Authentication to UI and Web Services in Grouper v2.5+

- Add a new password via gsh for UI
- Add a new password via gsh for WS
- Example with local entity and WS authentication
- Summary
- Password table in Grouper: grouper_password
- JWT table recently used in Grouper: grouper_password_recently_used
- Manage passwords
- Basic authn built in to Grouper
- Passwords for WS

Add a new password via gsh for UI

```
v2.5.29+
new GrouperPasswordSave().assignApplication(GrouperPassword.Application.UI).assignUsername("GrouperSystem").
assignPassword("password").save();
```

Add a new password via gsh for WS

Note: if you are setting a password for a local entity to do web service calls, you should probably use the uuid (unique id) as the username, though the system name (id) might work too (it works in grouper client). Colons shouldnt be used in HTTP usernames, so the uuid is better

```
v2.5.29+
new GrouperPasswordSave().assignApplication(GrouperPassword.Application.WS).assignUsername("GrouperSystem").
assignPassword("password").save();

Local entity with uuid
new GrouperPasswordSave().assignApplication(GrouperPassword.Application.WS).assignUsername
("7a7937ad646849fc8278fb2fc6c45156").assignPassword("password").save();
```

Example with local entity and WS authentication

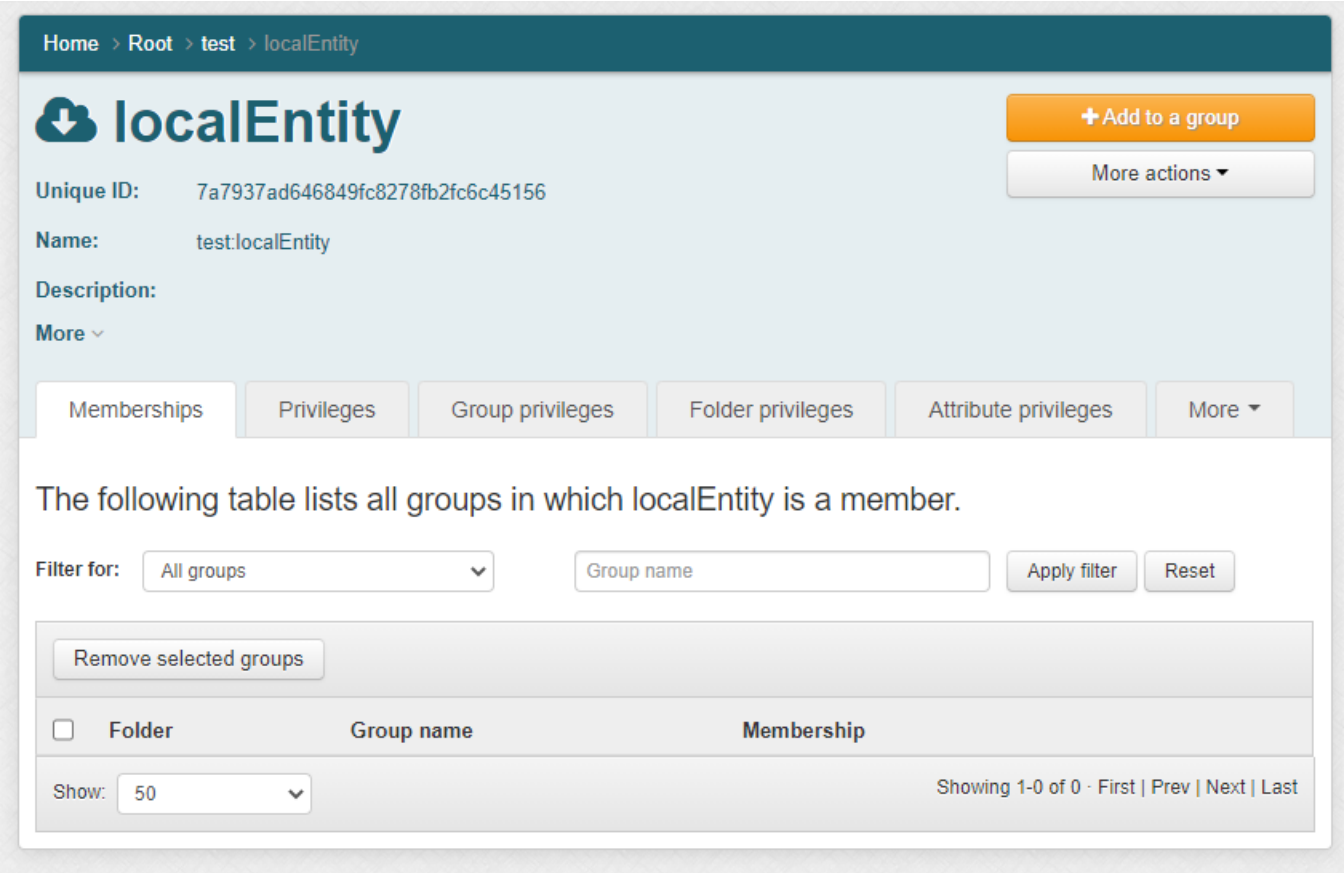
For example with container 2.5.36

Start the quickstart

```
docker run --detach --name grouper-qs \
  --publish 443:443 -e GROUPER_MORPHSTRING_ENCRYPT_KEY=abcdefg12345dontUseThis \
  -e GROUPERSYSTEM_QUICKSTART_PASS=pass  i2incommon/grouper:2.5.36 quickstart
```

Note: quickstart sets this env var: -e GROUPER_WS_GROUPER_AUTH=true

Add a local entity



Set a password:

```
mchyzer@ISC20-0637-WL:~/container$ docker exec -it -u tomcat grouper-qs bash
[tomcat@f7adb51426d3 WEB-INF]$ cd bin
[tomcat@f7adb51426d3 bin]$ ./gsh.sh
groovy:000> new GrouperPasswordSave().assignApplication(GrouperPassword.Application.WS).assignUsername
("7a7937ad646849fc8278fb2fc6c45156").assignPassword("myPass").save();
groovy:000> :q
