Grouper Provisioning: Locally & Cloud

Bill Thompson, Lafayette College
Chris Hyzer, University of Pennsylvania
Bert Bee-Lindgren, Georgia Tech
Happy Boy IAM Architecture 2.0

- **HR/SIS/etc identity sources**
  - Institutional identity provisioning, account lifecycle, identifier management

- **midPoint**
  - Create/manage Institutional Identity/Account
  - institutional identity provisioning, account lifecycle, identifier management

- **Grouper UI**
  - What labels go on which people? (reference groups)
  - Group Owners
  - Grouper Loader

- **Grouper UI/CLI**
  - Configure policy via group math and rules (account and membership groups)
  - IAM Admin

- **Grouper**
  - access policy, audit, compliance
  - Reference groups represent the current state of membership for all subjects as known to the enterprise. They are used to configure access management policy and provide the means for automated provisioning of groups and accounts as well as audit and compliance.
  - Account and membership groups represent authorization policy. Effective membership configured via group math or rules generates change notifications.

- **Grouper Subject Source**
  - Keeps application accounts and group memberships in sync and consistent with policy

- **OpenLdap**
  - Identity/credential store

- **Provisioning Engine**
  - routes provisioning messages based on change of membership or subject attributes. Resolves subject attributes if necessary.

- **CAS/Shibboleth**
  - WebSSO, SAML Federation

- **PWM**
  - self-service account/password management

- **Various target services**
VPN Access

- VPN access requires membership in an LDAP group (**vpn2**) and at least one more LDAP group representing 4 levels of access:
  - exceptions
  - students
  - facstaff
  - netadmins
Reference Groups and Access Policy

thomp sow is a **direct member** of

- `lc:ref:role:banner:admin_ft`

  which is a **direct member** of

  - `lc:ref:role:employee_proxies`

    which is a **direct member** of

    - `lc:app:vpn:vpn_roles:facstaff_include`

      which is a **composite factor** minus `facstaff_exclude` of

      - `lc:app:vpn:vpn_roles:facstaff`

        which is a **direct member** of

        - `lc:app:vpn:vpn_include`

          which is a **composite factor** minus `vpn_exclude` of

          - `lc:app:vpn:vpn`
Reference Groups and Access Policy

thompson is a direct member of

\[ \text{lc:org:admin:its:di} \]

which is a direct member of

\[ \text{lc:app:vpn:vpn\_roles:netadmins\_include} \]

which is a composite factor minus netadmins\_exclude of

\[ \text{lc:app:vpn:vpn\_roles:netadmins} \]

which is a direct member of

\[ \text{lc:app:vpn:vpn\_include} \]

which is a composite factor minus vpn\_exclude of

\[ \text{lc:app:vpn:vpn} \]
Now What?

- Bill's account is now in the Grouper groups that represent the VPN access policies.
- His account is a member of `vpn2`, `facstaff`, and `netadmins`.
- What mechanisms reflect this policy into target systems?
Out of Grouper! Changelogger

- Lafayette has a simple Grouper changelogger written in Jython and built on the Shell Wrappers for Grouper
  https://github.com/wgthom/groovysh4grouper
- The group membership changelogger sends messages to a RabbitMQ Exchange via AMQP protocol.
RabbitMQ Exchange

- Group membership changelogger sends messages to the exchange via AMQP protocol.
- An exchange is like a post office where messages are sent. The exchange uses rules called **bindings** to determine where to deliver a message, based on its routing key.
- The Lafayette changelogger uses the routing key **kiki.grouper**.
## Exchange: grouper_exchange

### Overview

#### Message rates (chart: last minute)

![Message rates chart]

### Details

<table>
<thead>
<tr>
<th>Type</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>topic</td>
</tr>
</tbody>
</table>

### Bindings

#### To  | Routing key | Arguments | Unbind
--- | --- | --- | ---
| kiki_memb_q | kiki.* | Unbind |
Exchange to Provisioner

---

```
{
  "name": "VPN",
  "group": "lc:app:vpn:vpn",
  "include_attributes": false,
  "route_key": "vpn"
}
{
  "name": "VPN Roles",
  "stem": "lc:app:vpn:vpn_roles",
  "recursive": false,
  "include_attributes": false,
  "route_key": "vpn"
}
{
  "name": "Default",
  "group": "*",
  "discard": true
}
```
Accounts and Memberships

- Some remote systems only care about membership or access control lists.
- Sometimes provisioning a remote system means creating and maintaining an account on the remote system.
- The Lafayette College provisioning architecture allows for both types of provisioning.
2016 TechEx Grouper Duo and Box provisioning
Grouper Duo and Box provisioning agenda

- Look at two provisioners
- At the specifics
- But also compare the general designs
- Duo
- Box
- Btw, if you are interested in using either of these or building something like this let me know
Grouper Duo provisioning description

- Change log consumer
- Sync a folder in grouper and use the extensions of groups in the folder as group names in Duo
- Syncs group description
- Periodical full sync daemon (nightly?)
- Change log consumers are real time incremental
Why Grouper Duo provisioning?

