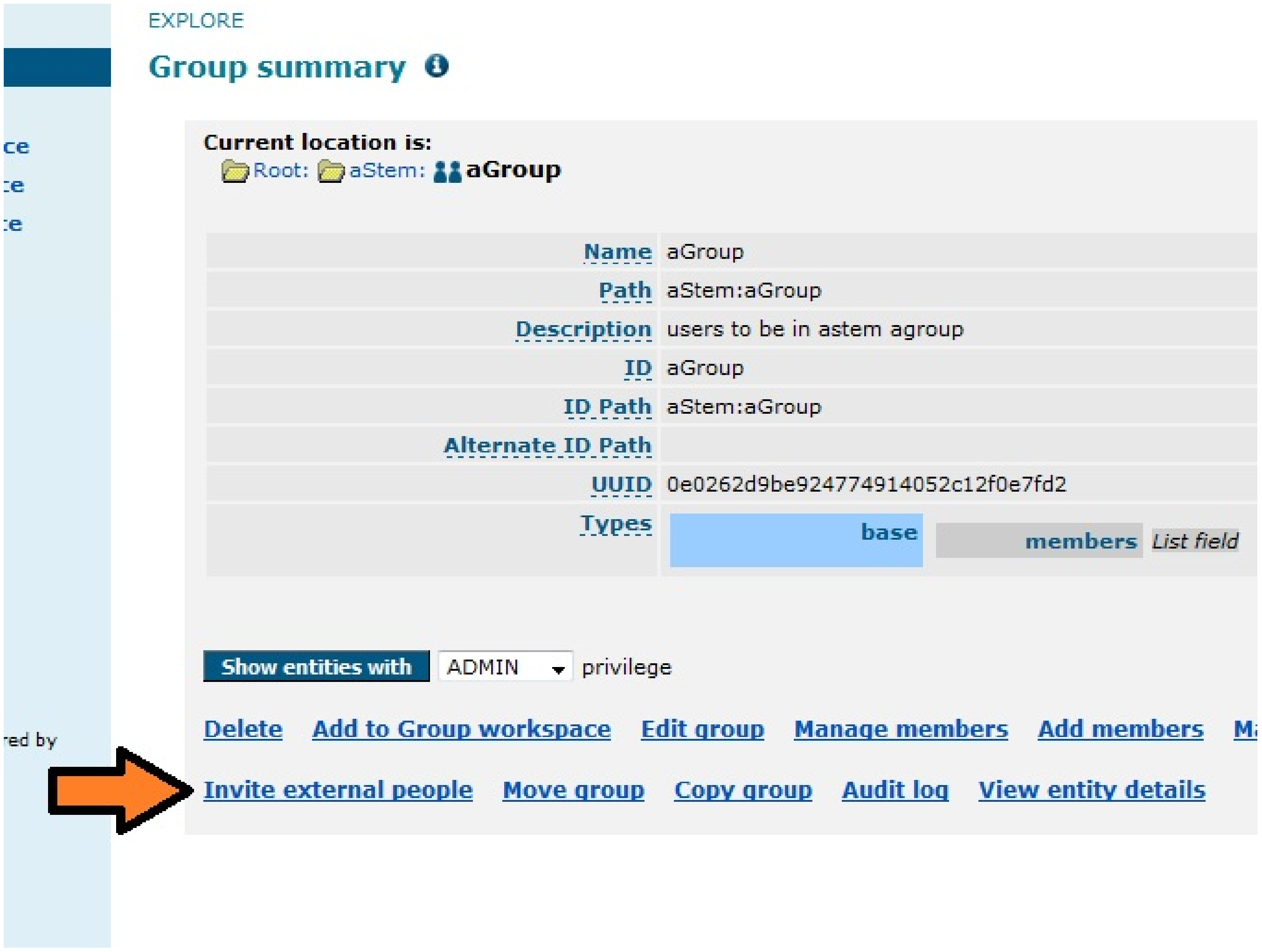
```
inviteExternalPeople.link-from-admin-ui = true
```

