26-April-2010, Spring Member Meeting
Chris Hyzer, Grouper developer

What’s new with Grouper
What’s new with Grouper - Agenda

- XMPP integration
  - XMPP and the Grouper loader
  - XMPP and the Grouper client
- Kuali Rice integration
  - Rice groups
  - Rice subjects
  - Automatic workflow provisioning
- What’s new with Grouper 1.6
  - Jira runthrough
Grouper XMPP
XMPP messaging for “real time” updates

1. UI, WS, API, loader
2. Grouper Registry
3. Grouper loader change log
4. Grouper XMPP change log consumer
5. XMPP server
6. XMPP recipient
7. Local authorization list
8. Could be Grouper client

XMPP server filters events, then sends XML messages to the XMPP recipient. The recipient could be the Grouper client. Every minute, the UI, WS, API, and loader poll the Grouper Registry for changes. Optional, the recipient might need more data or do a full refresh.
XMPP step 1: registry is updated

Grouper client

UI
WS
API
loader
GSH

Grouper Registry
XMPP step 2,3: Grouper loader change log

grouper-loader.properties (run every minute)

changeLog.consumer.xmpp.class=edu...xmpp.XmppChangeLogConsumer

changeLog.consumer.xmpp.quartzCron = 0 * * * * ?
XMPP stem 4,5: filter and notify

grouper-loader.properties (Jabber server configuration)

```properties
## general xmpp configuration
xmpp.server.host = jabber.school.edu
xmpp.server.port = 5222
xmpp.user = penngroupsjabber
# note, pass can be in an external file with morphstring
xmpp.pass = /opt/pass/jabber.pass
xmpp.resource = grouperServer
```
grouper-loader.properties (send membership changes to Jabber ID’s)

## In this case, these configs have job name "myJobName"
## Add as many jobs as you want with different names
# groupNames is optional, though you need groupNames

```properties
xmpp.job.myJobName.groupNames =
xmpp.job.myJobName.groupRegex = ^test\\:xmppGroups\\:.*$
xmpp.job.myJobName.subjectAttributeNames = email,firstName
xmpp.job.myJobName.sendToXmppJabberIds = abc@school.edu/resource
xmpp.job.myJobName.requireSources = pennperson
xmpp.job.myJobName.requireAttributes = pennname
```
XMPP step 6: the message

```xml
<XmppMembershipChange>
  <action>ADD_MEMBER</action>
  <groupName>test:xmppGroups:test1</groupName>
  <jobName>myJobName</jobName>
  <subjectAttributeNames>
    <string>PENNAME</string>
  </subjectAttributeNames>
  <xmppSubject>
    <attributeValues>
      <string>mchyzer</string>
    </attributeValues>
    <id>12345678</id>
    <sourceId>pennperson</sourceId>
  </xmppSubject>
</XmppMembershipChange>
```

Could be Grouper client
XMPP step 7,8: process the message

- Option 1: Use the incoming payload to incrementally change local list
- Option 2: Ignore or append to payload with Grouper WS
  - If need more info
  - If not trust message payload
  - If not trust message reliability
Grouper client: XMPP mode

• Start client as daemon, it will listen on XMPP channel for events and kick off logic
• Full refresh on startup
• Quartz cron is included for periodic full refresh
Grouper client: XMPP mode config

grouperClient.properties (XMPP server settings)
grouperClient.xmpp.server.host = jabber.school.edu
grouperClient.xmpp.server.port = 5222
grouperClient.xmpp.user = grouperJabberUser
grouperClient.xmpp.pass = /opt/pass/grouperClientJabber.pass
grouperClient.xmpp.resource = grouperClient
# note, you need the exact id and resource here or it won't match
grouperClient.xmpp.trustedMessagesFromJabberIds = user@school.edu/resource, user2@school.edu/resource2
groupClient.properties (XMPP job settings, can have multiple jobs)

gc.xmpp.job.myJobName.groupNames = test:xmppGroups:test1

gc.xmpp.job.myJobName.handlerClass = \_..grouper.XmppFileHandler

gc.xmpp.job.myJobName.eventAction = incremental|reload_group

gc.xmpp.job.myJobName.fullRefreshQuartzCronString = 0 0 5 * * ?

