

Grouper Messaging System development guide

Wiki Home	Deploying Grouper	Grouper Guides	Grouper Deployment Guide	Community Contributions	Internal Developer Resources
---------------------------	-----------------------------------	--------------------------------	--	---	--

This page provides sample code showing how to develop with the [Grouper Messaging System](#)

[GSH to manage built in messaging](#)

[GSH to send / receive messages](#)

Integrating Grouper messaging with a new messaging system (e.g. if an AcmeMQ adapter is not implemented, this is how to implement one)

- Interface `GrouperMessagingSystem` defines a messaging system. Implement this to integrate Grouper messaging with a messaging system
- `GrouperBuiltinMessagingSystem` implements `GrouperMessagingSystem` - Grouper's implementation of the messaging interface
- `GrouperMessagingEngine` is the Grouper API to send a message to various messaging systems

Configure messaging systems in the `grouper.client.properties`

```
#####
## Grouper Messaging System
#####

# name of messaging system which is the default
grouper.messaging.default.name.of.messaging.system = grouperBuiltinMessaging

# name of a messaging system. note, "grouperBuiltinMessaging" can be arbitrary
grouper.messaging.system.grouperBuiltinMessaging.name = grouperBuiltinMessaging
# class that implements edu.internet2.middleware.grouperClient.messaging.GrouperMessagingSystem
grouper.messaging.system.grouperBuiltinMessaging.class = edu.internet2.middleware.grouper.messaging.
GrouperBuiltinMessagingSystem

# name of a messaging system. note, "myAwsMessagingSystem" can be arbitrary
# grouper.messaging.system.myAwsMessagingSystem.name = aws
# class that implements edu.internet2.middleware.grouperClient.messaging.GrouperMessagingSystem
# grouper.messaging.system.myAwsMessagingSystem.class =
```

The messaging interface is:

```

/**
 *
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouperClient.messaging;

/**
 * Represents the methods that a messaging system
 * needs to support
 */
public interface GrouperMessagingSystem {
    /**
     * send a message to a queue name. Note, the recipient could be a
     * queue or a topic (generally always one or the other) based on the
     * implementation of the messaging system. Messages must be delivered
     * in the order that collection iterator designates. If there is a problem
     * delivering the messages, the implementation should log, wait (back off)
     * and retry until it is successful.
     * @param grouperMessageSendParam has the queue or topic, and the message(s) and perhaps args
     * @return result
     */
    public GrouperMessageSendResult send(GrouperMessageSendParam grouperMessageSendParam);

    /**
     * this will generally block until there are messages to process. These messages
     * are ordered in the order that they were sent.
     * @param grouperMessageReceiveParam grouper messaging receive param
     * @return a message or multiple messages. It will block until there are messages
     * available for this recipient to process
     */
    public GrouperMessageReceiveResult receive(GrouperMessageReceiveParam grouperMessageReceiveParam);

    /**
     * tell the messaging system that these messages are processed
     * generally the message system will use the message id. Note, the objects
     * sent to this method must be the same that were received in the
     * receiveMessages method. If there is a problem
     * delivering the messages, the implementation should wait (back off)
     * and retry until it is successful. Alternatively the message should be
     * returned to queue, returned to end of queue, or sent to another queue
     * @param grouperMessageAcknowledgeParam
     * @return result
     */
    public GrouperMessageAcknowledgeResult acknowledge(GrouperMessageAcknowledgeParam
grouperMessageAcknowledgeParam);
}

```

Messaging listener

A messaging listener is a daemon job which will check a message system queue for messages and act on them (calling an interface)

```
#####
## Messaging listener using the messaging API
#####

# note, change "messagingListener" in key to be the name of the listener.  e.g. messaging.listener.
myAzureListener.class
# extends edu.internet2.middleware.grouper.messaging.MessagingListenerBase
# this listener will just print out messages: edu.internet2.middleware.grouper.messaging.MessagingListenerPrint
#

#messaging.listener.messagingListener.class = edu.internet2.middleware.grouper.messaging.MessagingListener
#messaging.listener.messagingListener.quartzCron = 0 * * * * ?
#messaging.listener.messagingListener.messagingSystemName = grouperBuiltinMessaging
#messaging.listener.messagingListener.queueName = abc
#messaging.listener.messagingListener.numberOfTriesPerIteration = 3
#messaging.listener.messagingListener.pollingTimeoutSeconds = 18
#messaging.listener.messagingListener.sleepSecondsInBetweenIterations = 0
#messaging.listener.messagingListener.maxMessagesToReceiveAtOnce = 20
# if there are 20 messages to receive at once, then do this 50 times per call max
#messaging.listener.messagingListener.maxOuterLoops = 50
```