[tomcat@f7adb51426d3 bin]$ exit
```

Make a group: test:testGroup and allow test:localEntity to READ it, and add GrouperSystem as member

Call web service with grouper client

```
mchyzer@ISC20-0637-WL:~/container$ docker cp grouper-qs:/opt/grouper/grouperWebapp/WEB-INF/lib/grouperClient-
2.5.36.jar .

mchyzer@ISC20-0637-WL:~/container$ vi grouper.client.properties
grouperClient.webService.url = https://localhost:443/grouper-ws/servicesRest
grouperClient.webService.login = 7a7937ad646849fc8278fb2fc6c45156
grouperClient.webService.password = myPass
# turn off SSL until a real SSL certificate is installed
# NOTE, THIS IS NOT GOOD SECURITY AND IS FOR THE QUICK START ONLY!
grouperClient.https.customSocketFactory = edu.internet2.middleware.grouperClient.ssl.EasySslSocketFactory

mchyzer@ISC20-0637-WL:~/container$ java -jar grouperClient-2.5.36.jar --operation=getMembersWs --
groupNames=test:testGroup
GroupIndex 0: success: T: code: SUCCESS: group: test:testGroup: subjectIndex: 0: GrouperSystem
```

Or curl:

```
mchyzer@ISC20-0637-WL:~/container$ curl --insecure --user 1ebc381f335c4c6f8dadfc5b76e85dc8:myPass
https://localhost:443/grouper-ws/servicesRest/v2_5_000/groups/test%3AtestGroup/members
```

Summary

This page outlines the approach to authentication to UI and web services in Grouper 2.5 and above.

First off, whatever authentication you may have used in the past for Grouper UI and WS will still be available.
For example, if you are using Shibboleth and LDAP WS authentication, then most of this won't apply to you (only the section on restricting source IP address).

Grouper 2.5+ provides a better built-in WS authentication method than basic auth

- Passwords not stored in clear text in tomcat-users
- Passwords not transmitted on the wire
- We need something not tomcat specific
- Tomcat config files are painful to automate the use/removal of
- Ability to filter the source address for WS

This provides easier quick starts and bootstraps in UI/WS

These default to off and can be enabled in config.