gc.xmpp.job.myJobName.fileHandler.targetFile = c:/temp/targetFile.txt

gc.xmpp.job.myJobName.fileHandler.filePrefix = c:/temp/filePrefix.txt

gc.xmpp.job.myJobName.fileHandler.iteratorEl =
    ${subject.attribute['pennname']}$space$

gc.xmpp.job.myJobName.fileHandler.fileSuffix = c:/temp/fileSuffix.txt

gc.xmpp.job.myJobName.subjectAttributeNames = pennname

gc.xmpp.job.myJobName.requireSources = pennperson

gc.xmpp.job.myJobName.requireAttributes = pennname
Grouper XMPP demo:

• UI membership update
• XMPP change log consumer
• Sends XMPP message
• Grouper client processes the message
• Updates a .htaccess file incrementally
• Note, in addition to the grouperClient.jar, you need:
  • smack.jar (XMPP)
  • quartz.jar (cron)
  • Smack and quartz dependencies: commons-logging.jar, commons-collections.jar, jta.jar
Grouper Kuali Rice integration
Kuali

- Higher education open source Java software
- Rice: middleware
- Kuali Financials
- Kuali Student
- Coeus: Research management
- OLE: Library application
Kuali Rice

• Middleware used by other Kuali and non-Kuali products
• KIM: Kuali Identity Management
• KSB: Kuali service bus
• KEN: Kuali enterprise notification
• KEW: Kuali enterprise workflow
• Components
  • UI
  • SOAP web services
  • Web framework
  • eDocLite: declarative workflow applications
Kuali Rice overridable services

• Group service
  • getMembers, hasMember, assignMember, etc
• Identity service
  • getPersonById, getPersonByPrincipal, etc
• Permissions service
• etc
Kuali Rice overridable services

- Rice request
- Rice server
- grouperRice.jar
- grouperClient.jar
- Grouper.client.properties
- Kuali DB
- Grouper Registry
- Grouper WS server
Why connect Rice to Grouper?

- Already use Grouper
- Want richer group model
- Add workflow to Grouper
- Quickstart integration to IdM
How to connect Rice to Grouper?

• Add two jars to Rice (grouperRice.jar and grouperClient.jar)
• Add / configure grouper.client.properties
• Configure Rice spring override to group and/or identity service
• Setup a Grouper folder for the “Rice root”
Rice Grouper workflow

1. Initiator starts workflow
   - On login to Rice, get subject details

2. Routes to approver group
   - Get members to route
   - One in group approves

3. Routes to approver group2
   - Add a member to a Grouper group/role and/or assign permissions

4. Final
   - Archive the document data, and workflow history

Grouper web services

Kuali DB

Grouper Registry
Grouper Rice demo

- Penn access management eDocLite form
- Initiator starts workflow
- Selects own supervisor to route to
- Route to the admins group
- Route to the implementors group (if manual actions need to occur)
What’s new with Grouper 1.6
What’s new with Grouper 1.6 besides...

- XMPP
- Kuali Integration
What’s new with Grouper 1.6

- GRP-190: Migrate to subversion
- GRP-384: Support SQL server
- GRP-390: Virtual subject attributes
  - Create subject attributes based on regular expressions
- GRP-406: Read-only mode for Grouper
  - Useful for upgrades / data migrations
- GRP-413: Flattened memberships
  - More efficient notifications
  - Groundwork for future read performance enhancements
New import/export

• Handles all tables/columns (e.g. new attribute framework)
• Normalized XML format
• Streams on input and output to reduce memory problems
• Maintains last edited dates, last edited person, etc
What’s new with Grouper 1.6 (WS)

• Note: all web service enhancements are REST/SOAP, Lite/batched, and included in command line grouperClient
• GRP-242: Multiple privilege management service
  • For group/folder privileges, READ/UPDATE/etc
  • E.g. assign READ/UPDATE to groups a:b,b:c for users 123,345
• GRP-356: Non-immediate query filter
  • Old way was effective or composite
  • E.g. get all members in group a:b who have a membership that cannot be unassigned directly
What’s new 1.6 (WS) (continued)