Sample messaging listener implementation

```

/**
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.messaging;
import java.util.Collection;
import edu.internet2.middleware.grouper.changeLog.ChangeLogEntry;
import edu.internet2.middleware.grouperClient.messaging.GrouperMessage;
import edu.internet2.middleware.grouperClient.messaging.GrouperMessageAcknowledgeParam;
import edu.internet2.middleware.grouperClient.messaging.GrouperMessageAcknowledgeType;
import edu.internet2.middleware.grouperClient.messaging.GrouperMessagingEngine;

/**
 *
 */
public class MessagingListenerPrint extends MessagingListenerBase {
    /**
     *
     */
    public MessagingListenerPrint() {
    }
    /**
     * @see edu.internet2.middleware.grouper.messaging.MessagingListenerBase#processMessages(java.lang.String,
     java.lang.String, java.util.Collection, edu.internet2.middleware.grouper.messaging.MessagingListenerMetadata)
     */
    @Override
    public void processMessages(String messageSystemName, String queue,
        Collection<GrouperMessage> grouperMessageList,
        MessagingListenerMetadata messagingListenerMetadata) {

        for (GrouperMessage grouperMessage : grouperMessageList) {
            try {

                String json = grouperMessage.getMessageBody();

                //try to convert to change log entry
                try {
                    Collection<ChangeLogEntry> changeLogEntries = ChangeLogEntry.fromJsonToCollection(json);
                    for (ChangeLogEntry changeLogEntry : changeLogEntries) {
                        System.out.println("Change log entry: " + changeLogEntry.getChangeLogType().getChangeLogCategory() +
                            " -> " + changeLogEntry.getChangeLogType().getActionName() + ", " + changeLogEntry.getId());
                    }
                    System.out.println("Change log entry: " + json);
                } catch (Exception e) {
                    System.out.println("Not change log entry: " + grouperMessage.getId() + ", " + json);
                }

                //mark it as processed
                GrouperMessagingEngine.acknowledge(new GrouperMessageAcknowledgeParam()
                    .assignAcknowledgeType(GrouperMessageAcknowledgeType.mark_as_processed)
                    .assignQueueName(queue).assignGrouperMessageSystemName(messageSystemName)
                    .addGrouperMessage(grouperMessage));

            } catch (Exception e) {
                messagingListenerMetadata.registerProblem(e, "Problem in message: " + grouperMessage.getId(),
                    grouperMessage.getId());
                break;
            }
        }
    }
}

```

Messaging listener that uses ChangeLogConsumerBase implementations

If you have a change log consumer and you want to have it process messages, use this:

```
#####  
## Messaging listener using the change log consumer API  
#####  
  
# note, change "messagingListenerChangeLogConsumer" in key to be the name of the listener. e.g. messaging.  
listener.myAzureListener.class  
#  
# keep this class to be MessagingListenerToChangeLogConsumer  
#messaging.listener.messagingListenerChangeLogConsumer.class = edu.internet2.middleware.grouper.messaging.  
MessagingListenerToChangeLogConsumer  
#messaging.listener.messagingListenerChangeLogConsumer.changeLogConsumerClass = edu.internet2.middleware.  
grouper.messaging.SomethingExtendsChangeLogConsumerBase  
#messaging.listener.messagingListenerChangeLogConsumer.quartzCron = 0 * * * * ?  
#messaging.listener.messagingListenerChangeLogConsumer.messagingSystemName = grouperBuiltinMessaging  
#messaging.listener.messagingListenerChangeLogConsumer.queueName = abc  
#messaging.listener.messagingListenerChangeLogConsumer.numberOfTriesPerIteration = 3  
#messaging.listener.messagingListenerChangeLogConsumer.pollingTimeoutSeconds = 18  
#messaging.listener.messagingListenerChangeLogConsumer.sleepSecondsInBetweenIterations = 0  
#messaging.listener.messagingListenerChangeLogConsumer.maxMessagesToReceiveAtOnce = 20  
# if there are 20 messages to receive at once, then do this 50 times per call max  
#messaging.listener.messagingListenerChangeLogConsumer.maxOuterLoops = 50
```

Change log consumer that sends to messaging

This is a change log consumer that will send change log entries to a queue or topic with a JSON format that can be easily converted back to ChangeLogEntries

```
#####  
## Messaging integration with change log, send change log entries to a messaging system  
#####  
# note, change "messaging" in key to be the name of the consumer. e.g. changeLog.consumer.myAzureConsumer.class  
#changeLog.consumer.messaging.class = edu.internet2.middleware.grouper.changeLog.ChangeLogConsumerToMessage  
#changeLog.consumer.messaging.quartzCron = 0 * * * * ?  
#changeLog.consumer.messaging.messagingSystemName = grouperBuiltinMessaging  
# queue or topic  
#changeLog.consumer.messaging.messageQueueType = queue  
#changeLog.consumer.messaging.queueOrTopicName = abc
```

