

Grouper + ActiveMQ + GAP

Carnegie Mellon University

August 2014

General Flow (outbound)

1. A change happens to a group
2. GAP Changelog Consumer for grouper grabs changes and sends to ActiveMQ queue
3. An AMQ consumer processor (running some place in some language) gets messages from the queue, in order, and processes the change based on rules of the target it is servicing. You write one consumer per target. You can have many consumers for a single target. Lots of flexibility.
4. Remove message from queue.
5. Done.

Some rationale

- We have a complex arrangement of groups with hierarchies of groups and composites and composites as part of hierarchies. These groups represent user populations and are used to define who gets what services like GoogleApps, kerberos, AFS, ActiveDirectory and so on.
- A user added to a single group cascades into dozens of groups very rapidly. Speed is important.

GAP Changelog Consumer

- By default, a changelog consumer runs every 60 seconds. We run every 15 seconds.
- Changes get pushed to ActiveMQ using the AMQ java library.

grouper.changelog.dispatcher

- We push to a queue called `grouper.changelog.dispatcher` in order for us to filter which queues/consumers/apps see which groups. For example, we don't push non-AD or course groups over to AD.
- The consumer for `grouper.changelog.dispatcher` reads from this queue and dispatches the msgs to other queues based on a config file.
- Gives us lots of flexibility.

GAP

- The dispatcher sends to 389.groups, 389.isMemberOf and ad.groups
 - Dispatcher also sends to other groups as needed
- The 3 consumers all run the same code with different conf
- GAP normally processes single change operation messages (incremental mode). A message describes addGroup, removeGroup, addMember, removeMember and so on.
- GAP also supports fullsync messages. A single message includes the entire membership of the group. Fullsync messages perform single LDAP MOD operations to be very efficient.
- Performance stats logged (msgs + ldap ops / sec)

GAP 389.groups

- We have a consumer dedicated to handling 389.groups – static group objects only.
- We do not have 389 configured for referential integrity to produce isMemberOf internally like AD handles memberOf
- We are configured for flat group naming instead of bushy style.
- We will be modifying this consumer to support aggregating many incremental changes into a single MOD where possible. Will be significant performance improvement by peeking into the queue.

GAP 389.isMemberOf

- We have a consumer dedicated to handling 389.isMemberOf – attributes in the user object referencing group memberships.
- We do not have 389 configured for referential integrity to produce isMemberOf internally like AD handles memberOf
- This consumer also supports fullsync messages and is configured for flat naming instead of bushy style.

GAP ad.groups

- ad.groups is an instance of GAP configured for ActiveDirectory. Based on our experiences, it configures groups bushy style with samAccountName, CN and other attributes appropriate for use with AD.
- AD internally handles memberOf
- It took a while to deal with AD idiosyncrasies



AMQ admin + monitoring

- What follows are some screen shots of AMQ admin and monitoring
- We use LDAP authentication for AMQ and queue definitions (names and roles admin/read/write)
- LDAP groups are used for AMQ access control
- The hawtio dynamic page automatically updates and easy to use; now separate for AMQ distro
- We have 2 AMQ servers in a broker network using the internal KahaDB. Very reliable.
- <https://activemq.apache.org>

Inbound – injecting into Grouper with AMQ

- We also have an async capability
- App sends messages to a queue
 - AMQ queues are authenticated, of course
- Queue consumer reads from queue and pushes into Grouper via grouper-ws

AMQ monitoring (static pages)



Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Support

Queue Name

Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
389.groups	0	1	645325	645325	Browse Active Consumers atom rss	Send To Purge Delete
389.ismemberof	0	1	644773	644773	Browse Active Consumers atom rss	Send To Purge Delete
389.provisioner.errors	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete
ActiveMQ.Statistics	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete
ActiveMQ.Statistics.Broker	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete
ad.groups	0	1	19749	19749	Browse Active Consumers atom rss	Send To Purge Delete
ad.provisioner.errors	0	0	0	0	Browse Active Consumers atom rss	Send To Purge Delete

Queue Views

Graph

XML

Topic Views

XML

Subscribers Views

XML

Useful Links


Documentation

FAQ

Downloads

Forums

AMQ: browsing a queue




Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Browse devo2.apps.oim.groups

Message ID ↑	Correlation ID	Persistence	Priority	Redelivered	Reply To	Timestamp	Type	Operations
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-409:1:1:3:1		Persistent	4	false		2014-03-24 10:30:03:479 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-409:1:1:3:2		Persistent	4	false		2014-03-24 10:30:03:652 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-429:1:1:3:1		Persistent	4	false		2014-03-25 11:23:03:318 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-433:1:1:4:1		Persistent	4	false		2014-03-25 12:50:00:421 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-433:1:1:4:2		Persistent	4	false		2014-03-25 12:50:00:437 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-433:1:1:4:3		Persistent	4	false		2014-03-25 12:51:00:455 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-433:1:1:4:4		Persistent	4	false		2014-03-25 12:52:00:474 EDT		Delete
ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-433:1:1:4:5		Persistent	4	false		2014-03-25 12:52:00:487 EDT		Delete

AMQ: viewing a message

5000
http

Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Headers ↑

Correlation ID	
Destination	queue://dev02.apps.oim.groups
Expiration	0
Group	Apps:OIM-DEV02:exchange
Message ID	ID:grouper-t01.andrew.cmu.edu-60383-1395339302107-409:1:1:3:1
Persistence	Persistent
Priority	4
Redelivered	false
Reply To	
Sequence	0
Timestamp	2014-03-24 10:30:03:479 EDT
Type	