Configure the button text and tooltip in nav.properties

```
ui-lite.invite-link=Invite external people
tooltipTargetted.ui-lite.invite-link=Invite external people who are not already registered to be a member of
this group.<br />Note: the systems that use this group must be ready to use external people<br/>(Opens in a new
window)
```

ad

---


EXPLORE

Group summary

Current location is:
Root: aStem: aGroup

| | |
|---|---|
| Name | aGroup |
| Path | aStem:aGroup |
| Description | users to be in astem agroup |
| ID | aGroup |
| ID Path | aStem:aGroup |
| Alternate ID Path | |
| UUID | 0e0262d9be924774914052c12f0e7fd2 |
| Types | base members List field |

Show entities with ADMIN privilege

Delete Add to Group workspace Edit group Manage members Add members Ma

Invite external people Move group Copy group Audit log View entity details

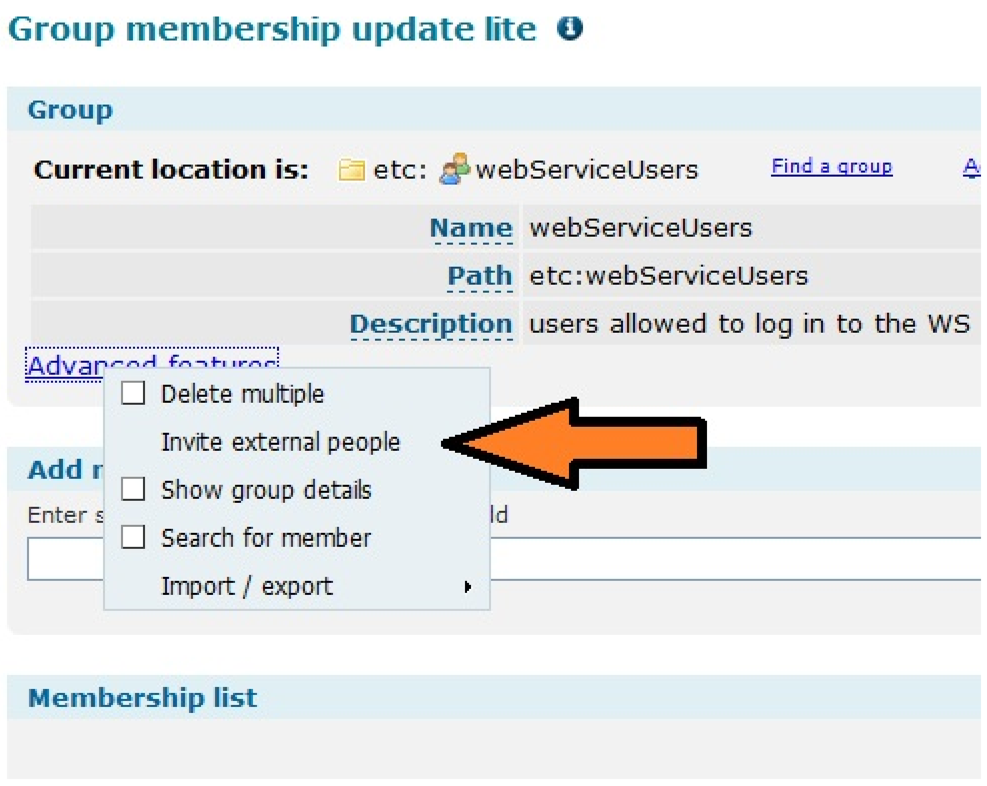
Enable lite UI link from media.properties:

```
inviteExternalPeople.link-from-lite-ui
```

Configure the button text and tooltip in nav.properties

```
ui-lite.invite-menu=Invite external people
ui-lite.invite-menuTooltip=Invite external people who are not already registered to be a member of this group.
Note: the systems that use this group must be ready to use external people.
```

See the menu item on groups that allow it and that the user has privileges for:


Group membership update lite

Group

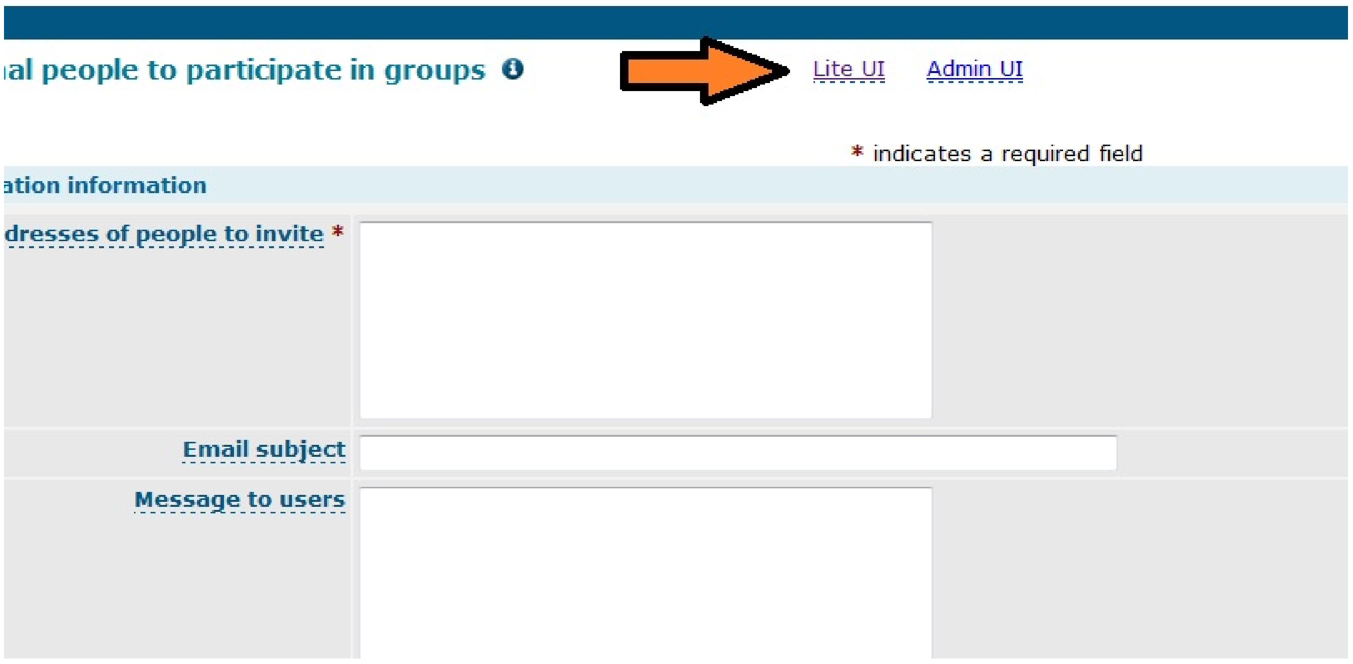
Current location is: etc: webServiceUsers

| | |
|---|---|
| Name | webServiceUsers |
| Path | etc:webServiceUsers |
| Description | users allowed to log in to the WS |

Note if you are linking to the invite screen from the admin or lite UI, and the group ID is passed in, then there will be links back to the admin or lite UI

You can customize the text in the nav.properties:

```
ui-lite.fromInvite-link=Lite UI
tooltipTargetted.ui-lite.fromInvite-link=Return to the lite membership management UI for this group

ui-lite.fromInvite-admin-link=Admin UI
tooltipTargetted.ui-lite.fromInvite-admin-link=Return to the admin UI for this group
```


al people to participate in groups
Lite UI
Admin UI
* indicates a required field
ation information
dresses of people to invite *
Email subject
Message to users

sdf

## Vetted email addresses

Once a user registers based on email, then it is vetted that the email address that the user was invited by is a vetted email address.  There is a column in the external subject table to hold a list of comma separated email addresses that are vetted.  Currently this is not used for anything (i.e. it is not a subject attribute), however, if an admin wants to contact someone, these are email addresses that the user has responded to...