• GRP-357: Filter members by source
  • E.g. get all members of a group that all University people (not groups or kerberos principals)
• GRP-358: Allow client calls by Group ID (in addition to group name)
• GRP-359: Get groups by stem
  • E.g. get all groups a subject is a member of in a certain folder, under a folder, or matching a pattern
• GRP-360: Delete all members of group from client
• GRP-367: findGroups accepts group names and ID’s
What’s new 1.6 (WS) (continued)

• GRP-369: Get memberships service
  • E.g. get all reasons why a subject is in a group
• GRP-370: Can createParentFoldersIfNotExist on create group
• GRP-372: Get subjects service
  • E.g. get subject by ID or identifier or search criteria
  • Now you can write a Grouper UI on WS!
• GRP-420: Get attributes service (new attribute framework). On groups, folders, members, etc
• GRP-415: Assign / unassign attributes service (new attribute framework)
What’s new 1.6 (WS) (continued)

• GRP-416: Get permissions service (new central permissions management module)
  • E.g. get all permissions for a subject in an app
• GRP-417: Assign (or unassign) permissions (new central permissions management service)
What’s new 1.6 (UI)

• GRP-376: Subject picker component
  • E.g. find an employee in a certain org unit, and when selected, insert their ID in the underlying web app

• GRP-377: UI access control lists
  • E.g. in various parts of UI, assure user is in a certain group

• GRP-378: UI lite default delete multiple

• GRP-379: UI lite import in textarea option

• GRP-382: Skin and customize text on Grouper lite UI (without requiring changes on server)
  • E.g. manage members page linked from external application that looks like the app, and has applicable text and help
Cardiff University Grouper+ESB use case

- Identity Management system uses Mule Enterprise Service Bus to provide interfaces for event-driven, real-time data sync with applications, databases and directories.
- Drools rules make decisions for account provisioning, including what groups an account should be a member of.
- Grouper is authoritative for groups, so need a way to add/remove group membership via the ESB.
- Also need to sync membership add/remove from Grouper to LDAP directory in (near) real-time via the ESB.
- Objective is to complete provisioning actions within 2 minutes from data being changed in an authoritative system.
Events synchronised between applications via the ESB
Synchronisation over HTTP(S) and XMPP

Events packaged as JSON and dispatched over appropriate interface
# grouper-loader.properties

- # xmpp sync test, consumer side, sending events to ESB
  - changeLog.consumer.xmppTest.class = edu.internet2.middleware.grouper.changeLog.esb.EsbConsumer
  - changeLog.consumer.xmppTest.elfilter = event.eventType eq 'GROUP_DELETE' || event.eventType eq 'GROUP_ADD' || event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq 'MEMBERSHIP_ADD'
  - changeLog.consumer.xmppTest.publisher.class = edu.internet2.middleware.grouper.changeLog.esb.EsbXmppPublisher
  - changeLog.consumer.xmppTest.publisher.server = procopiue
  - changeLog.consumer.xmppTest.publisher.port = 5222
  - changeLog.consumer.xmppTest.publisher.username = groupersend
  - changeLog.consumer.xmppTest.publisher.password = groupersend
  - changeLog.consumer.xmppTest.publisher.recipient = esb@procopius
  - changeLog.consumer.xmppTest.publisher.addSubjectAttributes = cardiffidmanaffiliation

- # xmpp sync test, listener side, receiving events from ESB
  - esb.listeners.xmpp.enable = true
  - esb.listeners.xmpp.server = propcopius
  - esb.listeners.xmpp.port = 5222
  - esb.listeners.xmpp.username = grouper
  - esb.listeners.xmpp.password = grouper
  - esb.listeners.xmpp.sendername = esb@procopius
Example XMPP data flow

1. JSON string sent to grouper account on XMPP server

2. Message picked up by client running in Grouper daemon, membership of group removed

3. Changelog consumer notified of change, which is published to XMPP server

4. ESB picks up message and adds a user account to a group in LDAP
What’s new with Grouper

26-April-2010, Spring Member Meeting
Chris Hyzer, Grouper developer

For more information, visit www.internet2.edu