Properties

JMSXGroupID	Apps:OIM-DEV02:exchange
JMSXGroupSeq	0

Message Actions

Delete

Copy

Move

-- Please select --

Message Details

```
{"operation": "addMember", "name": "Apps:OIM-DEV02:exchange", "memberId": "jiemath"}
```

AMQ: dynamic page (hawtio)

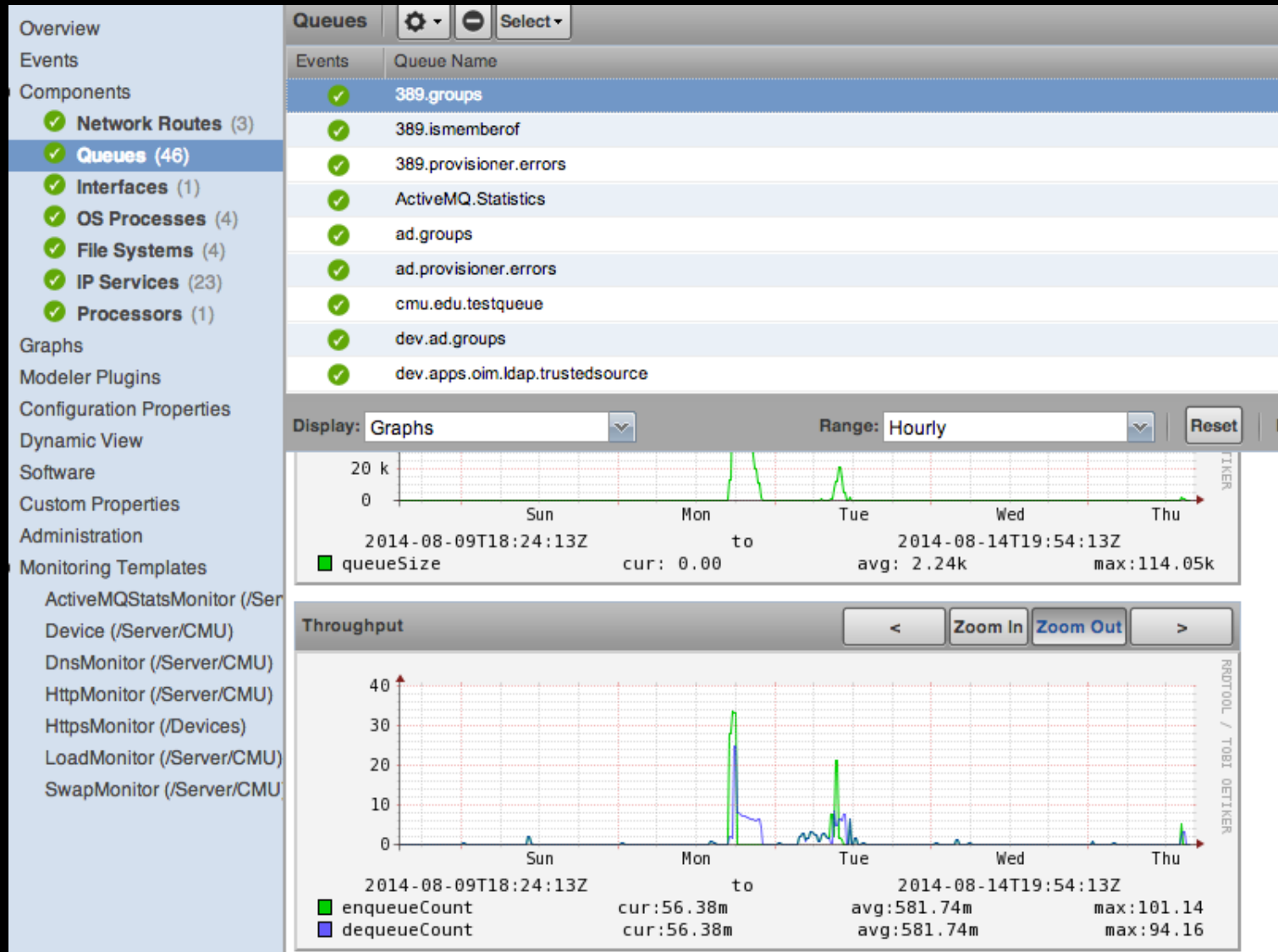
The screenshot shows the hawtio web interface for ActiveMQ. The top navigation bar includes links for Dashboard, Health, JMX, Connect, and Logs. The left sidebar displays the 'ActiveMQ Tree' with a tree view of the broker's structure. The main content area shows a table of queue statistics, with a search bar and tabs for Diagram, Create, Attributes, Operations, and Chart. The table lists various queues and their corresponding statistics.

Name	Queue ...	Produc...	Consu...	Enqueu...	Dequeu...	Mem
389.groups	0	0	1	645325	645325	0
389.ismemberof	0	0	1	644773	644773	0
389.provisioner.errors	0	0	0	0	0	0
ActiveMQ.Statistics	0	0	0	0	0	0
ActiveMQ.Statistics.Broker	0	0	0	0	0	0
ad.groups	0	0	1	19749	19749	0
ad.provisioner.errors	0	0	0	0	0	0
cmu.edu.testqueue	0	0	0	0	0	0
dev.ad.groups	0	1	1	569	569	0
dev.apps.oim ldap.trustedsource	0	0	1	0	0	0
dev.apps.oim ldap.trustedsource...	0	0	0	0	0	0

Zenoss/nagios monitoring

- We have some scripts to monitor queue depth and ActiveMQ server health via Zenoss network monitoring system.
 - Zenoss uses Nagios style scripts, easy to monitor
- If a queue depth $> N$, send alerts. Alert again when $< N$
- Alert on pool % used thresholds on all queues

Zenoss graphing...



Pulling it all together...

