Grouper Atlassian Provisioner

Integrating Atlassian with Grouper

People have had success with this SAML connector: https://github.com/chauth/confluence_http_authenticator

Here is a sample remoteUserAuthenticator.properties

```
create.users=true
update.info=true
update.roles=true
default.roles=confluence-users
dynamicroles.auto_create_role=true
header.fullname=displayName
header.email=mail
header.remote_user=REMOTE_USER
header.dynamicroles.attributenames=member
dynamicroles.header.member=groupmap
dynamicroles.mapper.groupmap.match=(.*)
dynamicroles.mapper.groupmap.transform=$1
reload.config=true
reload.config.check.interval=15000
purge.roles=(.*)
```

You can also integrate through LDAP. In the cloud an option is through azure.

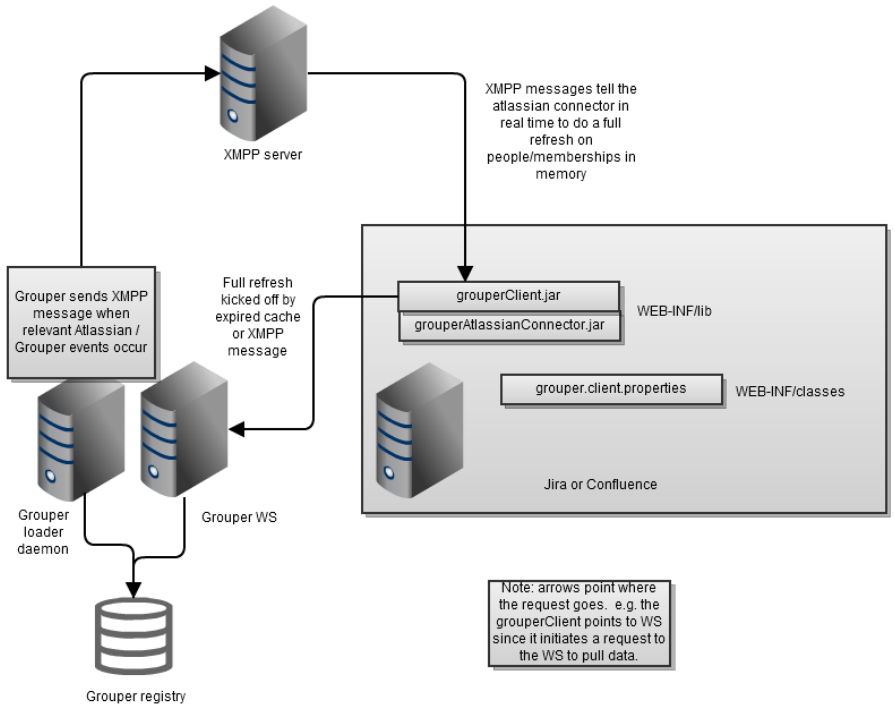
Outdated connector

To do a full reconcile on the new 2.2.1+ grouper atlassian conector, run this:

```
java -cp conf:lib/* edu.internet2.middleware.grouperAtlassianConnector.db.GrouperAtlassianDataReconcile full
notReadonly
```

The grouper Atlassian connector implements the Atlassian access and profile providers (which are OpenSymphony interfaces, so other software which uses those interfaces might be able to use this as well). The connector uses the grouperClient which uses Grouper REST web services. Note that you can connect Atlassian products to LDAP, which can be provisioned from Grouper. One difference between that and this connector is this connector is read /write. This connector has been tested with Grouper 1.6 and Atlassian as of 12/2010 (e.g. Jira 4.2.1).

Note: if the diagram here is obstructed, try clicking the "Full Size" option below it.



