Sample change log consumer

Sample change log consumer

Configure in grouper-loader.properties

```
# note, change "myApp" in configId to be the name of the consumer
changeLog.consumer.myApp.class = edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbConsumer

# quartz cron, run every minute
changeLog.consumer.myApp.quartzCron = 0 * * * * ?

# el filter to filter events before they reach the consumer
changeLog.consumer.myApp.elfilter = (event.eventType eq 'MEMBERSHIP_UPDATE' || event.eventType eq
'MEMBERSHIP_DELETE' || event.eventType eq 'MEMBERSHIP_ADD') && (event.groupName =~ '^app\\:groups\\:.*$')

changeLog.consumer.myApp.publisher.class = edu.school.it.EsbConsumerForApp

# some ad hoc param for the consumer if needed
changeLog.consumer.myApp.publisher.groupName = a:b:c:Group
```

Code

```
package edu.school.it;

import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;

import edu.internet2.middleware.grouper.SubjectFinder;
import edu.internet2.middleware.grouper.app.loader.GrouperLoaderConfig;
import edu.internet2.middleware.grouper.app.loader.db.Hib3GrouperLoaderLog;
import edu.internet2.middleware.grouper.app.provisioning.GrouperProvisioningProcessingResult;
import edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbEvent;
import edu.internet2.middleware.grouper.changeLog.esb.consumer.EsbEventContainer;
import edu.internet2.middleware.grouper.esb.listener.EsbListenerBase;
import edu.internet2.middleware.grouper.esb.listener.ProvisioningSyncConsumerResult;
import edu.internet2.middleware.grouper.util.GrouperUtil;
import edu.internet2.middleware.subject.Subject;

public class EsbConsumerForApp extends EsbListenerBase {

  /** logger */
  private static final Log LOG = GrouperUtil.getLog(EsbConsumerForApp.class);

  @Override
  public boolean dispatchEvent(String eventJsonString, String consumerName) {
    // we arent using this method
    throw new RuntimeException("Not implemented");
  }

  @Override
  public void disconnect() {
    // maybe not needed

  }

  private Map<String, Object> debugMap = new LinkedHashMap<String, Object>();

  @Override
  public ProvisioningSyncConsumerResult dispatchEventList(
      List<EsbEventContainer> esbEventContainers,
```

```
      GrouperProvisioningProcessingResult grouperProvisioningProcessingResult) {

    // this object maps to daemon (loader) log in database
    Hib3GrouperLoaderLog hib3GrouperLoaderLog = this.getEsbConsumer().getChangeLogProcessorMetadata().
getHib3GrouperLoaderLog();

    // keep track of counts
    int countToProcess = 0;
    int countToNotProcess = 0;

    try {

      // add things to this map to print in loader log to show on ui
      this.debugMap.put("eventCount", GrouperUtil.length(esbEventContainers));
      ProvisioningSyncConsumerResult provisioningSyncConsumerResult = new ProvisioningSyncConsumerResult();

      for (EsbEventContainer esbEventContainer : GrouperUtil.nonNull(esbEventContainers)) {

        EsbEvent esbEvent = esbEventContainer.getEsbEvent();

        // organize things in a "shouldProcess" method
        if (shouldProcess(esbEventContainer)) {
          countToProcess++;
          try {
            // note you could batch things if the api to the application supports it for efficiency
            processEvent(esbEventContainer);
          } catch (RuntimeException e) {

            // error handling on each event
            String errorMessage = "Error processing event " + esbEvent;
            LOG.error(errorMessage, e);
            this.debugMap.put("error", errorMessage);
            this.debugMap.put("exception", GrouperUtil.getFullStackTrace(e));
            // ignore or end?
            return provisioningSyncConsumerResult;

          }
        } else {
          countToNotProcess++;
        }

        // if we are retrying failures, then return the last success
        provisioningSyncConsumerResult.setLastProcessedSequenceNumber(esbEventContainer.getSequenceNumber());

      }
      return provisioningSyncConsumerResult;
    } finally {
      debugMap.put("countToProcess", countToProcess);
      debugMap.put("countToNotProcess", countToNotProcess);
      String debugMapString = GrouperUtil.mapToString(this.debugMap);
      LOG.debug(debugMapString);
      // save the debug map in the database
      hib3GrouperLoaderLog.appendJobMessage(debugMapString);
    }

  }

  private void processEvent(EsbEventContainer esbEventContainer) {

    EsbEvent esbEvent = esbEventContainer.getEsbEvent();

    // Get the userName from the changelog entry
    String userName = esbEvent.getSubjectId();

    // Try to find userName in LDAP
    Subject ldapSubject = SubjectFinder.findByIdAndSource(userName, "ldap", false);

    // If not in LDAP, don't try to proceed.
    // This will catch group membership records that are not actually "subjects"
    if (ldapSubject == null) {
      return;
```

```
    }

    Hib3GrouperLoaderLog hib3GrouperLoaderLog = this.getEsbConsumer().getChangeLogProcessorMetadata().
getHib3GrouperLoaderLog();

    // Otherwise, proceed
    String login = userName + "@school.edu";

    // Check if the user already exists
    AppUser appUser = null;

    appUser = someLogicToGetUser(login);

    switch (esbEventContainer.getEsbEventType()) {
      case MEMBERSHIP_DELETE:
        if (appUser != null) {
          hib3GrouperLoaderLog.addDeleteCount(1);
          deleteUser(login);
        }
        break;
      case MEMBERSHIP_ADD:
      case MEMBERSHIP_UPDATE:
        boolean isUpdate = false;
        // Create a new Box user
        if (appUser == null) {
          appUser = createUser(login);
          hib3GrouperLoaderLog.addInsertCount(1);
        } else {
          // Ensure existing user is active
          if (activateUser(appUser)) {
            isUpdate = true;
          }
        }

        // Try to add schoolMail values as aliases to user
        Set<String> schoolMail = ldapSubject.getAttributeValues("schoolMail");
        if (schoolMail != null) {
          if (setAliases(schoolMail, login, appUser)) {
            isUpdate = true;
          }
        }

        if (isUpdate) {
          hib3GrouperLoaderLog.addUpdateCount(1);
        }
        break;
      default :
        throw new RuntimeException("Not expecting event type: " + esbEventContainer.getEsbEventType());
    }


  }

  private boolean shouldProcess(EsbEventContainer esbEventContainer) {
    // generally the ESB config in loader will filter events appropriately, if not, do that here
    // e.g. the below can easily be dont in grouper-loader.properties for this consumer.  This is just an
example
    EsbEvent esbEvent = esbEventContainer.getEsbEvent();
    switch (esbEventContainer.getEsbEventType()) {
      case MEMBERSHIP_ADD:
      case MEMBERSHIP_DELETE:
      case MEMBERSHIP_UPDATE:
        String groupName = GrouperLoaderConfig.retrieveConfig().propertyValueStringRequired("changeLog.consumer.
app.grouperGroupIdPath");
        return StringUtils.equals(esbEvent.getGroupName(), groupName);
      default:
         return false;
    }
  }
```

```
}
```