

# Grouper loader managed apache example

<a href="#">Wiki Home</a>	<a href="#">Grouper Release Announcements</a>	<a href="#">Grouper Guides</a>	<a href="#">Grouper Deployment Guide</a>	<a href="#">Community Contributions</a>	<a href="#">Internal Developer Resources</a>
---------------------------	---	--------------------------------	--	---	--

University of Pennsylvania wants to protect part of Apache with Grouper based on a SQL loaded group for a new application. Note that most institutions can use the apache mod\_authnz\_ldap, but our LDAP doesn't have memberOf and isn't conducive to it.

So the architecture we will have is the grouperClient which calls a WS to keep an apache .htaccess up to date via cron.

First, we use kerberos principals for authentication to Grouper WS, so we created a kerberos principal / pass: application\_grouper/server.school.edu

Then we have a custom non-grouper application to enter this kerberos principal into our kerberos principal subject source, note, this is an admin action.

The screenshot shows the PennGroups web application interface. On the left is a dark blue sidebar with the University of Pennsylvania logo and the text 'PennGroups' and 'Service principals'. The main content area has a title 'Enter service principal' and instructions: 'Enter a kerberos service principal below. This will allow this service principal to log in to the Pennkey to PennID translation service LDAP (which is also the PennGroups LDAP). Note that the Penn ISC Data Administration group will be emailed about this action. The change will take effect in a few hours after the data propagates.' Below the instructions are two input fields: 'Service principal name' and 'Reason\*'. A 'Submit' button is at the bottom right. The footer contains copyright information: 'Copyright © 2011, University of Pennsylvania. All rights reserved. Statement on privacy'.

Then I add this kerberos principal to the WS users group: school:etc:webServiceClientUsers

This is an application which didn't previously use Grouper, so I create a folder in the applications folders in the IT department: school:it:ait:apps:appName

In there, I like to keep the privileges in groups, so make a group for that: school:it:ait:apps:appName:etc:appNameReaders

Add the kerberosPrincipal and the client person to that group.

Make a loader group: school:it:ait:apps:appName:groups:apacheGroup

Make sure that group is of grouperLoader type

Add the school:it:ait:apps:appName:etc:appNameReaders group to have the READ privilege on school:it:ait:apps:appName:groups:apacheGroup. Note, dont give admin because you dont want a non admin editing the attributes on a loader group. Generally you dont give "update" to a grouper loader group since the memberships will be reset. Maybe the updater knows what they are doing and can onboard quicker...

In this case we have a new database connection since the SQL is not loaded from the Grouper database. The DBA setup a new schema for use so that all we have permissions in the DB is to select from a certain view. We need to add in a database connection in the grouper-loader.properties:

```
db.application.user = APP_SCHEMA
db.application.pass = /home/user/pass/grouper/grouperMorphApp.pass
db.application.url = jdbc:oracle:thin:@(DESCRIPTION=(LOAD_BALANCE=YES) (FAILOVER=YES) (ADDRESS_LIST=(ADDRESS=
(PROTOCOL=tcp) (HOST=1.2.3.4) (PORT=1234)) (ADDRESS=(PROTOCOL=tcp) (HOST=1.2.3.5) (PORT=1235))) (CONNECT_DATA=
(SERVICE_NAME=appdb.school.edu)))
db.application.driver = oracle.jdbc.driver.OracleDriver
```

Note the password is not in the file, so encrypt that and put it in the file:

```
[user@server bin]$ cd /opt/tomcat_9b/webapps/grouper/WEB-INF/lib
[user@server lib]$ java -jar morphString.jar
Enter the location of morphString.properties: ../classes/morphString.properties
Type the string to encrypt (note: pasting might echo it back):
The encrypted string is: abc123abc123abc123==
[user@server lib]$ echo abc123abc123abc123== > /home/user/pass/grouper/grouperMorphApp.pass
```

Now we can fill in the grouperLoader attributes, this will update every couple of hours during the day, and simple sql:

The screenshot shows the Grouper web interface for the University of Pennsylvania. The top navigation bar includes the Penn logo and the Grouper logo. The main content area displays the 'Group summary' page for the 'Root' group. A message box at the top states 'Note: Attributes were saved'. Below this, the 'Current location is: Root' is shown. A table of attributes for the 'grouperLoader' entity is displayed, with columns for Name, Path, Description, ID, ID Path, Alternate ID Path, UUID, Types, base, members, and List field. The 'grouperLoader' entity is highlighted in blue. The 'base' column shows 'grouperLoader'. The 'members' column lists various attributes like 'grouperLoaderAndGroups', 'grouperLoaderDbName', 'grouperLoaderGroupQuery', 'grouperLoaderGroupTypes', 'grouperLoaderGroupsLike', 'grouperLoaderIntervalSeconds', 'grouperLoaderPriority', 'grouperLoaderQuartzCron', 'grouperLoaderQuery', 'grouperLoaderScheduleType', and 'grouperLoaderType'. The 'List field' column shows the value 'select pennid as subject\_id from'.