To set this up, check it out and build it:

```
[mchyzer@flash grouperAtlassian]$ svn co http://anonsvn.internet2.edu/svn/i2mi/branches/GROUPER_1_6_BRANCH
/grouper-misc/grouperAtlassianConnector
[mchyzer@flash grouperAtlassian]$ cd grouperAtlassianConnector/
[mchyzer@flash grouperAtlassianConnector]$ ant
```

Take the grouper.client.example.properties in the grouperAtlassianConnector/conf/grouper.client.example.properties, and put the connector properties in your own grouper.client.properties:

```
################################
## Atlassian connector settings
################################

# put a folder name that is the root for atlassian groups
atlassian.root =

# atlassian source to use (leave blank for all sources)
atlassian.subject.search.sourceId =

# atlassian search by id, identifier, or idOrIdentifer (idOrIdentifier is Grouper 2.0+)
atlassian.subject.search.subjectId = identifier

# number of minutes to cache reads
# defaults to 10.  Note, crank this up  to 25 hours if you are doing XMPP notifications
atlassian.cache.minutes = 10

# number of minutes to cache profile reads
# defaults to 10
atlassian.cache.profile.minutes = 20

# each cache has a failsafe cache, so that if grouper is down, and the data has been loaded,
# since atlassian has been started, the stale verison of the data can be retrieved
atlassian.cache.failsafe.hours = 48
```

```
# list all sources here, and how to get the atlassian id
atlassian.source.jdbc.sourceId = jdbc
# should be "id" or an attribute name to get the identifier for atlassian
atlassian.source.jdbc.idOrAttribute = loginid
# email attribute for this source (needed if using the ProfileProvider)
atlassian.source.jdbc.emailAttribute = EMAIL
# should be "name" or "description" or an attribute name to get the name for atlassian (needed if using the
ProfileProvider)
atlassian.source.jdbc.nameAttribute = name

#atlassian name of group which has all users in it, e.g. jira-users
atlassian.usersGroup = jira-users

# grouper group name of all users that have ever been in atlassian (profile service has access to these).
Leave blank to
# just use the users group
atlassian.grouperAllUsersGroup =

# if you are doing XMPP for cache clearing, set to true, and set the XMPP sections of this config
atlassian.registerXmppListeners = false

# if incremental changes come through, then dont clear now, clear sometime in the future so that multiple
changes
# cause fewer cache refreshes.  Note that changes come through the change log so that they are already buffered
a little bit
# this should probably at least be 15 seconds...
atlassian.xmppIncrementalClearCacheSecondsInFuture = 75

# if all users must be in atlassian.grouperAllUsersGroup,
# or if lookups of old users can be done without having to be in this group
atlassian.requireGrouperAllUsersGroupForLookups = false

# groups which should be assigned to various privileges for new groups created in confluence
atlassian.updaters =
atlassian.admins =
atlassian.readers =

# pretend these memberships exist (e.g. to bootstrap or for users not in grouper)
atlassian.autoadd.administrators.groupname = jira-administrators
atlassian.autoadd.administrators.usernames = admin

atlassian.autoadd.users.groupname = jira-users
atlassian.autoadd.users.usernames = admin

# users not in idm, this is needed if using the profile provider
atlassian.autoadd.admin.user.id = admin
atlassian.autoadd.admin.user.name = Atlassian ADMIN
atlassian.autoadd.admin.user.email = you@yourschool.edu


#ignore calls on this user to the web service
atlassian.ws.users.to.ignore = admin

#put a valid subject id or identifier here for testing, and that user's email and name
atlassian.test.subjectIdOrIdentifier =
atlassian.test.email =
atlassian.test.name =

# if you are using the edu.internet2.middleware.grouperAtlassianConnector.GrouperLoggingAccessProviderWrapper
# to log an access provider, set the underlying class here
atlassian.logging.accessProvider.class = com.atlassian.jira.user.osuser.JiraOFBizAccessProvider

# if you are using the edu.internet2.middleware.grouperAtlassianConnector.GrouperLoggingAccessProviderWrapper
# to log an access provider, set the underlying class here
atlassian.logging.profileProvider.class = com.atlassian.jira.user.osuser.JiraOFBizProfileProvider

# if using the external authenticator, then this says if we should store the user token in session
# as opposed to getting it from the external authentication each time
atlassian.authentication.cacheUserToken = false

# if using the external authenticator, then this is the request attribute where the principal name is
```

```
# note, if it is REMOTE_USER, that will already be checked
atlassian.authentication.requestPrincipalAttributeName =

# if you are not in prod, and you want to backdoor as someone else, put a parameter name here (e.g. gibberish
alphanumeric, or backdoorNetid)
# and if that param is in the URL or posted to the application, it will be used.  Note, if you use this then
cacheUserToken will be true
# since this will not be in the URL for every request
atlassian.authentication.backdoorRequestParameterName =

# if you are not in prod, and you want to backdoor as someone else, put comma separated usernames here who
# are allowed to backdoor as someone else
atlassian.authentication.backdoorAllowedUsers =
```