• Groups in Duo can be required for integrations
• Layer of authorization and deprovisioning for your systems
• e.g.
  • Grouper group for your IT department
  • Require that group for your IT dept VPN, RDP, SSH
  • Someone not in that group can’t access
Grouper Duo architecture
Acquire Duo WS credentials from Duo admin

# these are properties to add to grouper-loader.properties
# group duo admin domain name credentials

grouperDuo.adminIntegrationKey =
grouperDuo.adminSecretKey =
grouperDuo.adminDomainName =
Configure other grouper-loader.properties for Duo

# folder for duo, the name in duo will be the extension here
grouperDuo.folder.name.withDuoGroups = a:b:c
# put the comma separated list of sources to send to duo
grouperDuo.sourcesForSubjects = someSource
# subject attribute for the duo username (e.g. netId)
grouperDuo.subjectAttributeForDuoUsername = id
# delete duo groups which don't exist in grouper
grouperDuo.deleteGroupsInDuoWhichArentInGrouper = true
Configure Duo incremental and full sync

# configure the duo change log consumer
  changeLog.consumer.duo.class = edu...GrouperDuoChangeLogConsumer
# the quartz cron is a cron-like string. it defaults to every minute on the minute
  changeLog.consumer.duo.quartzCron =
# Schedule full refresh
  otherJob.duo.class = edu...GrouperDuoFullRefresh
  otherJob.duo.quartzCron = 0 0 5 * * ?
Grouper Box provisioning description

- Change log consumer sends messages to queue/topic
- Messaging listener processes messages uses WS
- Sync one folder, grouper extensions are box group names
- Periodic full sync daemon (nightly?)
- SAML attributes for just-in-time provisioning of new users
- Only MEMBER roles are managed in box, not ADMIN
- Handful of jars with grouper client, runs as unix process
Grouper Box group issue

- If someone is a group admin in box, they are also automatically a user admin
- The Grouper box integration does not have this issue
- Group ADMINs who are people are not needed
  - (no Group ADMINs are needed!)
Why Grouper provisioning and SAML JIT?

• Are users themselves already in box or provisioned on first login?
• If provisioned already, then you don’t need SAML
• Users in box can read a group membership list and see who is in it (with Grouper integration)
• SAML and Grouper are useful by themselves but can add more value if used together
Why Grouper Box provisioning?

- Share resources in Box with lists of people maintained in Grouper
- Automatically fix deprovisioning gap in Box
  - If you need an entitlement to use box at your institution
  - SSO makes sure they can’t login
  - Though box-sync is still active, needs to be reset
Why use messaging?

• The Box admins at your institution might want to closely protect the WS credentials
• Does not want those creds running on the Grouper server
• Can be notified remotely when changes occur on Box groups in grouper
• Scales better
• Another dev group can manage the provisioning
Current state of Grouper box integration

- 70% complete, in progress
- Full sync works
- Messages are sent on applicable Grouper actions
- Finish incremental provisioning
- Implement box sync deprovisioning
Configure messaging queue

Make messaging queue, grant access to the box user (in this case keep it simple and use GrouperSystem, though you would make your own user and not user GrouperSystem)

```java
GrouperSession grouperSession = GrouperSession.startRootSession();
GrouperBuiltinMessagingSystem.createQueue("box_queue");
Subject subject = SubjectFinder.findById("GrouperSystem");
GrouperBuiltinMessagingSystem.allowSendToQueue("box_queue", subject);
GrouperBuiltinMessagingSystem.allowReceiveFromQueue("box_queue", subject);
```
Configure messaging change log consumer

changeLog.consumer.boxEsb.class = edu...EsbConsumer
changeLog.consumer.boxEsb.quartzCron = 0 * * * * ?
changeLog.consumer.boxEsb.elfilter = event.groupName =~ '^.+\:groups\.\:.+$' &&
  (event.eventType eq 'GROUP_DELETE' || event.eventType eq 'GROUP_ADD' ||
   event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq
   'MEMBERSHIP_ADD')
changeLog.consumer.boxEsb.publisher.class = edu...EsbMessagingPublisher
changeLog.consumer.boxEsb.publisher.messagingSystemName =
grouperBuiltinMessaging
# queue or topic
changeLog.consumer.boxEsb.publisher.messageQueueType = queue
changeLog.consumer.boxEsb.publisher.queueOrTopicName = box_queue
Configure Box WS credentials

- Create a box application
- OAUTH 2.0 JWT server authentication
- Create public/private key
- Authorize the application
- Note all the settings and configure in grouper.client.properties
Configure Box WS credentials (continued)