Run this with GSH to test it out:

```
gsh 0% grouperSession = GrouperSession.startRootSession();
gsh 1% loaderGroup = GroupFinder.findByName(grouperSession, "school:it:ait:apps:appName:groups:apacheGroup");
gsh 2% loaderRunOneJob(loaderGroup);
```

Bounce the grouper-loader process (at Penn this run in a tomcat, so we bounce that tomcat).

Make a .htaccess file to protect a directory in apache (or do something else in apache config) (note, we use cosign, protect with whatever authentication you have)

```
[mchyzer@flash public_html]$ pwd
/home/mchyzer/public_html
[mchyzer@flash public_html]$ cat .htaccess
CosignService isc-seo-studentHomeSso_dev-0
CosignCrypto /opt/appserv/weblogin/cert_isc-seo-studentHomeSso_dev/isc-seo-studentHomeSso_dev-0.key /opt
/appserv/weblogin/cert_isc-seo-studentHomeSso_dev/isc-seo-studentHomeSso_dev-0.crt /opt/appserv/weblogin/ca/

CosignProtected On
AuthType Cosign

CosignRequireFactor UPENN.EDU

AuthGroupFile /home/mchyzer/public_html_config/appUsers.txt
Require group appUsers
[mchyzer@flash public_html]$
```

Now we need the beginning and end of the generated group file:

```
[mchyzer@flash public_html_config]$ more appUsersStart.txt
appUsers:
[mchyzer@flash public_html_config]$
```

Get the grouper client in that dir, and configure the grouper.client.properties WS URL, user, pass (based on kerberos principal):

```
[mchyzer@flash public_html_config]$ ls grouper*
grouperClient.jar grouper.client.properties grouper.client.properties~
[mchyzer@flash public_html_config]$ cat grouper.client.properties

...

# url of web service, should include everything up to the first resource to access
# e.g. http://groups.school.edu:8090/grouperWs/servicesRest
# e.g. https://groups.school.edu/grouperWs/servicesRest
grouperClient.webService.url = http://server.school.edu/grouperWs/servicesRest

# kerberos principal used to connect to web service
# e.g. name/server.whatever.upenn.edu
grouperClient.webService.kerberosPrincipal = application_grouper/server.school.edu

# password for shared secret authentication to web service
# or you can put a filename with an encrypted password
grouperClient.webService.password = abc123abc123abc123

...
```

Test the connection:

```
[mchyzer@flash public_html_config]$ java -jar grouperClient.jar --operation=getMembersWs --groupNames=school:it:
ait:apps:appName:groups:apacheGroup
GroupIndex 0: success: T: code: SUCCESS: group: school:it:ait:apps:appName:groups:apacheGroup: subjectIndex: 0:
12345678
GroupIndex 0: success: T: code: SUCCESS: group: school:it:ait:apps:appName:groups:apacheGroup: subjectIndex: 1:
87654321
[mchyzer@flash public_html_config]$
```

Write a script to generate the user file:

```

[mchyzer@flash public_html_config]$ more appUsers.sh
#!/bin/bash

#make a backup
cp /home/mchyzer/public_html_config/appUsers.txt /home/mchyzer/public_html_config/appUsersBak.txt

#build the new file into a temp file so its not temporarily empty
cat /home/mchyzer/public_html_config/appUsersStart.txt > /home/mchyzer/public_html_config/appUsersTemp.txt

/opt/java6/bin/java -jar /home/mchyzer/public_html_config/grouperClient.jar--operation=getMembersWs --
groupNames=school:it:ait:apps:appName:groups:apacheGroup --subjectAttributeNames=PENNNNAME --
outputTemplate='${wsSubject.attributeValues\[0\]}$space$' >> /home/mchyzer/public_html_config/appUsersTemp.txt
2>/home/mchyzer/public_html_config/appUsersLog.txt

ret=$?

if [ $ret -ne 0 ]; then
    mail someone@school.edu -s "Error syncing app apache group" < /home/mchyzer/public_html_config/appUsersLog.txt
    exit 1
fi

mv /home/mchyzer/public_html_config/appUsersTemp.txt /home/mchyzer/public_html_config/appUsers.txt

```

Put this in a cron, note, this cron should be right after the previous loader cron

```

[mchyzer@flash public_html_config]$ crontab -l
25 8,10,12,14,16 * * * /home/mchyzer/public_html_config/appUsers.sh > /home/mchyzer/public_html_config/appUsers.
log 2>&1

```

sdf