Configure the WS url, and authentication in the grouper.client.properties, as you normally would, here is a kerberos (cleansed) example

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

 # kerberos principal used to connect to web service
 grouperClient.webService.login = atlassianGrouper/server.school.edu

 # password for shared secret authentication to web service
 # or you can put a filename with an encrypted password
grouperClient.webService.password = /home/user/pass/atlassianGrouper.pass
```

Take the grouperAtlassianConnector/dist/grouperAtlassianConnector.jar, the grouperClient.jar, and put them in the jira edit-webapp/WEB-INF/lib dir. Take the grouper.client.properties and put it in the jira edit-webapps/WEB-INF/classes dir. Take the osuser.xml file that was in the jira WEB-INF/classes dir. Comment out the existing access and profile (optional) providers, and configure the grouper one(s)

```
<!-- provider class="com.atlassian.jira.user.osuser.JiraOFBizCredentialsProvider">
    <property name="exclusive-access">true</property>
  </provider>

        <provider class="com.atlassian.jira.user.osuser.JiraOFBizProfileProvider">
    <property name="exclusive-access">true</property>
  </provider>

  <provider class="com.atlassian.jira.user.osuser.JiraOFBizAccessProvider">
    <property name="exclusive-access">true</property>
  </provider -->

        <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperCredentialsProvider">
    <property name="exclusive-access">true</property>
  </provider>

        <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperProfileProvider">
    <property name="exclusive-access">true</property>
  </provider>

  <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperAccessProvider">
    <property name="exclusive-access">true</property>
  </provider>
```

For confluence, the file looks a little different, but same concept:

```
<!--
    <provider class="bucket.user.providers.CachingCredentialsProvider">
        <property name="chain.classname">com.opensymphony.user.provider.hibernate.HibernateCredentialsProvider<
/property>
        <property name="chain.configuration.provider.class">bucket.user.BucketHibernateConfigProvider</property>
    </provider>

    <provider class="bucket.user.providers.CachingAccessProvider">
        <property name="chain.classname">com.opensymphony.user.provider.hibernate.HibernateAccessProvider<
/property>
        <property name="chain.configuration.provider.class">bucket.user.BucketHibernateConfigProvider</property>
    </provider>

    <provider class="bucket.user.providers.CachingProfileProvider">
        <property name="chain.classname">com.opensymphony.user.provider.hibernate.HibernateProfileProvider<
/property>
        <property name="chain.configuration.provider.class">bucket.user.BucketHibernateConfigProvider</property>
    </provider>
-->

        <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperCredentialsProvider">
                <property name="exclusive-access">true</property>
        </provider>

        <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperProfileProvider">
                <property name="exclusive-access">true</property>
        </provider>

        <provider class="edu.internet2.middleware.grouperAtlassianConnector.GrouperAccessProvider">
                <property name="exclusive-access">true</property>
        </provider>
```

Note, for confluence, you also need to edit the atlassian-user.xml file:

```
<osuser key="osuserRepository" name="OSUser Repository"/>

        <!-- Default confluence user repository -->
        <hibernate name="Hibernate Repository" key="hibernateRepository"  description="Hibernate Repository"
cache="true"/>
```

Stop the jira tomcat, build the package, delete the webapps/jira dir in the tomcat, and start the jira tomcat again (note: these steps will vary on how you installed Jira, but in generally, you need the two jars, the client config, and the osuser.xml config in the right place in WEB-INF subdirs.

Note, in the system settings for Jira/Confluence, you can check the checkbox for external user management, or external password management. Note that these affect more than just if users can update their profile, external user management means you cant edit groups or memberships from the UI anymore...

Modules

There are 4 modules to this connector:

AccessProvider: externalizes groups

ProfileProvider: externalize name/email/user list to grouper and subjects

CredentialsProvider: integrates external authentication with external profile provider

external authentication: allows web server plugin authentication

External authenticator

If you use shib or cosign or some web server plugin for authentication, there are other ways to integrate with confluence, but this jar has a way too. Just set this in the seraph-config.xml