Change log JSON

```
{
  "event": [
    {
      "changeLogTypeId": "7db1fb2d34944668bc7d4485f1c7854b",
      "contextId": "7e4ca9c1dfd14111a60c8b057e5ae5aa",
      "createdOnDb": 1458577616745723,
      "sequenceNumber": 473,
      "changeLogTypeCategory": "privilege",
      "changeLogTypeAction": "addPrivilege",
      "field_id": "435e7190d1814d86a26e6bbd70e9bca3",
      "field_privilegeName": "read",
      "field_subjectId": "GrouperAll",
      "field_sourceId": "g:isa",
      "field_privilegeType": "access",
      "field_ownerType": "group",
      "field_ownerId": "7d09cfff2b444db696a0771b224081ef",
      "field_ownerName": "test:testGroup3",
      "field_memberId": "e6e154ea21f64c0d9ecb9f4137ab1ac3",
      "field_fieldId": "5da0071e39ff4fb2a17cec73ac25efb3",
      "field_membershipType": "flattened"
    }
  ]
}
```

Convert Change Log Entry to and from json

```
String json = changeLogEntry.toJson(true);
ChangeLogEntry newEntry = ChangeLogEntry.fromJsonToCollection(json).iterator().next();
```

Messaging ESB change log consumer, configure in grouper-loader.properties

```
#####
## Messaging integration with ESB, send change log entries to a messaging system
#####
# note, change "messagingEsb" in key to be the name of the consumer. e.g. changeLog.consumer.myAzureConsumer.
class
#changeLog.consumer.messagingEsb.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbConsumer
#changeLog.consumer.messagingEsb.quartzCron = 0 * * * * ?
#changeLog.consumer.messagingEsb.filter = event.eventType eq 'GROUP_DELETE' || event.eventType eq 'GROUP_ADD'
|| event.eventType eq 'MEMBERSHIP_DELETE' || event.eventType eq 'MEMBERSHIP_ADD'
#changeLog.consumer.messagingEsb.publisher.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.
EsbMessagingPublisher
#changeLog.consumer.messagingEsb.publisher.messagingSystemName = grouperBuiltinMessaging
# queue or topic
#changeLog.consumer.messagingEsb.messageQueueType = queue
#changeLog.consumer.messagingEsb.publisher.queueOrTopicName = abc
```

Configure builtin messaging cleanup jobs in grouper-loader.properties

```
#####
## grouper builtin messaging cleanup cron
#####

#quartz cron-like schedule for grouper messaging daemon.
#leave blank to disable this, the default is every hour, 10 minutes after the hour
#this daemon does cleanup on the builtin messaging table
changeLog.builtinMessagingDaemon.quartz.cron = 0 10 * * * ?

# after three days of not consuming messages, delete them, if -1, dont run this daemon
grouper.builtin.messaging.deleteAllMessagesMoreThanHoursOld = 72

# after three hours of having processed messages, delete them. Note, if this is -1 just delete when marking
processed
grouper.builtin.messaging.deleteProcessedMessagesMoreThanMinutesOld = 180
```

Test messaging

Run the test: `edu.internet2.middleware.grouper.messaging.GrouperBuiltinMessagingSystemTest`

Set this in `log4j.properties` to see debug and performance info:

```
log4j.logger.edu.internet2.middleware.grouper.messaging.GrouperBuiltinMessagingSystemTest = DEBUG
```

Example to get messages

```
String messageSystemConfigName = "someConfigName";
String queueName = "someQueue";
GrouperMessageReceiveResult grouperMessageReceiveResult = GrouperMessagingEngine.receive(new
GrouperMessageReceiveParam()
    .assignGrouperMessageSystemName(messageSystemConfigName)
    .assignGrouperMessageQueueParam(new GrouperMessageQueueParam().assignQueueOrTopicName(queueName).
assignQueueType(GrouperMessageQueueType.queue))
    .assignMaxMessagesToReceiveAtOnce(20));

for (GrouperMessage grouperMessage : grouperMessageReceiveResult.getGrouperMessages()) {
    String body = grouperMessage.getMessageBody();
    //do something with message
    GrouperMessagingEngine.acknowledge(new GrouperMessageAcknowledgeParam()
        .assignGrouperMessageSystemName(messageSystemConfigName)
        .assignGrouperMessageQueueParam(new GrouperMessageQueueParam().assignQueueOrTopicName(queueName).
assignQueueType(GrouperMessageQueueType.queue))
        .assignAcknowledgeType(GrouperMessageAcknowledgeType.mark_as_processed)
        .assignGrouperMessages(GrouperClientUtils.toSet(grouperMessage)));
}
```

See Also

[Grouper Messaging System](#)

[Grouper Built In Messaging](#)

[Message Format Detail](#)

[Message Format Config Example](#)