# authentication settings for WS in grouper.client.properties

grouperBox.privateKeyFileName =
grouperBox.privateKeyPass =
grouperBox.publicKeyId =
grouperBox.enterpriseId =
grouperBox.clientId =
grouperBox.clientSecret =
Configure Box grouper.client.properties

# put groups in here which go to box, the name in box will be the extension here
  grouperBox.folder.name.withBoxGroups = a:b:c
# put the comma separated list of sources to send to box
  grouperBox.sourcesForSubjects = someSource
# either have id for subject id or an attribute for the box username (e.g. netId)
  grouperBox.subjectAttributeForBoxUsername = id
  grouperBox.fullSync.quartzCron = 0 0 5 * * ?
Configure Box grouper.client.properties (continued)

# if using include/exclude in grouper then exclude these groups in box
grouperBox.ignoreGroupSuffixes = _systemOfRecord, _includes, _excludes,
   _systemOfRecordAndIncludes, _includesMinusExcludes
# if there is a suffix…
grouperBox.subjectIdSuffix = @upenn.edu
# if require…
grouperBox.requireGroupInGrouper = community:active
# if delete groups which aren't in grouper (will lose permissions if assigned)
grouperBox.deleteGroupsInBoxWhichArentInGrouper = false
Configure Box grouper.client.properties (continued)

# remove memberships in box in groups which aren't in grouper
   # note if not all groups in box are managed in grouper, this should be false
   grouperBox.removeMembershipsInBoxWhenGroupsArentInGrouper = true
# delete groups incremental when groups removed in grouper
   grouperBox.deleteGroupWhenGroupDeletedInGrouper = false
# deprovision box sync for inactive users
   grouperBox.deprovisionBoxSyncForInactiveUsers = false
Grouper PSPNG

- PSPNG 2.3.0 (Spring)
- Status
- Fast Provisioning
- [PSPNG Cookbook]
Release 2.3.0 - provisioning

• Provisioning Service Provider Next Generation (PSPNG)
  • Simplify Configuration
  • Increase performance
• 2.3.0 Provisioning Targets
  • LDAP Groups
  • Active Directory Groups
  • LDAP Attributes (Entitlements)
• Other features:
  • Incremental and full refresh
  • Group selection: By folder or group, or both
  • Higher-level and higher-performance programming API
Grouper Provisioning: Status

• Progress is reaccelerating
• Automated integration testing paying off
  • More realistic & varied
  • Avoiding regressions
• Immediate Goals:
  • A couple more bugs
  • Patches
  • Support
  • Performance measurement
Fast Provisioning: Background

• Theory:
  • Deltas In: Changelog
  • Deltas Out: Modifications

• Practice:
  • Supporting data is needed
    • Subject details
    • Group info
  • Individual Modifications: Slow
Fast Provisioning: Mechanisms

• Supporting Data
  • Bulk Read
    • Just read everything
    • Query entire batch (Big ORs or INs)
  • Cache it

• Modifications
  • Combine changes
    • Adding 50 people to a group: One Group Mod
    • Adding a person to 50 groups: One Entitlement Mod
Fast Provisioning: PSPNG

• Provisioner Subclasses: Easy
  • Mostly single-group, single-subject methods
  • Bulk read [Optional]
  • Template: Native caching objects

• Provisioner Harness: Harder
  • Bookkeeping
  • Change Merging & Chunking
  • Incremental & Full Sync
PSPNG Cookbook - ToC

• LDAP Groups
  • Active Directory
  • Posix Groups
  • Group of Unique Names
• LDAP Attributes
  • eduPersonEntitlement
• Wiki:
  https://spaces.internet2.edu/display/Grouper/Grouper+Provisioning%3A+PSPNG
Provisioning Opportunities (post 2.3.0)

- Automated Integration Tests
- Deprovisioning Safety Nets
- Performance Measurements
- Grouper Messaging
- More Targets: Suggestions?
## Back to Provisioning

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grouper Changelog Reader In Grouper Daemon</td>
<td>Can be out of the box. Fewer moving parts.</td>
</tr>
<tr>
<td>(PSPNG, Duo)</td>
<td>Constrained to Grouper Data.</td>
</tr>
<tr>
<td>Messaging Using Grouper Java API Possibly in</td>
<td>More flexible about triggering.</td>
</tr>
<tr>
<td>Grouper Daemon</td>
<td>Retry Messages</td>
</tr>
<tr>
<td></td>
<td>Slightly fewer moving parts. API Performance &amp; Power</td>
</tr>
<tr>
<td>Messaging Using Grouper WS Not in Grouper Daemon</td>
<td>Provisioning by app owner. Read/Trigger lots of</td>
</tr>
<tr>
<td>(Box &amp; Lafayette)</td>
<td>Subject Data</td>
</tr>
<tr>
<td></td>
<td>Triggering &amp; Retry Flexibility</td>
</tr>
</tbody>
</table>