```
<!-- authenticator class="com.atlassian.confluence.user.ConfluenceAuthenticator"/ -->
    <authenticator class="edu.internet2.middleware.grouperAtlassianConnector.externalAuthentication.
ExternalConfluenceAuthenticator"/>
```

Note that you can enable the backdoor with this in the grouper.client.properties to allow certain people to be able to login as other users:

```
# if you are not in prod, and you want to backdoor as someone else, put a parameter name here (e.g. gibberish
alphanumeric, or backdoorNetid)
 # and if that param is in the URL or posted to the application, it will be used.  Note, if you use this then
cacheUserToken will be true
 # since this will not be in the URL for every request
 atlassian.authentication.backdoorRequestParameterName = backdoorNetId

 # if you are not in prod, and you want to backdoor as someone else, put comma separated usernames here who
 # are allowed to backdoor as someone else
 atlassian.authentication.backdoorAllowedUsers = jsmith, whoever
```

Then, close all your browser windows to stop your session. Then open a browser and go to this URL:

https://server.school.edu/jira/secure/Dashboard.jspa?backdoorNetId=rwilson

Migrate Jira groups and memberships

First step should be when exporting stuff from old jira to new, look at group names, and search and replace the old names to new names of groups you want to rename. e.g. if there is a space of invalid char, you could replace with a dash or underscore or whatever you want to do. Then import.

Note, you will need to migrate existing groups and memberships to grouper. You can easily do this with some database scripts that create GSH scripts. Here is an example of a group create script for mysql:

We are creating a script with a line for each group like this:

```
addGroup(parent stem name, extension, displayExtension)
```

Run a query like this against the atlassian schema:

```
SELECT CONCAT(CONCAT(CONCAT(CONCAT('addGroup("test:school:apps:atlassian:groups", "', groupname), '", "'),
groupname), '");') AS gsh_script FROM groupbase;
```

Run the resulting GSH script against grouper.

```
grouperSession = GrouperSession.startRootSession();
 grantPriv("test:school:apps:atlassian:groups:some-group", "test:school:apps:atlassian:admin:admins",
AccessPrivilege.ADMIN); grantPriv("test:school:apps:atlassian:groups:another-group", "test:school:apps:
atlassian:admin:admins", AccessPrivilege.ADMIN);
```

If you want to set privileges in grouper, then do a SQL script like this:

```
SELECT CONCAT(CONCAT('grantPriv("test:school:apps:atlassian:groups:', groupname), '", "test:school:apps:
atlassian:admin:admins", AccessPrivilege.ADMIN);') AS gsh_script FROM groupbase SELECT CONCAT(CONCAT('grantPriv
("test:school:apps:atlassian:groups:', groupname), '", "test:school:apps:atlassian:admin:readers",
AccessPrivilege.READ);') AS gsh_script FROM groupbase SELECT CONCAT(CONCAT('grantPriv("test:school:apps:
atlassian:groups:', groupname), '", "test:school:apps:atlassian:admin:updaters", AccessPrivilege.UPDATE);') AS
gsh_script FROM groupbase
```

Then do the memberships:

```
addMember(group name, subject id)
```

Run thie sql script

```
SELECT CONCAT(CONCAT(CONCAT(CONCAT('addMember("test:school:apps:atlassian:groups:', group_name), '", "'),
user_name), '");') AS gsh_script FROM membershipbase WHERE user_name <> 'admin' AND user_name NOT LIKE '%.%'
AND user_name NOT LIKE '%@%' AND user_name NOT LIKE '%\_%' ;
```

Once you are migrated, keep a backup of groupbase and membershipbase, and truncate those tables

Migrate Confluence groups and memberships

Generate a GSH script for groups

```
SELECT CONCAT(CONCAT(CONCAT(CONCAT('addGroup("test:school:apps:atlassian:groupsConfluence", "', groupname), '",
"'), groupname), '");') AS gsh_script FROM groups;
```

This script looks like this, add a grouper session and run with GSH (gsh.sh file.script):