Who can change passwords? Admins or admins of service accounts

Complexity? Grouper assigns complex passwords

Password table in Grouper: grouper_password

Note, even if Grouper is not doing authn, it could still restrict the source address. For WS, any authns would get a record inserted or updated here

Column	Type	Description
id	varchar (40)	uuid of this entry (one user could have ui and ws credential)
username	varchar (255)	username or local entity system name
member_id	varchar (40)	this is a reference to the grouper members table. dont make a foreign key right now. When someone logs in, save their GrouperPassword object in the request somewhere, and when the subject is resolved, if the member id resolved doesnt match the member id in the GrouperPassword row, then update it and store to the database. this column should have a non-unique index (since same entity can have multiple rows here)
entity_type	varchar (20)	username or localEntity
is_hashed	varchar (1)	T for is hashed, F for is public key
encryption_type	varchar (20)	e.g. SHA-256 or RS-256 (key type)
the_salt	varchar (255)	secure random prepended to hashed pass
the_password	varchar (4000)	encrypted public key or encrypted hashed salted password
application	varchar (20)	ws (includes scim) or ui
allowed_from_cidrs	varchar (4000)	network cidrs where credential is allowed from

recent_source_addresses	varchar (4000)	json with timestamps. only successes. (limit to most recent 20) [{ millis: 123455667, ip: "1.2.3.4" }, { millis: 123455669, ip: "1.2.3.5" }] To parse from json to JSONObject to array System.arrayCopy, move index 0-(length-1) -> 1-(length), add most recent to index 0 convert back to JSON array and object and string and store
failed_source_addresses	varchar (4000)	if restricted by cidr, this was failed IPs (json with timestamp?) (limit to most recent 10) [{ millis: 123455667, ip: "1.2.3.4" }, { millis: 123455669, ip: "1.2.3.5" }]
last_authenticated	timestamp	when last authenticated successful
last_edited	timestamp	when this was last edited
failed_logins	varchar (4000)	Keep 20 [{ millis: 123455667, ip: "1.2.3.4" }, { millis: 123455669, ip: "1.2.3.5" }]

JWT table recently used in Grouper: grouper_password_recently_used

A process would clean these out after the configured drift (10 minutes)

Unique index on the tuple: grouper_password_id and jwt_jti

Column	Type	Description
id	varchar(40)	uuid of this row (primary key)
grouper_password_id	varchar(40)	foreign key to grouper_password table
jwt_jti	varchar (100)	e.g. uuid of this entry (sent from client)
jwt_iat	integer (11)	seconds since 1970 that this was issued

Manage passwords

UI for admins to set a user's (or local entity's) UI password or could restrict source IP cidrs. UI passwords would need to follow strength rules

UI for admins or end users (self serve) to download a new generated WS private key or password for a local entity they can ADMIN or restrict source IP cidrs

- Someone who can create in a folder (and optionally in a group who can create WS credentials)
- Create a local entity
- Download its password or private key (can only download once)
- Grant privs to the local entity
- Use it in WS calls

Admins and end users can not view or re-download passwords or private keys

Basic authn built in to Grouper

If configured (for quick start only), the UI could use basic auth and use passwords configured for users

Its possible users could reset their password using their old password to authenticate.

Passwords for WS

Your LDAP or Kerberos or apache or tomcat authn would still work. Its possible there could be multiple allowed... i.e. to transition into local entity JWT authn. Depending on configuration.

Private key signed JWT would be recommended with WS, or required at some sites. Source IP's could be required too

- Username is the system name of the local entity
- Private key is a generated by Grouper and downloaded once
    - This is not sent across the wire in WS calls

JWT details

- To authenticate with JWT the client would
    - Generate a valid jwt jti (e.g. uuid)
    - Have the correct time within configured drift (10 minutes?), get the seconds since 1970 (GMT)
    - Send a "Bearer" authorization header sfdlh23kjh.kjhsdfkjhsf.kjh345kjhkjh (three parts separated by dot)
    - First part is the header is base64 url encoded

```
{
  alg: "RS-256",
  typ: "JWT"
}
```

    - The second part is what makes the token unique and identifies the user
        - jti is a unique value per request (across clusters), cannot be re-used. e.g. a uuid
        - username is: system name of local entity
        - iat: Number of seconds since 1970 (that the ticket is issued), the number received on server needs to be within the allowable time drift

```
{
  jti: "abc123",
  username: "org:businessSchool:credentials:wiki",
  iat: 1234567
}
```

- Thus the same request cannot be replayed

See Also

Grouper Web Services Authentication

Authentication to the Grouper UI