## Web service interface

If you are adding a member (batches or lite, or with client), you can specify addExternalSubjectIfNotFound T or F, if this is a search by id or identifier, with no source, or the external source, and the subject is not found, then add an external subject (if the user is allowed to do this)

## Admin audit emails

If you enable admin emails for invites in media.properties:

```
#if admins should be emailed after each action, put comma separated addresses here
inviteExternalMembers.emailAdminsAddressesAfterActions = mchyzer@isc.upenn.edu
```

Then this email is sent to the admins after each invitation:

```
-----Original Message-----
From: noreply@yourschool.edu [mailto:noreply@yourschool.edu]   <-- this is configurable in grouper.properties
Sent: Monday, November 29, 2010 2:24 PM
To: Chris Hyzer
Subject: TEST:Grouper external person invitation


Hey,

The Grouper external subject invite screen was used by Subject id: GrouperSystem, sourceId: g:isa -
GrouperSysAdmin

Email addresses to invite: someone@yahoo.com
Email addresses to notify once registered: inviter@school.edu
Email subject: test subject
Message to users: test body
Group Names (ID Path) to assign: etc:webServiceClientUsers
Group UUIDs to assign: 9662c7ee3e4d4e089a2ea8e376483601
Invitation expires on: 2010-12-06 14:24:16.826

Regards.
```

You can get emails to admins on registrations too, configure in media.properties

```
#if admins should be emailed after each action, put comma separated addresses here
externalMembers.emailAdminsAddressesAfterActions = mchyzer@isc.upenn.edu
```

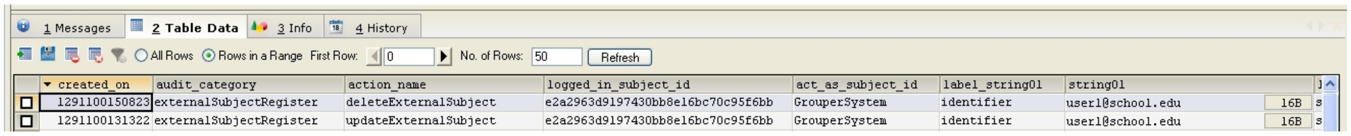
Here is an example of an email

```
-----Original Message-----
From: noreply@school.edu                         <--- this is configurable in grouper.properties
Sent: Monday, November 29, 2010 3:40 PM
To: Chris Hyzer
Subject: TEST:Grouper external person registration


Hey,

The Grouper external person registration screen was used by user1@school.edu

User: uuid: 2e4b40f631b442ae828d669da3e14007, identifier: user1@school.edu, name: User One, description: User
One - institution,
From invite? true
Edit type: update
Valid identifier: true
Message to user on screen: Success: your invitation from GrouperSysAdmin has been processed.  You were added to
the roles: webServiceClientUsers.<br /><br />

Regards.
```

## Auditing

Each invitation, registration, edit of personal information, or delete of personal information will insert a user audit record into the database.  Here is the user audit view showing some of the fields: grouper_audit_entry_v

---



| created_on | audit_category | action_name | logged_in_subject_id | act_as_subject_id | label_string01 | string01 |
|---|---|---|---|---|---|---|
| 1291100150823 | externalSubjectRegister | deleteExternalSubject | e2a2963d9197430bb8e16bc70c95f6bb | GrouperSystem | identifier | user1@school.edu 16B |
| 1291100131322 | externalSubjectRegister | updateExternalSubject | e2a2963d9197430bb8e16bc70c95f6bb | GrouperSystem | identifier | user1@school.edu 16B |

---

## A Use Case Implementation For Reference

You may want to read about the Penn Secure Space Implementation of handling external subjects

## Rule to allow/disallow external users

At Penn we didnt want external users all throughout our registry, we just wanted them in certain folders.  So at the root folder we disallowed external subjects, and in the app, we allowed them:

```
 RuleApi.vetoSubjectAssignInFolderIfNotInGroup(SubjectFinder.findRootSubject(), rootStem, null, false, "grouperExternal", Stem.Scope.SUB, "rule.entity.cannot.be.external", "Person cannot be assigned if an external user");

RuleApi.vetoSubjectAssignInFolderIfNotInGroup(SubjectFinder.findRootSubject(), allowedStem, null, true, "grouperExternal", Stem.Scope.SUB, "rule.entity.can.be.external", "Person can be external");
```

## To do

- Add multiple search strings and sort fields based on new member columns