```
subject = findSubject("atlassianGrouper/school.edu");
 grouperSession = GrouperSession.start(subject);
 addGroup("test:school:apps:atlassian:groupsConfluence", "admin_systems_financials_users",
"admin_systems_financials_users");
 addGroup("test:school:apps:atlassian:groupsConfluence", "admissions_admin", "admissions_admin");
 addGroup("test:school:apps:atlassian:groupsConfluence", "ait", "ait");
 addGroup("test:school:apps:atlassian:groupsConfluence", "ait_directors", "ait_directors"); ...
```

Generate a GSH script for privileges

```
SELECT CONCAT(CONCAT('grantPriv("test:school:apps:atlassian:groupsConfluence:', groupname), '", "test:school:
apps:atlassian:admin:admins", AccessPrivilege.ADMIN);') AS gsh_script FROM groups;
```

Run the script (gsh.sh file.script):

```
subject = findSubject("atlassianGrouper/school.edu");
 grouperSession = GrouperSession.start(subject);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admin_systems_financials_users", "test:school:apps:
atlassian:admin:admins", AccessPrivilege.ADMIN);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admissions_admin", "test:school:apps:atlassian:admin:
admins", AccessPrivilege.ADMIN);
 grantPriv("test:school:apps:atlassian:groupsConfluence:ait", "test:school:apps:atlassian:admin:admins",
AccessPrivilege.ADMIN); ...
```

Generate script:

```
SELECT CONCAT(CONCAT('grantPriv("test:school:apps:atlassian:groupsConfluence:', groupname), '", "test:school:
apps:atlassian:admin:readers", AccessPrivilege.READ);') AS gsh_script FROM groups;
```

Run the GSH

```
subject = findSubject("atlassianGrouper/school.edu");
 grouperSession = GrouperSession.start(subject);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admin_systems_financials_users", "test:school:apps:
atlassian:admin:readers", AccessPrivilege.READ);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admissions_admin", "test:school:apps:atlassian:admin:
readers", AccessPrivilege.READ); ...
```

Generate script:

```
SELECT CONCAT(CONCAT('grantPriv("test:school:apps:atlassian:groupsConfluence:', groupname), '", "test:school:
apps:atlassian:admin:readers", AccessPrivilege.READ);') AS gsh_script FROM groups;
```

Run the GSH:

```
subject = findSubject("atlassianGrouper/school.edu");
 grouperSession = GrouperSession.start(subject);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admin_systems_financials_users", "test:school:apps:
atlassian:admin:updaters", AccessPrivilege.UPDATE);
 grantPriv("test:school:apps:atlassian:groupsConfluence:admissions_admin", "test:school:apps:atlassian:admin:
updaters", AccessPrivilege.UPDATE);
```

Generate membership script:

```
SELECT CONCAT(CONCAT(CONCAT(CONCAT('addMember("test:school:apps:atlassian:groupsConfluence:', the_group.
groupname), '", "'), the_user.name), '");') AS gsh_script FROM groups AS the_group, users AS the_user,
local_members AS the_membership WHERE the_group.id = the_membership.groupid AND the_user.id = the_membership.
userid AND the_user.name NOT LIKE '% %' AND the_user.name <> 'admin' AND the_user.name NOT LIKE '%.%' AND
the_user.name NOT LIKE '%@%' AND the_user.name NOT LIKE '%\_%' ;
```

Run the GSH script

```
subject = findSubject("atlassianPenngroups/medley.isc-seo.upenn.edu");
grouperSession = GrouperSession.start(subject);
addMember("test:school:ait:apps:atlassian:groupsConfluence:admin_systems_financials_users", "jsmith");
addMember("test:school:ait:apps:atlassian:groupsConfluence:admin_systems_financials_users", "asmith");
addMember("test:school:ait:apps:atlassian:groupsConfluence:admin_systems_financials_users", "bsmith");
addMember("test:school:ait:apps:atlassian:groupsConfluence:admin_systems_financials_users", "csmith");
addMember("test:school:ait:apps:atlassian:groupsConfluence:some_other_confluence_group", "asmith");
```

sdf

Logging

The connector has excellent logging, each method will log in DEBUG mode, including if it was cached, and how long the method call took. Change this in the log4j.properties of jira/confluence

```
#log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.GrouperAccessProvider = DEBUG
#log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.GrouperLoggingProfileProviderWrapper = DEBUG
log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.GrouperProfileProvider = DEBUG
#log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.GrouperCredentialsProvider = DEBUG
#log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.externalAuthentication = DEBUG
#log4j.logger.edu.internet2.middleware.grouperAtlassianConnector.xmpp = DEBUG

#log4j.logger.org.jivesoftware.smack = DEBUG
```

The logs look like this:

```
2010-12-18 15:53:22,516 TP-Processor8 DEBUG admin 953x51x1 x4b0lo 1.2.3.4 /plugins/servlet/streams [internet2.
middleware.grouperAtlassianConnector.GrouperProfileProvider] operation: getPropertySet, username: myuser,
retrievedFromPropertySetCache: true, fullName: Chris Hyzer ADMIN, email: mchyzer@school.edu, timeMillis: 0
2010-12-18 15:53:22,516 TP-Processor8 DEBUG admin 953x51x1 x4b0lo 1.2.3.4 /plugins/servlet/streams [internet2.
middleware.grouperAtlassianConnector.GrouperProfileProvider] operation: list, retrievedFromUserCache: true,
resultList: Size 262: asmith, bsmith, csmith, dsmith..., timeMillis: 0
2010-12-18 15:53:22,516 TP-Processor8 DEBUG admin 953x51x1 x4b0lo 1.2.3.4 /plugins/servlet/streams [internet2.
middleware.grouperAtlassianConnector.GrouperProfileProvider] operation: handles, username: myuser,
retrievedFromPropertySetCache: true, fullName: Chris Hyzer ADMIN, email: mchyzer@school.edu, timeMillis: 0
```

Not, in the logs above, you see the wrapper providers, if you want to see how atlassian handles things, configure the atlassian default connector to be the provider in the grouper.client.properties, and configure the osuser.xml to point to the grouper wrapper, and it will print to the logs (INFO level) what the atlassian connector (or other connector) is doing.

Access provider

The atlassian root folder in grouper is where the atlassian groups are sandboxed. I believe you could have descendants in that folder, put a group name in atlassian with a colon in it, but I havent tried it. The access provider allows you to add / remove memberships from the Atlassian (e.g. Jira) admin console, or from Grouper. You can create/delete groups in the atlassian UI also. Note, obviously the WS user that atlassian uses needs access to the all the relevant groups.

Profile provider

This allows users id's, names, and emails to be retrieved from Grouper. Note that if there is a user in Atlassian that is not resolvable in Grouper, that you would need to add it to the grouper.client.properties file as an autoadd user. Note that users cannot be added/edited/deleted from the Atlassian admin console since Grouper does not control that.

Unit tests

Every method of the profile interfaces are unit tested. To get these to work, you need to enter information in the grouper.client.properties (described above). You can run these tests against a real installation and it will not negatively affect anything (shouldnt do this in prod though unless you are careful 🙂)

Caching

Atlassian calls methods frequently, so the connector does a lot of caching. The default is to cache for 10 minutes. If the write action is performed in Atlassian admin console, the caches are cleared. Otherwise it could take 10 minutes for Grouper actions to propagate to Atlassian (or however you configure in grouper.client.properties). Note that if you have lots of groups, and lots of members, the cache refreshes can take some time (5-20 seconds?) For this reason you could have a long cache timeout, and use XMPP notifications for real time updates.

```
# number of minutes to cache reads
 # defaults to 10.  Note, crank this up  to 25 hours if you are doing XMPP notifications
 atlassian.cache.minutes = 10

 # number of minutes to cache profile reads
 # defaults to 10
 atlassian.cache.profile.minutes = 20

 # each cache has a failsafe cache, so that if grouper is down, and the data has been loaded,
 # since atlassian has been started, the stale verison of the data can be retrieved
 atlassian.cache.failsafe.hours = 48
```

Note, in the config above, there is a failsafe cache. This means that the last successful call to Grouper is cached, and stored until it times out or until another successful call. So if Grouper is down, and Jira/Confluence is not restarted, it should be fine. Note that the calls to Grouper are batched so if there is a group query, all groups and memberships are retrieved.

XMPP notifications for real time updates

If you want to set the cache timeout to something long (1 day?), then you can enable XMPP notifications from the grouper-loader server to the jira or confluence system. Note, when a message comes from Grouper, the grouper.client.properties specifies a number of seconds to buffer the request. The cache clear will take place normally in the background so users will not notice. Also, there is a croned full refresh that happens in the background so users do not notice a delay. Here is an example of the grouper.client.properties config (note: you need grouper client 1.6.4+, if it is not released, you can build it from SVN or ask the Grouper team for a build of it):

```
################################
## XMPP client settings
## Note: you need the smack.jar in your classpath, see the grouper xmpp wiki for usage
## https://spaces.at.internet2.edu/display/GrouperWG/Grouper+XMPP+notifications+v1.6.0
################################


## general xmpp configuration
grouperClient.xmpp.server.host = jabber.school.edu
grouperClient.xmpp.server.port = 5222
grouperClient.xmpp.user = atlassianjabber
# note, pass can be in an external file with morphstring
grouperClient.xmpp.pass = /home/dir/pass/grouper/grouperjabberClient.pass
grouperClient.xmpp.resource = jiraProd
# note, you need the exact id and resource here or it wont match
grouperClient.xmpp.trustedMessagesFromJabberIds = grouperjabber@school.edu/grouperServer

# if true, then each quartz trigger name will be unique
# do this for atlassian since it doesnt do quartz right, and wont delete or reuse old triggers
grouperClient.xmpp.uniqueQuartzTriggerNames = true

# just need a group here, any group, preferably not too large
grouperClient.xmpp.job.atlassian.groupNames = school:atlassian:groupsJira:jira-administrators
grouperClient.xmpp.job.atlassian.elfilter = (event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq
'MEMBERSHIP_ADD') && event.membershipType eq 'flattened' && event.groupName.startsWith('school:apps:atlassian:
groupsJira')
grouperClient.xmpp.job.atlassian.handlerClass = edu.internet2.middleware.grouperAtlassianConnector.xmpp.
GrouperAtlassianXmppHandler
# set this to reload_group or incremental if not reload on each event
grouperClient.xmpp.job.atlassian.eventAction = incremental
# how often a full refresh should occur regardless of events
grouperClient.xmpp.job.atlassian.fullRefreshQuartzCronString =
grouperClient.xmpp.job.atlassian.subjectAttributeNames =
# subjects wont notify in not in these sources, comma separated, or blank for all
grouperClient.xmpp.job.atlassian.requireSources =
# subjects wont notify if they dont have a non blank value for these attributes, or blank for all
grouperClient.xmpp.job.atlassian.requireAttributes =
```

Here is an example of the grouper-loader.properties that will send the XMPP:

```
###################################
## XMPP notifications
## (note, uncomment the consumer class and cron above)
## this will get grouper ws getMembers rest lite xmp:
## http://anonsvn.internet2.edu/cgi-bin/viewvc.cgi/i2mi/trunk/grouper-ws/grouper-ws/doc/samples/getMembers
/WsSampleGetMembersRestLite_xml.txt?view=log
###################################

## general xmpp configuration
xmpp.server.host = jabber.school.edu
xmpp.server.port = 5222
xmpp.user = grouperjabber
# note, pass can be in an external file with morphstring
xmpp.pass = /home/dir/pass/grouper/grouperjabber.pass
xmpp.resource = grouperServer


changeLog.consumer.atlassianProd.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbConsumer
changeLog.consumer.atlassianProd.quartzCron = 0 * * * * ?
#match all events that are add or remove membership, only flattened, if the subject is from the jdbc source,
with the loginid attribute and in the folder test:xmppGroups
changeLog.consumer.atlassianProd.elfilter = (event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq
'MEMBERSHIP_ADD') && event.membershipType == 'flattened' && event.groupName =~ '^school\\:apps\\:atlassian\\:.
*$'
changeLog.consumer.atlassianProd.publisher.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.
EsbXmppPublisher
#add in the recipients
changeLog.consumer.atlassianProd.publisher.recipient = atlassianjabber@school.edu/jiraProd,
atlassianjabber@schools.edu/confluenceProd
#changeLog.consumer.atlassianProd.publisher.addSubjectAttributes =
```

Note, you also need to add these jars to the confluence WEB-INF/lib: ezmorph.jar, json-lib.jar, smack.jar (copies are in the grouperAtlassianConnector project)